Відповідно до навчальної програми з інформатики для учнів загальноосвітніх навчальних закладів

Козолуп Є.В.

Програмування в школі. Мова Python

Навчальний посібник

8 клас

Суми 2017

УДК 004.45
ББК 32.973+74.262
К 59

Відповідно до навчальної програми з інформатики для учнів загальноосвітніх навчальних закладів

Козолуп Є.В.

Програмування в школі. Мова Python : Навчальний посібник. 8 клас. / Є.В. Козолуп. - Суми, 2017. - 82 с.

У навчальному посібнику розглядаються теми з розділу "Основи подійно- та об'єктно-орієнтованого програмування" за навчальною програмою для учнів 8 класу на прикладі мови програмування Python.

Матеріал посібника розділений на 15 лекцій, кожна з яких містить теоретичний матеріал, проілюстрований великою кількістю наочних прикладів (кодів програм) з детальним поясненням, контрольні запитання та практичні завдання для самостійного опрацювання.

Призначений для учнів загальноосвітніх шкіл, вчителів. Рекомендується як посібник і для самостійного вивчення мови програмування Python.

ЗМІСТ

Майбутньому програмісту!

Шановний восьмикласнику, ти вивчаєш інформатику вже не перший рік, знаєш можливості багатьох програм, за допомогою яких можеш створювати різноманітні та цікаві проекти. У цьому навчальному році ти маєш змогу навчитися створювати власне програмне забезпечення.

Я сподіваюся, що цей посібник допоможе тобі зробити перші кроки у світ програмування. Вміст даного видання поділено на лекції, для більш зручного сприйняття, в кінці яких є перелік контрольних запитань та практичних завдань. Це допоможе тобі краще засвоїти матеріал, поданий у лекціях. Звертай велику увагу на примітки та вчитуйся в кожен рядок.

Успіхів тобі!

Лекція 1. Поняття програми та мови програмування. Мова Python

1.1 Поняття програми

Сьогодні, у золотий вік технологій, мало хто обходиться без комп'ютера. Практично в кожного вдома є персональний комп'ютер. Чи то громіздкий стаціонарний, чи переносний ноутбук, чи маленький планшет - усе це відноситься до комп'ютерів, а, як відомо, жоден комп'ютер не буде працювати без програмного забезпечення. *Найважливішим набором програм для керування роботи комп'ютера є* ***операційна система***. Та існують також прикладні програми, які не керують процесами комп'ютера, а виконують тільки певні функції. Ви не зможете "зайти в *Інтернет"* без вашого улюбленого *Google Chrome* чи всіма відомого браузера *Opera*; підготувати реферат у школу без *Microsoft Word* або побачити свого далекого родича, який зараз в іншій країні, без *Skype*. Усі ми користуємося цими програмами, але мало хто знає, як їх було створено.

Ми дуже часто чуємо слово «програма», але мало хто до кінця розуміє його значення. ***Комп'ютерна програма*** *– це набір команд для виконання комп'ютером виражені у зрозумілій для комп'ютера формі.* Кожна комп'ютерна програма виконує певну зазначену програмістом дію або алгоритм дій. І в цьому курсі ми розберемося, як саме відбувається процес **програмування** та навчимося самостійно створювати прості програми.

1.2 Мова програмування

Над розробкою кожної вище зазначеної програми працювали сотні програмістів не один рік. Адже ці програми мають складну внутрішню будову та складний процес написання. Кожна програма пишеться певною мовою програмування. ***Мова програмування*** *– це система позначень для опису дії або алгоритму дій, яку повинна виконувати програма; за допомогою цих позначень програма передає завдання комп'ютеру для їх виконання.* Такими

Мал. 1

позначеннями є **програмний код**, який представляє собою набір символів та слів, зазвичай англійською мовою, який описує певний алгоритм. А самих мов програмування існує дуже багато (мал. 1), і це обумовлено тим, що кожна мова виконує призначену їй функцію. Наприклад, існує така мова як *HTML*, основною задачею якої є створення веб-сторінок та їх наповнення. Або *Java, я*кою зазвичай пишуться додатки до смартфонів з операційною системою *Android*, тощо. А ось ми будемо вивчати мову *Python,* яка на сьогодні є дуже популярною серед веб-програмістів, а також у створювачів операційних систем на базі *Linux*, ну, і звичайно, для написання прикладних програм для ПК, але про неї пізніше.

У роботі програмісту допомагають *спеціальні програми для створення програм, а саме* ***середовище розробки програмного забезпечення (IDE)***. Кожна мова програмування має своє середовище розробки програм, хоча бувають і універсальні середовища, які підходять одразу для декількох мов.

Оскільки комп'ютер використовує двійковий код, а програми пишуться кодом, який зазвичай включає в себе англійські слова та символи, не зрозумілі комп'ютеру, то для того, щоб комп'ютер зрозумів цей код, йому потрібні додаткові *засоби переведення програмного коду в машинний*, а саме ***компілятор*** або ***інтерпретатор***. Один із цих засобів обов'язково повинен входити до середовища розробки програм. Головна відмінність цих двох засобів у тому, що *компілятор* переводить програмний код у машинний тільки один раз, створюючи окремий виконавчий файл програми, а *інтерпретатор* буде переводити програмний код кожен раз при його запуску, що набагато довше, ніж при використанні компіляторів.

Найвідомішим середовищем розробки програмного забезпечення є *Блокнот*. Якщо його правильно налаштувати, з нього вийде чудове IDE . Але, на жаль, він не має ані компілятора, ані інтерпретатора і в ньому можна лише написати програмний код та зберегти відповідний файл.

1.3 Мова програмування Python та її середовище розробки програмного забезпечення

Python *— інтерпретована об'єктно-орієнтована мова програмування високого рівня.*

Мал. 2

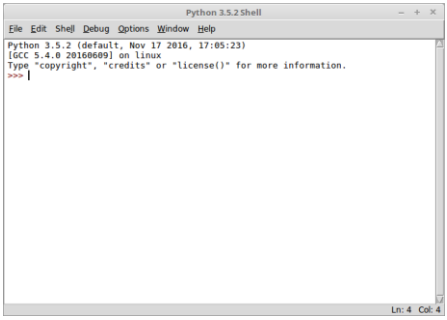
Розробка мови Python була розпочата в кінці **1980-х** років співробітником голландського інституту CWI ***Гвідо ван Россумом*** (мал. 2).

Так як і звичайні мови, мови програмуваня мають свої діалекти (версії). Мова програмування Python має три діалекти: 1.7, 2.7 та 3.6. Перша версія вже не підтримується, а от друга була розроблена дуже вдало, тому її досі використовують у розробці програм.

Python — стабільна та поширена мова. Вона використовується в багатьох проектах та в різних якостях: як основна мова програмування або для створення розширень та інтеграції додатків. На Python реалізована велика кількість проектів, також вона активно використовується для створення прототипів майбутніх програм.

Python використовують такі компанії як Google, Яндекс, Mail.Ru та інші. Мовою написані такі відомі програми як Blander, DropBox, також вона використовувалася у розробці Ubuntu та всесвітньо відомої гри World of Tanks. І це ще далеко не все!

Більш детально про дану мову програмування ви можете дізнатися на офіційному сайті www.python.org.

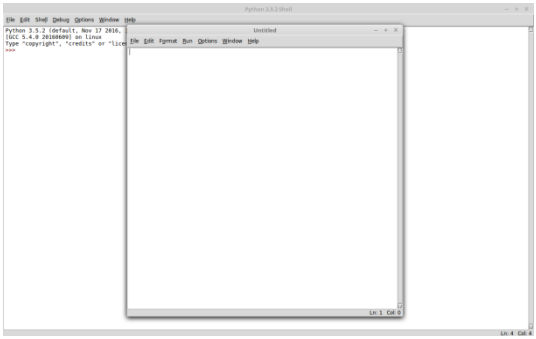


Мал. 3

Як вже було раніше сказано, кожна мова програмування має своє середовище розробки програмного забезпечення, для Python це **IDLE.** Цю програму, можна також завантажити з офіційного сайту.

Для того, щоб запустити середовище розробки програмного забезпечення IDLE потрібно виконати наступні дії: заходимо в меню Пуск→Усі програми→Python 3.5→IDLE. Або, якщо ви користуєтесь операційними системами на базі *Linux* такі: Меню→Програмування→IDLE. Після чого відкриється вікно інтерпретатора даного середовища, який має назву **Python 3.5.2 Shell** (мал.

3). Даний інтерпретатор представлений таким собі командним рядком, в який вже можна вносити прості команди, наприклад: 23*45, і після натискання клавіші *Enter* на наступному рядку з'явиться результат. Як ми бачимо, інтерфейс програми зображений англійською мовою, бо, не має інтерфейсу російською або українською мовою.



Мал. 4

Для того, щоб створити новий файл програми мовою Python, потрібно виконати команду File→New file. Після того відкриється вікно редагування файлу, яке чимось нагадує вікно програми *Блокнот* (мал. 4). В робочу область якої потрібно вносити програмний код. Для того, щоб запустити написану програму в даному інтерпретаторі потрібно спочатку зберегти файл, а потім виконати команду Run→Run Module, або натиснути клавішу F5.

Щоб зберегти файл потрібно скористатися командою File→Save, або натиснути Ctrl+S. Після цього відкриється діалогове вікно збереження файлів і нам потрібно вибрати місце збереження файлу та вказати назву файлу. Основним розширенням файлу Python є розширення **.py*. Ну і для того, щоб відкрити вже створений файл, в вікні IDLE виконуємо команду File→Open. У діалоговому вікні обираємо потрібний файл і натискаємо на кнопку *Відкрити*.

Контрольні запитання

1. Що таке програма?
2. Що таке мова програмування? Які мови програмування ви знаєте?
3. Що таке середовище розробки програмного забезпечення?
4. Що таке Python?
5. Як називається середовище розробки програмного забезпечення, яке використовує мова Python?
6. Як називається інтерпретатор, який використовує IDLE?
7. Яке розширення має файл програми, написаної мовою Python?

Лекція 2. Змінні. Прості типи даних Python. Дії з типами даних

2.1 Змінні

З уроків математики та інформатики ви вже повинні знати про таке поняття як ***змінні***. Як вам відомо, ***змінна*** *— це математична величина, яка має свою назву та певне значення, яке може змінюватися залежно від умов задачі.*

Назва змінної може складатися *з латинських великих та малих літер, може містити цифри, та знак нижнього пробілу.* **Наприклад:** argument1, ArgUmEnt23, slovo_ukr та інші. **Зверніть увагу!** Змінна *arg та Arg* — це зовсім різні змінні, бо перші букви відрізняються за регістром. Немає обмежень щодо змісту назви змінної, головне щоб потім програміст зрозумів за що відповідає кожна з них.

Щоб надати якійсь змінній значення, потрібно після назви змінної поставити знак "=" та ввести її значення. Значенням змінної може бути: число, текст, математична дія з числами чи іншими змінними, або значення може запитуватися у користувача.

Якщо значення змінної число, то після знаку дорівнює *без жодних додаткових знаків ставиться задане число.* **Наприклад**: a=5, mod=76, і т.д. Вирази також записуються без додаткових синтаксичних знаків. **Наприклад:** a=2+b, sum=56+76, тощо. Якщо значення змінної текст, то в залежності від того, за допомогою яких символів він був написаний, будуть використовуватися такі конструкції: *для символів англійського алфавіту, текст береться в одинарні лапки (''), а для українського та російського алфавітів в подвійні ("").* **Наприклад:** a='Hello world!', b="Привіт світ!" і т.п. **Зверніть увагу!** *Числа, або вираз також можна представити як текст, але тоді вони будуть відображати лише рядок із символів.* **Наприклад:** c='145', або sum2='6+4=10' та інше.

2.2 Прості типи даних Python

У високорівневих мовах програмування всі дані належать до певних вбудованих типів. ***Тип даних*** *визначає множину допустимих значень змінної.* До простих типів даних Python відносяться: цілі та дійсні числа і рядки тексту. Детальну інформацію дивіться в таблиці 1.

Назва	Позначення в програмному коді	Елементи типу
Цілі числа	int	Всі натуральні, їм обернені і число 0. **Наприклад:** -1, -578, 0, 45, 98 і т.д.
Дійсні числа	float	Це множина всіх чисел. **Наприклад:** -23.56, 12.0, 0.112, -9.99999, 0, тощо.
Рядок	str	Будь які знаки, які розташовані на клавіатурі комп'ютера. **Наприклад:** 'book №1', '56+14', та інше.

Табл. 1

2.3 Дії з типами даних в Python

Давайте зараз спробуємо зрозуміти, які дії ми можемо виконувати з типами даних у Python.

Дуже часто програмістам доводиться переводити значення одного типу даних в інший, для подальшої роботи з ними. От і ми навчимося застосовувати такий прийом, як ***переведення типів даних*** у мові програмування Python. З цілого числа в текст, з дійсного в ціле і т.д.

Int — цілі числа. *Відображаються просто як числа.* Для перетворення будь-якого значення на ціле число використовується функція **int()**:

```
>>> a=61.7    #a-дійсне число
>>> b=int(a)
61
```

Зверніть увагу! Як було продемонстровано в прикладі, при переведенні дійсного числа в ціле залишається лише ціла частина, дробова "відкидається".

Float — дійсні числа. *Виглядають як 2 числа (ціла і дробова частина), розділені крапкою.* Для перетворення будь-якого значення на дійсне число використовується функція **float()**:

```
>>> a='45'    #a-рядок тексту
>>> b=float(a)
```

```
45.0
```

Str — рядки. *Виглядають як текстові фрагменти будь-якої довжини.* Для перетворення будь-якого значення у рядок використовується функція **str()**:

```
>>> a=56      #a-ціле число
>>> b=str(a)
'56'
```

Якщо ви не знаєте, який тип даних набуває змінна або конкретне значення, можна користуватися функцією **type()**:

```
>>> type('45')
<class 'str'>
```

Контрольні запитання

1. З чого може складатися назва змінної Python?
2. Яких значень може набувати змінна?
3. Які прості типи значень ви вже знаєте? Як вони позначаються в коді? З яких елементів вони складаються?
4. Як переводити типи даних між собою? Назвіть основний принцип.

Практичні завдання

Заповнити таблицю 2 без використання IDLE.

Вхідні дані (a)	Тип даних (a)	Результат(b)	Тип даних (b)	Конструкція переведення
a='120'		b=120		
	float	b='37.73'		
a=(45+5)*2			str	
a=(67+3)/2				b=float(a)
a=12.6		b=12		

Табл. 2

Лекція 3. Введення та виведення даних. Перша програма Python

3.1 Введення даних

Ми вже з вами вміємо присвоювати змінним конкретні значення. А зараз навчимося присвоювати змінним значення, які буде вводити користувач при запуску програми.

Отже в Python 3.5 існує наступна функція для запиту інформації у користувача:

input("Повідомлення**")**

В круглих дужках можна вносити повідомлення з яким у користувача будуть запитуватися дані. **Наприклад:** «Введіть a:», або «Натисніть Enter» та інше.

І давайте спробуємо присвоїти значення даної функції змінній *a*:

a=input("Введіть a:")

Але у нас при обробці даної змінної виникне суперечність, бо ми не вказали тип даних нашої змінної, і для того щоб це зробити ми повинні скористатися вже відомими вам функціями: *int(), float()* або *str().* Для прикладу застосуємо функцію *int()*, отримаємо наступну конструкцію:

a=int(input("Введіть a:"))

Тепер інтерпретатору все зрозуміло: програміст вказує на те, що змінна *a* буде запитуватися у користувача і тип даних цієї змінної – ціле число.

3.2 Виведення даних

Що стосується виведення даних, то існує наступна функція:

print()

Print перекладається з англійського як "друк", тобто вона "друкує" дані, які введені в круглі дужки. Це можуть бути числа, текст взятий в лапки, змінні або вирази. Більше того, можна виводити одразу декілька значень різних типів даних. Для цього значення чи зміні потрібно розділити комою. **Наприклад**:

print(a,"або",b)

Але якщо застосовувати попередню конструкцію, то перед та після слова "або" будуть автоматично встановлюватися пробіли, якщо ви

бажаєте розділити дві змінні певним знаком без пробілів, потрібно користуватися атрибутом **sep,** встановивши його після змінних між якими потрібно вставити символи:

print(a, b, sep=';')

3.3 Перша програма на Python

Тепер ми вже маємо достатньо теоретичних знань для того, щоб створити просту програму переведення типів даних змінних. На вході маємо число *a* - дробове. Програма повинна вивести цілу частину від числа *a*.

Рішення є дуже просте, але спочатку давайте створимо новий файл Python (File —> New File) і збережемо його. У нас відкрилося відповідне вікно для вводу програмного коду. Отже спочатку попросимо ввести користувача будь яке число, а потім виведемо лише цілу частину від даного числа, тобто переведемо дійсне число у ціле:

```
a=float(input())  #конструкцію введення даних користувачем
print(int(a))  #конструкція виводу цілої частини від числа
```

Для більш зручного орієнтування у програмному коді можна створювати помітки як зображено у прикладі за допомогою знаку **#**. Такі помітки можна робити у будь-якому місці файлу, та називаються вони коментарями і ні як не впливають на роботу програми та не виводяться на екран, а потрібні лише розробникам програми для кращого розуміння, що саме виконується в той чи іншій частині коду.

Ну що ж, спробуймо запустити нашу програму (Run —> Run Module) і для прикладу введемо число *555.555*, та натиснемо клавішу Enter. Програма виведе нам результат, а саме *555* (мал. 5).

```
>>> ==================
>>>
555.555
555
```

Мал. 5

Тепер трішки ускладнимо задачу, а саме додамо повідомлення до коду. Повідомлення при запиті змінної та при виведені результату. При введенні даних буде виводитися повідомлення "Введіть дробове число a :", а при виведенні "Ціла частина від числа a". Програмний код виглядає наступним чином:

```
a=float(input("Введіть дробове число a: "))
```

```
print("Ціла частина від числа a:", int(a))
```

Зверніть увагу! Після повідомлення про введення числа воно буде запитуватися одразу, тому для зручності після тексту повідомлення ми поставимо пропуск. З виведенням даних простіше, там пробіл ставиться автоматично.

Збережемо і запустимо наш файл знову, вводячи ті ж самі параметри, і слідкуємо за виведенням результату (мал. 6).

```
>>>
Введіть дробове число a: 555.555
Ціла частина від числа a: 555
>>>
```

Мал. 6

Очікуваного результату досягнуто, отже правильно побудували програму. Наголошу на тому, що ви можете бути впевнені у стабільній роботі програми лише після того, як протестуєте її.

Контрольні запитання

1. Яка команда служить для запиту даних у користувача?
2. За допомогою якої конструкції можна присвоїти змінній будь якого значення?
3. Що потрібно зробити, аби введені данні належали a) до цілих чисел; b) до дійсних чисел; c) до рядку тексту?
4. Як вивести опрацьовані данні на екран?
5. Що таке коментарі та як вони застосовуються в програмному коді?

Практичні завдання

Створіть новий файл Python за допомогою середовища розробки програмного забезпечення IDLE. Створіть програму за даною задачею:

a. на вході маємо дійсне число, яке запитується у користувача. Програма переводить його в ціле та виводить результат (не використовувати додаткові повідомлення);
b. на вході маємо ціле число, яке запитується у користувача. Програма переводить його в текст та виводить результат, додаючи до нього символ "!" в кінець (не використовувати додаткові повідомлення);
c. на вході маємо два числа, які запитуються у користувача (можна

використати додаткове повідомлення), програма виведе ці числа на екран з повідомленням "Ці числа утворюють множину:" в фігурних дужках, та між ними поставить кому;

d. на вході маємо ім'я користувача, яке буде запитуватися із відповідним повідомленням. На виході маємо повідомлення «Привіт» та ім'я користувача.

Лекція 4. Робота з цілими та дійсними числами в Python. Дії з текстом

4.1 Робота з цілими числами

В даній лекції пропоную ознайомитися з діями, які ми можемо робити з ними в Python 3.5. Так як int — це числа, з ними можна виконувати наступні дії:

1) Додавання

Ну з додаванням все просто. Між числами що додаються потрібно поставити знак "+". Якщо обидва числа цілі то і результат буде ціле число:

```
>>> a=int()
>>> b=int()
>>> c=a+b
>>> type(c)
<class 'int'>
```

2) Віднімання

Для віднімання чисел потрібно використовувати знак "-". Якщо обидва числа цілі, то і результат буде ціле число:

```
>>> a=int()
>>> b=int()
>>> c=a-b
>>> type(c)
<class 'int'>
```

3) Множення

Для виконання цієї дії між числами потрібно встановити знак множення "*". І в даному випадку маємо такий результат:

```
>>> a=int()
>>> b=int()
>>> c=a*b
>>> type(c)
<class 'int'>
```

4) Ділення

Для ділення встановлюємо знак "/" між числом яке ділять і числом на яке ділять (не дорівнює 0). **Зверніть увагу!** При діленні цілого на ціле

ми не отримуємо ціле число, а отримаємо дійсне, тобто:

```
>>> a=int()
>>> b=int()  #b не дорівнює 0
>>> c=a/b
>>> type(c)
<class 'float'>
```

5) Піднесення до степеня

Для піднесення числа до степеня потрібно застосувати знак "**". Степінь як ви вже знаєте з уроків математики може бути цілим числом та навіть дробом.

a) Якщо степінь натуральне число або число 0, то результат буде ціле число:

```
>>> a=int()
>>> b=int() #але b повинно бути натуральним або число 0
>>> c=a**b
>>> type(c)
<class 'int'>
```

b) Якщо степінь число від`ємне, то результат буде дійсним числом:

```
>>> a=int()
>>> b=int()    #але b повинно бути від`ємним
>>> c=a**b
>>> type(c)
<class 'float'>
```

6) Ділення націло

При діленні націло на число, буде відбуватися ділення але дробова частина буде "відкидатися". Саме при цьому виді ділення результат також буде ціле число, отже якщо вам необхідно отримати ціле число при діленні вам потрібно використовувати цей вид ділення. Позначається знаком "//". Ось конструкція:

```
>>> a=int()
>>> b=int()  #b не дорівнює 0
>>> c=a//b
>>> type(c)
<class 'int'>
```

7) Залишок від ділення

Позначається дія знаком "%". Результатом буде ціле число:

```
>>> a=int()
>>> b=int()  #b не дорівнює 0
>>> c=a%b
>>> type(c)
<class 'int'>
```

Що стосується дійсних чисел, то для них будуть притаманні всі попередні дії. Якщо при дії з декількома числами *всі або хоча б одне з них буде дійсним,* то результат від дії в будь-якому випадку буде *дійсним числом*.

При виконання дій з числами існують ***пріоритети операцій***. Так як і в математиці можна використовувати дужки і тому спочатку виконуються дії в дужках, потім піднесення до степені, далі множення і ділення і тільки після цього додавання та віднімання. Тому 5*(4-2)=10, а 5*4-2=18.

Давайте розглянемо приклад задачі. Пропоную створити простий калькулятор для цілих чисел. На вході маємо два числа, програма виведе їх суму, різницю, добуток, частку першого на друге та другого на перше. Ускладнюється задача тим, що буде виводитися не тільки результат, а й сама дія з числами і після знака дорівнює результат.

```
a=int(input("Введіть a: "))
b=int(input("Введіть b: "))
print(a,'+',b,'=',a+b) # виведення суми цих двох чисел
print(a,'-',b,'=',a-b) # виведення різниці цих двох чисел
print(a,'*',b,'=',a*b) #виведення добутку цих двох чисел
print(a,'/',b,'=',a/b)
# виведення частки першого від другого числа
print(b,'/',a,'=',b/a)
# виведення частки другого від першого числа
```

Це лише найпростіші дії з числами в Python, бо також існують наступні вбудовані функції для роботи із ними (замість x та y встановлюються числа або змінні, з якими потрібно виконати ці дії):

abs(x) - модуль від числа;

bin(x) - переведення числа у двійкову систему числення;

hex(x) - переведення числа у шістнадцяткову систему числення;

max(x,y) - пошук максимуму з 2 чисел, може приймати будь-яку кількість аргументів;
min(x,y) - пошук мінімуму з 2 чисел, також може приймати будь-яку кількість аргументів;
round(x) - округлення числа;
round(x,y) - округлення числа x із вказаною точністю - у знаків після коми.

4.2 Дії з рядком тексту

На відміну від чисел з текстом можна виконувати тільки 2 основні операції: це додавання тексту та множення на число, але в будь якому разі результатом може бути тільки рядок тексту. А тепер про кожну дію окремо:

1) Додавання тексту

Суть дії полягає у тому, що ми можемо утворити рядок, який буде складатися із інших рядків тексту (доданків). Для того, щоб додати рядки тексту між собою, потрібно використовувати знак "+". При цьому кожен із доданків повинен бути лише текстом, а, отже, у результаті отримуємо рядок тексту. А для того, щоб до тексту додати значення іншого типу даних, потрібно користуватися функцією *str()*. Отже:

```
>>> a=str()
>>> b=str()
>>> type(a+b)
<class 'str'>
```

Для того, щоб краще зрозуміти як саме рядки можуть додаватися приклад:

```
>>> a='q'
>>> b='w'
>>> print(a+b)
qw
```

2) Дублювання тексту

Також текст можна помножити на певне число, тобто записати його вказану кількість раз (число може бути лише цілим), так як записувати програма буде вказаний текст, то і в результаті все одно отримаємо текст:

```
>>> a=str()
```

```
>>> b=int()
>>> type(a*b)
<class 'str'>
```

Наприклад:

```
>>> a='q'
>>> b=3
>>> print(a*b)
qqq
```

Для дій з текстом також зберігаються приорітети операцій та використовуються дужки, тобто ('q'+'w')*3 не дорівнює виразу 'q'+'w'*3. Спочатку, як і завжди, виконується дія у дужках, потім множення і тільки після всього цього черга доходить до додавання.

Контрольні запитання

1. Які основні дії можна виконувати з цілими та дійсними числами в Python?
2. Які основні дії можна виконувати з рядком тексту в Python?
3. Проаналізуйте зміну типів даних при виконанні дій з: a) цілими числами; b) дійсними числами; c) рядком тексту.

Практичні завдання

Створіть новий файл Python за допомогою середовища розробки програмного забезпечення IDLE. Створіть програму за даною задачею:

a. на вході маємо число a, яке буде запитуватися у користувача. Задачею програми буде порахувати значення виразу $\frac{(a+2)^3}{4} - 5$. Додати повідомлення за бажанням;
b. написати калькулятор за прикладом у пункті, але для дійсних чисел;
c. на вході маємо 3 числа, програма повинна вивести максимальне та мінімальне з цих трьох чисел;
d. на вході число a. Написати програму розв'язання рівняння 1) $7x - a = 0$; 2) $\frac{45}{ax} = 3$; 3) $4x + a \cdot (x - 1) = 4$. Результатом роботи програми буде знайти x та вивести його значення після тексту "x = ";

e. маємо 2 числа: 1) 3,17 і 20,56; 2) 56,5 і 2; 3) 147,65 і 34+45,13. Задачею програми буде порахувати скільки цілих чисел розташовано між числами;
f. написати програму, яка б переводила введене число у 1) двійкову систему числення; 2) шістнадцяткову систему числення.
g. на вході маємо текст і число (ціле). Задачею програми буде вивести добуток тексту на число. Результат представити в круглих дужках;
h. на вході маємо текст (позначити змінну як *txt*) і число (ціле, змінну позначити як a). Задачею програми буде виконати наступні дії: 1) '|'+(a*txt)+'|'; 2) a*'!'+txt+(a-1)*'!'; 3) (a*$\frac{2}{3}$+4-5)*txt. Результат вивести у квадратних дужках.

Лекція 5. Підключення додаткових модулів. Модуль math

Нам вже відомі прості дії з числами та текстом у Python. Але, так як Python — це високорівнева мова програмування, вона має багато додаткових функцій для роботи з числами та текстом, та щоб не навантажувати написану нами програму існують спеціальні *набори різноманітних функцій, які називаються* ***модулями***. Додаткові модулі потрібно підключати окремо. Це зроблено, щоб забезпечити швидкість виконання програми.

5.1 Підключення додаткових модулів

Отже для того, щоб підключити додатковий модуль до нашої програми потрібно застосувати таку конструкцію:

import *назва_модуля*

В Python існує велика кількість додаткових модулів для роботи з даними, і якщо у подальшому вашому житті ви плануєте працювати у сфері розробки програмного забезпечення і саме із Python, то рекомендовано знати та уміти працювати із основними з них. Наприклад: існує такий модуль *unittest*, який створений для тестування своїх програм на Python, а ми будемо працювати з модулем *math*, який представляє собою пакет з додатковими функціями для роботи з числами.

А функції підключеного модулю вводяться в код наступним чином:

*назва_модуля***.***назва_функції***()**

5.2 Модуль math

Як ви вже дізналися, модуль *math* використовується для роботи з числами, та виконання з ними додаткових дій.

Отже, для того щоб підключити даний модуль потрібно лише на початку програмного коду встановити наступну конструкцію коду:

```
import math
```

Зверніть увагу! Підключати модулі потрібно саме на початку, бо, якщо ви підключите його вже після виконання функції, яка входить в даний модуль, то вона не буде зрозуміла інтерпретатору і ваша програма не буде виконана.

Що стосується можливостей даного модуля, то в нього входить велика кількість функцій для роботи із числами. **Наприклад**: *math.sqrt(x),*

яка знаходить квадратний корінь від числа *x*, або *math.fabs(x),* яка знаходить модуль від числа *x*, та багато інших. Про інші функції ви можете дізнатися із додатку.

Розглянемо приклад задачі. На вході є число, задачею програми буде вивести корінь квадратний з даного числа. У рішенні даної задачі нам допоможе модуль *math* та функція **math.sqrt(x)** цього модулю.

```
import math
# підключення модулю math
a=float(input("Введіть число a: "))
print(math.sqrt(a))
# виведення квадратного кореня з цього числа
```

Контрольні запитання

1. Для чого потрібні додаткові модулі у програмуванні?
2. Як підключити додаткові модулі до своєї програми?
3. З яким модулем ми познайомилися у даному пункті?
4. Наведіть приклади функцій із модуля math.

Практичні завдання

Створіть новий файл Python за допомогою середовища розробки програмного забезпечення IDLE. Створіть програму за даною задачею:

a. на вході маємо 2 числа (наприклад *a* і *b*). Програма повинна виконати дії: 1) a^b; 2) b^a; 3) a^{b-1}; 4) b^{a-1}; 5) $(a*b)^{a+b}$, за допомогою функцій модуля math. Результат вивести на екран;

b. на вході маємо число, яке є радіусом певного кола. Знайти довжину кола та площу круга із радіусом у 2 рази більшим. **Підказка:** використовувати функцію math.pi;

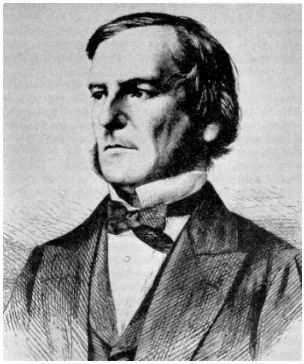
c. на вході маємо ціле число, яке запитується у користувача. На виході факторіал введеного числа.

Лекція 6. Логічний тип даних. Дії з даними логічного типа даних

6.1 Логічний тип даних

Думаю, що ви не раз зустрічалися з поняттям логіки, зокрема математичної. З ресурсів Вікіпедії маємо визначення логіки. Логіка — наука про закони і різновиди мислення, умови істинності знань і суджень. А тепер спробуємо сформувати своє визначення, більш простіше, яке ми будемо використовувати. ***Логіка*** *– наука про умови.* Без володіння логікою не можливо вирішити жодну задачу з математики, та й у повсякденному житті ми не раз користувалися різними видами логіки. Для програмістів математична логіка це така собі "зброя", без якої жоден програміст не зміг би створити жодної програми.

Ще з молодших класів ви навчилися порівнювати між собою числа та вирази, а також перевіряти певні умови, звичайно ж за допомогою математичної логіки. **Наприклад:** твердження 2>1 є вірним, або **істинним**, а твердження 1>2 не є вірним, а отже **хибне**.



Мал. 7

У всіх мовах програмування існує окремий тип даних для логічних виразів — ***логічний тип даних,*** або, як його ще називають, ***булевий.*** Назва пішла від імені відомого англійського математика, основоположника математичної логіки Джорджа Буля (мал. 7).

Всі логічні вирази (порівняння даних) належать до булевого типу даних. Для порівняння даних застосовуються наступні знаки (табл. 3).

Назва	Позначення
Рівність	==
Більше (Менше)	> (<)
Більше або дорівнює (Менше або дорівнює)	>= (<=)
Не дорівнює	!=

Табл. 3

Булевий тип даних вважають дуже примітивним, адже він має всього 2 значення: ***True*** (істина), або ***False*** (хиба). Залежно від того, чи

виконується умова вираз може набувати або значення *True,* або *False*. Тобто:

```
>>> a=5
>>> a>4
True
```

Якщо вам потрібно представити *True* або *False* за допомогою цифр, то *True* позначають 1, а *False* позначають 0, так як кожне із значень займає всього 1 біт, бо як відомо біт це 1 або 0.

6.2 Дії з даними логічного типа даних

Як і ті типи даних, які ми вивчили раніше, булевий тип даних, можна перевести в цілі та дійсні числа або рядки тексту. Для цього потрібно застосувати вже вам відому функції *int(), float(), str()*. Та зважаючи на те, що ми маємо тільки 2 значення, результат очевидний:

```
>>> int(True)
1
>>> int(False)
0
>>> float(True)
1.0
>>> float(False)
0.0
>>> str(True)
'True'
>>> str(False)
'False'
```

З того, що значення булевого типу даних можуть приймати значення цілих чисел, ми можемо додавати, віднімати, ділити та множити логічні вирази на числа, але при цьому потрібно користуватися функціями переводу. **Наприклад:** int(2>1)+3*(2+int(1>2)) в Python буде дорівнювати 7, так як int(2>1)=1, а int(1>2)=0, тобто 1+3*(2+0)=7.

Розглянемо приклад задачі. На вході маємо число *a*, яке запитується у користувача, перевіряється умова a>0, результат перевірки виводиться на екран з приставкою "It`s" в початку, тобто, якщо *a* буде дорівнювати 4 то на екран виведеться "It`s True" і т.п.

```
a=float(input("Введіть a: "))
```

```
print("It`s"+str(a>0)) # виведення результату
```

Контрольні запитання

1. Дайте визначення поняттю логіка.
2. Що відноситься до даних логічного типу?
3. Скільки значень може набувати будь-який логічний вираз?
4. Що це за значення?
5. Коротко розкажіть про дії з даними логічного типу.

Практичні завдання

1. Створіть новий файл Python за допомогою середовища розробки програмного забезпечення IDLE. Створіть програму за даною задачею:

   a. На вході маємо число *a*, програма перевіряє чи є число a: 1) більше 0; 2) менше 0 та виводить результат у вигляді True або False;

   b. на вході маємо 2 числа програма порівнює їх між собою і виводить результат у вигляді True або False. Порівняння відбувається за такими пунктами:

      1) Чи ці числа рівні;
      2) чи числа нерівні;
      3) чи число більше за друге;
      4) чи друге число більше за перше;
      5) чи квадрат першого числа більший за квадрат другого числа;
      6) чи сума цих чисел більша за їх добуток.

2.* Побудувати програму "Щасливий квиток". Припустимо користувач придбав квиток на автобус з чотирьохзначним номером. Квиток буде вважатися щасливим, якщо сума перших двох цифр буде дорівнювати сумі двох інших цифр цього квитка. На вході матимемо два числа, перше буде складатися із перших 2 цифр номеру білету, а наступне число із двох інших цифр. На виході маємо повідомлення у вигляді True/False.

Практична робота №1

Тема: Складання та виконання лінійних алгоритмів для опрацювання величин.

Для виконання даної практичної роботи необхідно знати:

- функції введення та виведення даних;
- дії з цілими та дійсними числами;
- функції додаткового модуля *math*;
- дії з даними логічного типу даних.

Завдання 1

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. Розробіть програму, для розв'язання рівняння $\frac{x^2-5}{a} = b$. Користувач вводить параметри *a* (*a* не дорівнює 0) і *b*, на виході маємо значення *x*, яке виводиться з відповідним повідомленням "x=".

Завдання 2

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. Задача програми проаналізувати чи введене тризначне ціле число *a* в зворотному записі не змінить своє значення. Відповідь отримаємо у вигляді *True/False*.

Лекція 7. Логічні оператори and, or, not. Розгалуження в Python

Ми дізналися про новий тип даних у мові програмування Python – булевий. І знаємо, що значення цього типу набувають логічні вирази (порівняння даних), а саме *True* або *False*. А тепер навчимося виконувати дії із логічними виразами.

Оператор (у програмуванні) *– це знак або вираз, який позначає виконання певної дії.* Ми вже знайомі із деякими математичними операторами, але не називали їх так. Тобто оператор «+» означав дію додавання, а оператор «/» - ділення. Тому і для дій з логічними виразами існують спеціальні оператори, про які зараз буде йти мова.

7.1 Оператор AND (з англ. «і»)

Оскільки ці оператори виконують дії з змінними логічного типу даних, а отже і результат буде *True* або *False*. Цей оператор засовується таким чином: [лог. вираз] *and* [лог. вираз] *and* ... Такий складений логічний вираз буде набувати значення *True* тільки у тому випадку, якщо всі зміні або логічні вирази будуть мати значення *True*. Якщо хоча б один із них буде мати значення *False* то і весь вираз буде набувати цього значення.

Його застосовують для запису декількох умов, які повинні виконатися одночасно. Так як в програмуванні мовою Python не рекомендовано застосувати вираз 1>a>5, нам потрібно об'єднати два вирази: 1>a та a>5. От саме для цього і потрібен оператор *and.* Тобто 1>a and a>5.

Давайте розглянемо таблицю 4. Перший стовпчик - це значення логічної змінної *a*, другий - значення змінної *b*, а третій це результат виразу *a and b*. Зверніть увагу, на результат дії та його зміну.

a	b	a and b
True	True	True
True	False	False
False	True	False
False	False	False

Табл. 4

Тепер попрацюємо з оператором *and* в самому середовищі розробки програмного забезпечення IDLE. Отже, задача наступна: на вході маємо число, яке буде запитуватися у користувача з повідомленням «a=». На виході маємо результат перевірки на те чи належить це число проміжку від 0 до 100 (тобто 0<a<100).

```
a=float(input("a="))
print(a>0 and a<100)
# виведення результату перевірки чи належить число a
# проміжку від 0 до 100
```

7.2 Оператор OR (з англ. «або»)

Щодо оператора *or*, то він застосовується так само як і оператор *and*. Результатом виразу буде *True* в тому випадку, якщо хоча б один із складових цього виразу буде мати значення *True* (табл. 5).

a	b	a or b
True	True	True
True	False	True
False	True	True
False	False	False

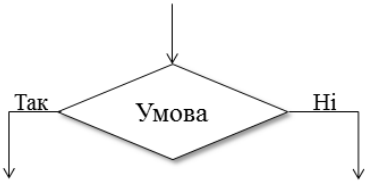
Табл. 5

7.3 Оператор NOT (з англ. «не»)

Цей оператор застосовується трішки по іншому: він ставиться на початку, тобто перед логічним виразом, надаючи йому при цьому протилежного значення. Тобто, вираз *not 2>1* буде приймати значення *False* (протилежний до значення виразу 2>1). Взагалі не дуже часто використовують цей оператор окремо, так як вираз *not 2>1* можна замінити на протилежний логічний вираз (в даному випадку 2<1). Та все ж його часто застосовують із іншими операторами. Для прикладу вираз *not 1>2 and 4>3* буде мати значення *True* (not 1>2 = True, 4>3 = True).

Взагалі, всі логічні оператори можна об'єднувати у різні вирази. Наприклад: not 2>1 or (3>2 and 4>=4).

7.4 Розгалуження на мові програмування Python



Мал. 8

Ну так як ми заговорили про логіку, то, звичайно, ми повинні згадати і про розгалуження. Знову повернемося до блок-схем. У графічному вигляді розгалуження позначається таким чином (мал. 9). Тобто є умова і в залежності від її виконання, чи не виконання будуть запускатися певні дії.

Для того, щоб представити умову у програмному коді Python, використовуються наступні оператори:

- ***if*** (з англ. «якщо»), застосовується у вигляді наступної конструкцій:

  if логічний вираз**:**

  дії, які будуть виконуватися якщо логічний вираз буде мати значення *True*

- ***else*** (з англ. «інакше»), є необов'язковою частиною конструкції if:

  if логічний вираз**:**

  дії, які будуть виконуватися якщо логічний вираз буде мати значення *True*

  else:

  дії, які будуть виконуватися якщо логічний вираз буде мати значення *False*

Конструкцію лише із оператором if називають **неповним розгалуженням**, а **повним** в свою чергу конструкцію з if та else.

Отже, основна конструкція розгалуження у мові Python складається із двох «частин» if та else. Давайте поглянемо, як це працює не практиці. Задачею буде написати програму на вході якої маємо число, програма перевіряє чи це число більше 0 і видає результат у випадку True: «Це число додатнє», а у випадку False: «Це число від'ємне».

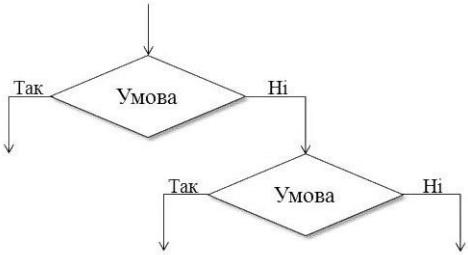
Маємо наступну конструкцію коду:

```
a=float(input("Введіть число a: "))
if a>0: # застосування оператора if з умовою a>0
    print("Це число додатнє")
# виведення повідомлення "Це число додатне", якщо вираз a>0
# дорівнює True
else: # застосування оператора else
    print("Це число від'ємне")
# виведення повідомлення "Це число від'ємне", якщо вираз
# a>0 дорівнює False
```

Запустимо програму і для прикладу спочатку введемо число -5 та отримаємо повідомлення "Це число від'ємне", а потім введемо число 6, та отримаємо повідомлення "Це число додатнє". (мал. 9)

```
================ RESTAR
Введіть число a: -5
Це число від'ємне
>>>
================ RESTAR
Введіть число a: 6
Це число додатнє
>>>
```

Мал. 9



Мал. 10

Але у нас часто можуть виникати складні розгалуження, які передбачають перевірку декількох умов (мал. 10). Для цього існує ще один оператор ***elif*****,** він також не використовується самостійно, а є необов'язковою складовою конструкції if. Застосовується він наступним чином:

if логічний вираз**:**

дії, які будуть виконуватися якщо логічний вираз буде мати значення *True*

elif логічний вираз**:**

дії, які будуть виконуватися якщо попередній логічний вираз має значення *False*, а даний логічний вираз буде має значення *True*

…

else:

дії, які будуть виконуватися якщо всі логічні вирази будуть мати значення *False*

Як ви вже могли помітити із конструкції коду, оператор elif можна використовувати нескінчену кількість раз.

Для того, щоб закріпити теоретичні знання практикою, пропоную вдосконалити нашу попередню задачу. Давайте розглянемо варіант, що число *a* буде дорівнювати 0, адже наша програма буде видавати неправильний результат при введенні нуля, просто так як в цьому випадку умова не буде виконуватися тому буде виводитися, що число 0 є від'ємним (мал. 11). Отже додамо ще одну перевірку, чи є введене число нулем, якщо так, то на екран буде виведено повідомлення «Число дорівнює 0». Після вдосконалення програма має такий вигляд:

```
========== RES
Введіть число a: 0
Це число від'ємне
>>>
```

Мал. 11

```
a=float(input("Введіть число a: "))
if a>0: # застосування оператори if з умовою a>0
```

```
      print("Це число додатнє")
      # виведення повідомлення "Це число додатне", якщо a>0
      # дорівнює True
elif a==0: # застосування оператори elif з умовою a==0
      print("Число дорівнює 0")
      # виведення повідомлення "Число дорівнює 0", якщо a==0
else:
      print("Це число від'ємне")
      # виведення повідомлення "Це число від'ємне", у іншому
      # випадку
```

Запустимо її та введемо числа -1, 1 та 0 і побачимо, що ми отримаємо (мал. 12).

```
Введіть число a: -1
Це число від'ємне
>>>
========== RESTA
Введіть число a: 1
Це число додатнє
>>>
========== RESTA
Введіть число a: 0
Число дорівнює 0
>>>
```

Мал. 12

Зверніть увагу! Всі дії які є пунктами оператора *if*, *else* та *elif* у коді починаються з нового рядка з відступом від початку рядка, тим самим ви позначаєте належність цих дій відповідному оператору.

Контрольні питання

1. Для чого використовуються логічні оператори and, or та not?
2. Як будується розгалуження у мові програмування Python?
3. Для чого служить оператор elif?

Практичні завдання

Створіть новий файл Python за допомогою середовища розробки програмного забезпечення IDLE. Створіть програму за даною задачею:

a. на вході маємо ціле число. Якщо число входить у проміжок (0;100) вивести повідомлення «Число входить в проміжок», якщо ні то вивести відповідне повідомлення;
b. на вході маємо ціле число. Якщо число входить у проміжок [-33;150)∪{151} вивести повідомлення «Число входить в проміжок», якщо ні то вивести відповідне повідомлення;
c. на вході маємо будь-яке число. Якщо число входить у проміжок (-100;0)∪(0;100) вивести повідомлення «Число входить в проміжок», якщо ні то вивести відповідне повідомлення;

d. на вході маємо три числа. Програма порівнює їх між собою та виводить результат на екран. **Наприклад:** якщо користувач введе числа 4, 7, 2, то програма виведе повідомлення «2<4<7». Розглянути всі варіанти;

e. на вході маємо довжини сторін трикутника. Якщо даний трикутник може існувати, то на екран виведеться повідомлення «Даний трикутник існує», в іншому випадку «Трикутника із такими сторонами не існує». **Підказка:** трикутник буде існувати якщо сума будь-яких двох його сторін буде більша за третю сторону;

f. маємо рівняння виду $\frac{a}{x-3} = b$. Параметри a і b запитуються у користувача. Задачею програми буде розв'язати дане рівняння та вивести чому дорівнює x;

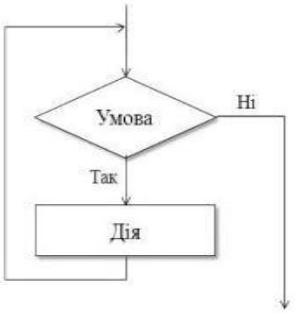
g. маємо рівняння виду $\frac{a+b}{(x-3)(x+2)} = c$. Параметри a, b і c запитуються у користувача. Задачею програми буде розв'язати дане рівняння та вивести чому дорівнює x.

Лекція 8. Цикли мовою Python. Умовні та з лічильником

Ми вже вміємо будувати програми з лінійною структурою та ознайомилися із розгалуженнями. Тепер розглянемо як будуються цикли мовою Python. ***Цикл*** *– це процес багаторазового повторення дії або послідовності дій*. У програмах цикли поділяються на ***умовні*** та з ***лічильником***.

8.1 Умовні цикли

В умовних циклах повторення буде відбуватися доти, доки буде виконуватися певна умова. Якщо зобразити умовний цикл графічно то ми отримаємо ось таку частину блок-схеми (мал. 13).



Мал. 13

У мові програмування Python умовний цикл застосовується за допомогою оператора ***while***. У коді це буде виглядати так:

while *лог. вираз***:**

тіло циклу

Тілом циклу *називають набір команд, які будуть повторюватися, доки працює цикл.*

Для прикладу розглянемо наступну задачу. Умова задачі наступна – на вході маємо ціле число, програма буде ділити його на 2 та виводити результат доки воно не перестане ділитися на 2. Коли число перестане бути парним - програма виведе повідомлення "Число не є парним".

```
a=int(input("Введіть ціле число a: "))
while a%2==0: # застосування циклу while
      a=a/2
      # зміна значення змінної a
      print(a)
      # виведення нового значення a
print("Число не є парним")
```

```
================== RESTAR
Введіть ціле число a: 64
32.0
16.0
8.0
4.0
2.0
1.0
Число не є парним
>>>
================== RESTAR
Введіть ціле число a: 7
Число не є парним
>>>
```

Мал. 14

Запустимо програму і для прикладу введемо 64 та число 7 подивимося на результат (мал. 14). Зверніть увагу на те, що число 7 не є парним, отже цикл навіть не буде розпочато, а одразу виведеться відповідне

повідомлення.

8.2 Цикли з лічильником

Цикли з лічильником відрізняються від умовних тим, що в них дії повторюються чітку кількість разів.

Конструкція в коді буде виглядати так (в загальному вигляді):

for **i** in range(**n**)**:**

тіло циклу

Зміст цього циклу звучить так: повторювати від *i* до *n* раз.

i - початкове значення, за замовчуванням дорівнює 0;

n - кінцеве значення.

Кількість повторів може бути як конкретним числом, так і запитуватися у користувача.

Зверніть увагу! Число *n* не може бути дійсним, так як позначає кількість повторів, а кількість повторів повинна бути тільки натуральним числом.

Для прикладу розглянемо просту задачу. На вході маємо 2 цілих числа (*a* і *b*). Задачею програми буде до числа *a* додавати одиницю *b* разів, а потім вивести кінцевий результат.

```
a=int(input("Введіть ціле число a: "))
b=int(input("Введіть кількість повторів: "))
for i in range(b):  # застосування циклу for
    a=a+1
    # зміна значення змінної a
print(a)
# виведення на екран кінцеве значення змінної a
```

Для прикладу введемо числа 3 і 5. Результатом роботи програми буде число 8 (3+1+1+1+1+1=8) (мал. 15).

```
========== RESTART: /n
Введіть ціле число a: 3
Введіть кількість повторів: 5
8
>>>
```

Мал. 15

Тепер давайте розглянемо більш складний приклад. На вході маємо ціле число *a*, задачею програми буде порахувати суму 5 чисел перше з яких *a*, а кожне наступне на 2 більше від попереднього. В даному випадку потрібно створити додаткову змінну для суми, яка на початку буде дорівнювати 0, а потім буде збільшуватись на число *a*.

```
a=int(input("Введіть ціле число a: "))
s=0
# початкове значення суми
for i in range(5):  # застосування циклу for
      s=s+a   # збільшення значення суми на a
      a=a+1   # зміна значення змінної a
print(s)
# виведення суми
```

Ну і запустимо програму та введемо 5, а на виході отримаємо значення суми 35 так, як 5+6+7+8+9=35 (мал. 16).

```
========== RESTAR
Введіть ціле число a: 5
35
>>>
```

Мал. 16

Контрольні питання

1. Дайте визначення поняттю цикл.
2. Що таке тіло циклу?
3. Умовний цикл у Python представлений оператором…
4. Яка конструкція застосовується для запуску циклу з лічильником? Що означають змінні 'i' та 'n'?

Практичні завдання

Створіть новий файл Python за допомогою середовища розробки програмного забезпечення IDLE. Розробіть запропоновані програми.

a. На вході маємо будь-яке число, яке запитується у користувача з повідомленням. Доки це число не перевершуватиме 100 до нього буде додаватися число 5 та виводитися кожне нове значення цього числа.

b. Створіть просту комп'ютерну гру в яку грають 2 учасники. Перший вводить будь-яке число від 0 до 10, а наступний має ввести задумане число, при цьому другий гравець не повинен бачити задумане число. Доки гравець не вгадає число програма не буде завершена.

c. Користувач банку поклав на депозит в банк 1000 грн. на n-ну кількість років (n запитується у користувача). Відомо, що річний відсоток становить 25%. Задачею програми буде порахувати скільки грошей отримає користувач на при кінці.

d. Програма має порахувати суму *n* чисел, перше з яких *a*, а кожне

наступне на 5 більше від попереднього. (*n* і *a* – запитуються у користувача).

е. Створіть програму, яка буде виводити факторіал введеного числа. Використовувати готові функції та модулі заборонено.

Практична робота №2

Тема: Складання та виконання алгоритмів з розгалуженнями та повтореннями для опрацювання величин.

Для виконання даної практичної роботи необхідно знати:

- основні логічні оператори для побудови складних виразів;
- як представити розгалуження засобами мови Python;
- яке призначення операторів *while* та *for*.

Завдання 1

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. На вході маємо день тижня (у нижньому регістрі українською мовою), програма повинна вивести скільки уроків стоїть у вас в розкладі на цей день. При введенні суботи або неділі виводитиметься повідомлення "На цей день уроків не заплановано".

Завдання 2

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. Маємо нескінчену послідовність із цілих чисел, перше з яких 0, а кожне наступне на d більше. Програма повинна вивести суму перших n чисел в цій послідовності, включаючи і число 0. Значення параметрів d і n запитується у користувача на початку.

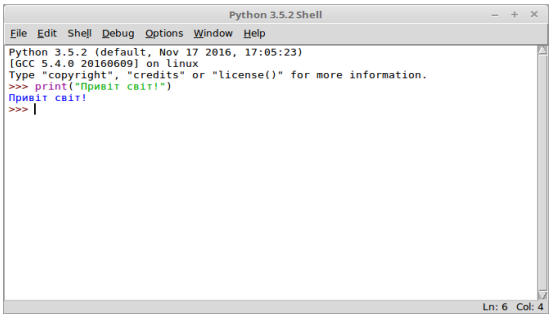
Лекція 9. Інтерфейс користувача на мові Python. Створення вікон та налаштовування їх властивостей

9.1 Інтерфейс користувача у мові Python

Кожна програма, має свій інтерфейс. Ви вже знаєте, що ***інтерфейс*** – *це сукупність засобів і правил, що забезпечують взаємодію пристроїв обчислювальної системи або програм.* Існує 2 види інтерфейсу програм: **командний** та **графічний**. Що стосується командного інтерфейсу, то взаємодія користувача та комп'ютера відбувається за допомогою команд, які користувач вводе в командну строку з клавіатури. А от графічний інтерфейс є більш зручнішим для користувача, адже взаємодія відбувається за допомогою кнопок, полів, та інших елементів управління.

Ми з вами також навчимося створювати програми, які будуть мати свій власний графічний інтерфейс мовою програмування Python. І почнемо із поняття форми. ***Форма*** *- об'єкт, в якому можна розмістити різні компоненти (елементи керування), зокрема кнопки, поля, написи, меню та інше. Зазвичай форми представляють собою звичайні програмні вікна*, в яких будуть відображатися вищезазначені елементи. Тож і ми спробуємо створити вікно програми мовою Python з різноманітними властивостями.

Мова Python більше орієнтована на командний інтерфейс, але в ній існує спеціальний модуль за допомогою якого можна створити інтерфейс користувача. Модулем у програмуванні називають певний пакет додаткових функцій для роботи над розробкою програм. Сам IDLE не зможе створити окреме вікно, максимум, що ми можемо отримати – це рядки з текстом, безпосередньо у вікні самого IDLE (мал. 17).



Мал. 17

Не будемо гаяти часу і створимо новий файл та почнемо писати нашу першу програму. Отже, як було сказано раніше, IDLE не може створювати окреме вікно програми самостійно і для цього ми повинні підключити додатковий модуль. Для підключення будь-яких модулів

використовується конструкція:

import *назва_модуля*

Також можна використовувати наступну конструкцію, для того, щоб потім повторно не вказувати модуль при використанні функцій:

from *назва_модуля* **import ***

В цьому випадку ми вказуємо, що з певного модуля ми імпортуємо всі функції (*).

Модуль, який нам знадобиться для створення графічного інтерфейсу, програми називається **tkinter**. За допомогою цього модуля можна виконувати як окремі графічні побудови, так і створити повноцінний графічний інтерфейс користувача. Отже, перший рядок коду в нашому новому файлі буде завжди виглядати так:

from tkinter **import ***

9.2 Створення вікон та налаштовування їх властивостей

Після підключення модулів, створимо вікно. Для створення вікна використовується функція **Tk()**, але, так як таких вікон можна створити нескінченно багато, то кожному вікну присвоюється певне ім'я:

назва_вікна=**Tk**()

Назва вікна може складатися з великих і малих літер англійського алфавіту, цифр та знаку нижнього пробілу.

Наприклад, створимо вікно *Window1*:

```
Window1=Tk()
```

Назва вікна потрібна для того, щоб потім при написанні коду ми змогли звертатися до даного вікна, присвоювати йому певні властивості та вставляти у нього елементи управління. І ще дуже важлива річ: після створення вікна та його елементів управління потрібно вказати інтерпретатору, що ми закінчили працювати з вікном за допомогою метода ***mainloop()***. Застосовується він так:

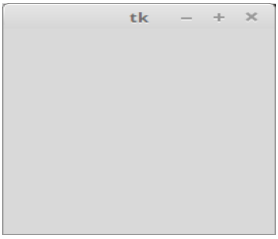
*назва_вікна***.mainloop**()

У випадку із нашим вікном цей рядок виглядатиме так:

```
Window1.mainloop()
```

Давайте ці всі команди та рядки коду об'єднаємо у один файл, який ми створили та подивимося на те, що ми отримали:

```
from tkinter import *
# підключення модулю
Window1=Tk()
# створення вікна
Window1.mainloop()
# закінчення роботи з вікном
```



Мал. 18

Спробуємо запустити нашу програму. Як ми бачимо у нас створилося пусте вікно (мал. 18).

Раніше вже було сказано, що ми можемо створювати велику кількість вікон, але в залежності від того, як ми розмістимо рядки коду, ці вікна будуть по різному відображатися. Наприклад, якщо функцію створення та метод закриття одного вікна ми розмістимо послідовно з функціями та методами іншого, то друге вікно буде створене тільки після того, як користувач закриє перше:

```
Window1=Tk()
Window1.mainloop()
Window2=Tk()
Window2.mainloop()
```

Для того, щоб краще зрозуміти написане, рекомендую повторити дії та запустити команду і побачити на практиці, як це працює.

А для того, щоб створити ці вікна паралельно, ми повинні послідовно розмістити відповідні команди між собою:

```
Window1=Tk()
Window2=Tk()
Window1.mainloop()
Window2.mainloop()
```

Тепер ви знаєте як створити вікна за допомогою модуля tkinter. А зараз попрацюємо з налаштовуванням їх властивостей. До основних властивостей вікна належать:

- **title(**"Заголовок"**)** – заголовок вікна. За замовчуванням заголовок вікна «tk»;
- **minsize(**x,y**)** – мінімальний розмір вікна у пікселях, x- ширина, y – висота. Якщо не встановити цю властивість, то вікно не матиме обмежень у зменшенні;

- **maxsize**(x,y) – максимальний розмір вікна у пікселях, , x- ширина, y – висота. Якщо не встановити цю властивість, вікно може приймати повні розміри екрану користувача;
- **geometry("**400**x**200**+**450**+**150**")** – розміри та положення вікна відносно розширення екрана у пікселях. 400 - ширина, 200-висота, 450-відстань від лівого краю екрану, 150 – відстань від верхнього краю екрану. Параметри відступу (відстань від лівого краю екрану та відстань від верхнього краю екрану) не є обов'язковими, вони додаються за бажанням. **Зверніть увагу!** Всі числа були введені як приклад;
- **resizable**(x,y) – чи може користувач змінювати розміри вікна і на скільки: x – ширина, y – висота. Для заборони змінення розмірів вікна замість параметрів встановити нулі.

Це основні властивості, які ви будете використовувати у навчанні для створення простих проектів за допомогою мови програмування Python. До вікна вони застосовуються таким чином:

*назва_вікна***.***властивість*(*параметри*)

Наприклад, встановимо розміри нашого вікна Window1: ширина та висота по 500 пікселів:

Window1.geometry(500x500)

Аналогічно застосовуються й інші властивості, та є й винятки із правил. Наприклад, для того, щоб застосувати певний колір фону нашого вікна виконується інша конструкція:

*назва_вікна***["bg"]="**колір**"**

Bg скорочено від background (фон), а параметром є назва кольору на англійській мові взята в лапки.

Наприклад, встановимо зелений колір для нашого вікна:

Window1["bg"]="green"

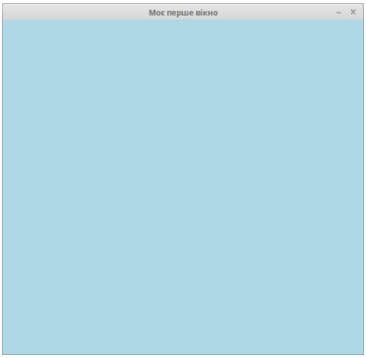
На самостійне опрацювання! Знайти якомога більше кольорів, які можливо встановити до вікна tkinter в мові програмування Python.

Ну ось і все, що вам потрібно знати на цьому етапі. Ну а тепер, давайте розглянемо приклад завдання. Отже, завданням буде: створити вікно світло-блакитного кольору зі сталими розмірами: ширина 600, висота

500 та з заголовком «Моє перше вікно».

Спочатку створюємо вікно, назвемо його як завгодно, наприклад *My_window*. Потім встановимо його розміри за допомогою властивості geometry, заборонимо змінювати його розміри та встановимо світло блакитний колір (light blue). Не забуваємо додати заголовок вікна та закриваючу команду mainloop. Ось що ми отримали:

```
from tkinter import *
# підключення модулю
My_window=Tk()
# створення вікна
My_window.title("Моє перше вікно")
My_window.geometry("600x500")
My_window.resizable(0,0)
My_window["bg"]="light blue"
# встановлення властивостей вікна
My_window.mainloop()
```



Мал. 19

Ну, і звичайно, спробуємо запустити нашу невелику програму та подивимося, що у нас вийшло (мал. 19)

Контрольні питання

1. Які існують два види інтерфейсу? Чим вони між собою відрізняються?
2. Що таке форма?
3. Що представляють собою форми в мові програмування Python?
4. Як називається модуль для створення графічного інтерфейсу користувача на мові Python? Як його підключити?
5. Як створити вікно на мові Python з використанням модуля tkinter?
6. Які властивості можна надати вікну? Як їх застосовувати?

Практичні завдання

1. Створіть новий файл Python. Підключіть відповідний модуль та створіть вікно зеленого кольору, з розмірами 500x600, та заголовком "Вікно №1".

2. Створіть новий файл Python. Підключіть відповідний модуль та створіть вікно жовтогарячого кольору, з розмірами 400x400, з заголовком "Це вікно! ", з мінімальними розмірами 100x100.

3. Створіть новий файл Python. Підключіть відповідний модуль та створіть вікно білого кольору, з розмірами 1000x500, з заголовком "Заголовок

вікна", та з забороною змінювати розміри вікна.

4. Створіть новий файл Python. Підключіть відповідний модуль та створіть вікно жовтого кольору, з розмірами 654x456, відступом від лівого краю 300 та відступом від верхнього краю 400. Заголовок "Вікно №4", та заборона змінювати розміри вікна.

5. Створіть новий файл Python. Підключіть відповідний модуль та створіть вікно блакитного кольору, з розмірами 700x300, відступом від лівого краю 100 та відступом від верхнього краю 0. Заголовок "Вікно №5", та заборона змінювати розміри вікна. Та створити вікно із заголовком "Вікно 5" синього кольору, з ідентичними розмірами та забороною змінювати розмір вікна. Вікна розташувати послідовно, тобто 2-ге вікно відкриється лише після закриття першого.

Лекція 10. Події та обробники подій. Вікно повідомлення

10.1 Події та обробка подій

Ми вже навчилися створювати статичні вікна та налаштовувати їх властивості. Але програми не будуються лише на статиці, їм властива і динаміка, тобто певні зміни. Тож розглянемо таке поняття як події.

Подія *— це зміна властивостей об'єкта, взаємодія між ними, утворення нового або знищення існуючого.* Кожна подія містить оцінку часу, що вказує, коли вона відбувається і місця, де вона відбувається.

В роботі із вікнами на мові Python існують такі події:

- **Button-1** — клік лівою клавішею миші по будь-якій області об'єкта;
- **Button-3** — клік правою клавішею миші по будь-якій області об'єкта;
- **KeyPress** — натискання будь-якої клавіші на клавіатурі;
- **Motion** — переміщення курсора миші по області об'єкта;
- **Destroy** — закриття вибраного вікна.

Звичайно, це не весь перелік, але й цього невеликого списку нам буде досить. Весь перелік ви зможете знайти за наступним посиланням: https://ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python (Перейдіть до пункуту «Привязка событий»)

З будь-якою подією, яка може відбутися з формою, можна пов'язати фрагмент програми, який буде виконуватися одразу після настання цієї події. Такий фрагмент програми називають **обробником події**.

У Python обробником подій є ***функція****, що являє собою набір команд.* Тобто спочатку потрібно створити функцію, щоб до неї можна було звертатися при виконанні певної події. Функцію потрібно оголошувати на початку, після підключення модулів, аби не вийшло непорозумінь з інтерпретатором.

Отже функції створюються наступним чином:

def *назва_функції***(***аргументи***):**

команда

...

Назву функції ми вигадуємо самостійно, а до команд можуть відноситися створення об'єктів, елементів управління, або змінення властивостей певних об'єктів. Саме зі зміною властивостей вікна ми і будемо працювати.

Отже давайте, наприклад, створимо функцію, яка б змінювала розміри нашого вікна Window на 500x500 пікселів та встановлювала зелений колір фону. Назвемо нашу функцію *change*, на місце аргументу встановлюємо *event* (з англійської - подія), це означатиме, що функція пов'язана з певною подією:

```
from tkinter import *
def change(event):
       Window.geometry("500x500")
       Window["bg"]="green"
...
```

Тепер створимо подію та присвоїмо їй дану функцію. Для цього ми користуємося наступною конструкцією:

назва_об'єкта_до_якого_буде_застосовуватися_подія.**bind**(**"<***подія***>"**, *назва_функції*)

Зверніть увагу на те, як потрібно записувати подію. Назву події потрібно вносити в відповідну конструкцію "<>", про це не треба забувати. Наприклад: "<Button-1>", "<Button-3>" і т.д.

Отже, створимо подію натисканням лівою клавішею миші по області вікна Window, при якій буде виконуватися функція *change*.

Тобто маємо:

Window.bind("<Button-1>", change)

Об'єднаємо ці конструкції в одну програму. Створимо новий файл в IDLE та внесемо в нього наступний код:

```
from tkinter import *
def change(event):
# створення функції change для змінення розмірів та кольору вікна
       Window.geometry("500x500")
       Window["bg"]="green"
Window=Tk()
Window.bind("<Button-1>", change)
```

```
Window.mainloop()
```

10.2 Вікно повідомлення

Окрім того, що ми можемо змінювати властивості об'єктів при виконанні певної події, ми можемо створювати вікна повідомлення, тобто командою функції буде створити вікно з відповідним повідомленням для користувача.

Вікна з повідомленням зручні тим, що вони допомагають донести інформацію користувачеві, при цьому не займаючи місця на головному вікні. Вікно повідомлення складається тільки із інформації та кнопки "Ок".

Функцією створення вікна повідомлення є ***messagebox.showinfo()*** і застосовується вона наступним чином:

messagebox.showinfo(**"**заголовок_вікна**"**, **"**зміст_повідомлення**"**)

Застосуємо для нашого вікна та встановимо заголовок "Повідомлення", та текст "Я програмую на Python". От, що у нас вийшло:

```
from tkinter import *
def change(event):
      Window.geometry("500x500")
      Window["bg"]="green"
      messagebox.showinfo("Повідомлення", "Я програмую на Python")
      # створення вікна повідомлення
Window=Tk()
Window.bind("<Button-1>", change)
Window.mainloop()
```

Давайте запустимо нашу програму та побачимо, що відбудеться при натисканні на вікно лівою клавішею миші (мал. 20).

Мал. 20

Контрольні питання

1. Дайте визначення поняттю подія.
2. Наведіть приклади подій в мові програмування Python.
3. Що таке обробник подій?
4. Що є обробником подій на мові програмування Python?
5. Як встановлюється подія в Python?

Практичні завдання

1. Створіть новий файл Python, та нове вікно. Створіть подію *Button-1*, яка встановить розміри вікна 400x300 пікселів, зелений колір фону і змінить текст заголовка вікна на назву вашого міста.
2. Створіть новий файл Python, та нове вікно. Створіть подію *Button-3*, яка встановить розміри вікна 700x600 пікселів, світло фіолетовий колір фону і змінить текст заголовка вікна на назву вашої школи. Також встановить заборону на змінення розмірів вікна.
3. Створіть новий файл Python, та нове вікно. Створіть подію *KeyPress*, яка встановить розміри вікна 300x200 пікселів, жовтий колір фону і змінить текст заголовка вікна на ваше ім'я та прізвище, встановить мінімальні розміри 200x100 та максимальні 1000x900.
4. Створіть новий файл Python, та нове вікно. Створіть подію *KeyPress*, яка встановить розміри вікна 500x600 пікселів, сірий колір фону і змінить текст заголовка вікна на клас, у якову ви навчаєтесь, встановить мінімальні розміри 400x500 та максимальні 900x1000. Також створіть подію Button-1, яка буде створювати вікно повідомлення із змістом "Я навчаюся у 8 класі!".
5. Створіть новий файл Python, та нове вікно. Дане вікно повинно бути червоного кольору із розмірами 700x400 та заголовком вікна "Вікно №1". Створіть подію *Button-1*, яка буде створювати нове вікно із розміри вікна 400x300 пікселів, зелений колір фону із заголовком вікна "Вікно №2".

Практична робота №3

Тема: Створення об'єктно-орієнтованої програми, що відображає вікно повідомлення.

Для виконання даної практичної роботи необхідно знати:

- основні функції та методи для роботи з вікнами;
- принципи роботи з методом *bind*;
- вміти застосовувати функцію *messagebox.showinfo()* для виведення вікон повідомлень.

Завдання 1

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. Створіть нове вікно світло зеленого кольору з заголовком "Вікно tkinter" та подію *Button-1*, яка змінить розміри цього вікна на 500x500 пікселів та колір на темно зелений. Також створіть подію *Button-3*, при виконанні якої всі зміни буде відмінено.

Завдання 2

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. Створіть довільне вікно з довільним заголовком, розмірами та кольором. Також створіть подію *Button-1*, яка зменшить розміри вікна вдвічі і встановить білий колір фону, а також виведе вікно повідомлення з заголовком "Вікно повідомлення" та текстом "Зміни застосовано". Та після натискання правою кнопкою миші по області вікна всі зміни анулюються та виводиться вікно повідомлення з текстом "Зміни анульовано".

Лекція 11. Створення кнопок та настроювання їх властивостей.

Так як ми з вами вже вміємо створювати вікна та налаштовувати їх властивості, давайте почнемо працювати з їх елементами. І одним із найголовніших елементів управління є кнопка. Саме про створення кнопок, налаштовування їх властивостей та присвоювання їм подій і буде йти мова.

Кнопки створюються дуже просто - за допомогою функції ***Button()*** (з англ. "кнопка"). Тобто працюємо за старою схемою і спочатку дамо кнопці назву, а після знаку дорівнює встановлюємо нашу функцію:

назва_кнопки=**Button**()

Ми вже знаємо, що вікон у нас може бути багато, тому нам потрібно вказати до якого вікна вона відноситься:

назва_кнопки=**Button**(*назва_вікна_до_якого_вона_належить*, *атрибут1*, ...)

Окрім цього ми можемо додавати атрибути (властивості) до нашої кнопки. Такі як:

- **bg=**"колір" - колір кнопки;
- **text=**"текст" - текст кнопки;
- **fg=**"колір" - колір тексту кнопки;
- **font=**"шрифт та розмір шрифту" - шрифт тексту кнопки;
- **width=***число* - ширина кнопки;
- **height=***число* - висота кнопки.

На відміну від розмірів вікна та розмірів інших об'єктів, розміри кнопки вимірюються в кількості символів, які можна розмістити в кнопку. Це буде залежати від розміру тексту в кнопці та розмірів тексту.

Давайте розглянемо, як застосовуються властивості кнопки на прикладі. Всі властивості можна застосувати так як і до вікна, але в даному випадку зручніше їх записати в дужки через кому.

Створимо кнопку *but,* та встановимо для неї всі вище згадані властивості. Нехай ширина кнопки буде 20 пікселів, висота 10 пікселів, колір фону зелений, текст "Ok" червоного кольору, шрифт Times New Roman 12. А розміри кнопки: 4 знаки — ширина і 2 знаки — висота. Давайте поглянемо, як це повинно виглядати:

but=Button(Window, width=4, height=2, bg="green", text="Ok", fg="red", font="Times 12")

Звичайно, атрибути можна вказувати не всі - це за бажанням розробника програми.

На самостійне опрацювання. Знайдіть якомога більше шрифтів, які можна застосувати для кнопок у Python.

Це лише перша частина створення кнопки. Нам ще потрібно її розмістити у вікні, тобто вказати її точне місце знаходження. Це робиться за допомогою метода **place** (з англ. "місце").

Метод *(в об'єктно-орієнтованому програмуванні) — підпрограма (процедура, функція), що використовується виключно разом з об'єктом.*

Має він такі параметри:

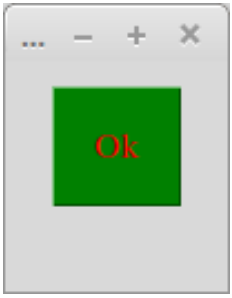
- **x=***число у пікселях* - відступ від лівого краю вікна;
- **y=***число у пікселях* - відступ від верхнього краю вікна.

Застосовується таким чином одразу після створення кнопки:

*назва_кнопки***.place(**x=**"**число**"**, y=**"**число**"**)**

Отже, ми маємо таку просту програму, яка буде виводити на екран нашу кнопку та розміщати її в 20 пікселях від лівого краю, та в 10 пікселях від верхнього краю. Нехай наше вікно буде розмірами 100x100. Маємо (мал. 21):

```
from tkinter import *
Window=Tk()
Window.geometry("100x100")
but=Button(Window, width=4, height=2,
     bg="green", text="Ok", fg="red",
     font="Times 12")
# створення кнопки та встановлення її атрибутів
but.place(x=20, y=10)
# розміщення кнопки
Window.mainloop()
```



Мал. 21

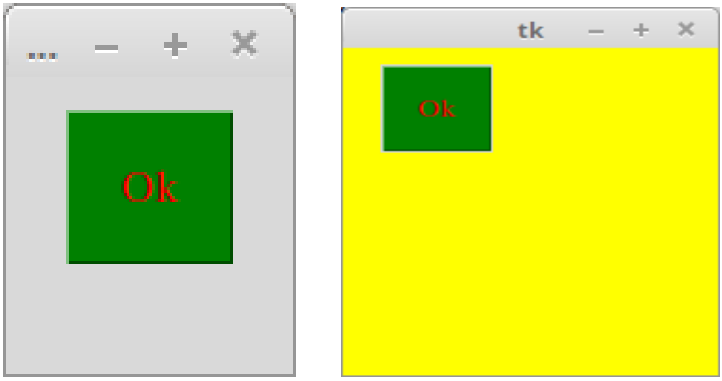
Ну і головна функція кнопок - це об'єкт управління, тобто для нього також можна прив'язувати події. Це робиться так як і з вікнами, але замість назви вікна встановлюються назви кнопок. Тобто:

назва_кнопки.bind("<*подія*>", *назва_функції*)

Давайте для прикладу нашій кнопці присвоїмо подію Button-1, при якій розміри та колір вікна будуть змінюватися. Розміри стануть "200x200", а колір жовтий. Ось як буде виглядати наша програма, функцію назвемо fun1:

```
from tkinter import *
def fun1(event):
       Window.geometry("200x200")
       Window["bg"]="yellow"
Window=Tk()
Window.geometry("100x100")
but=Button(Window,  width=4,  height=2,  bg="green",  text="Ok",
   fg="red", font="Times 12")
# створення кнопки та встановлення її атрибутів
but.place(x=20, y=10)
# розміщення кнопки
but.bind("<Button-1>", fun1)
# встановлення події
Window.mainloop()
```

Тепер давайте спробуємо запустити нашу програму і подивимося на результат (мал. 22).



Мал. 22

Контрольні питання

1. Що таке кнопка?
2. Для чого вона потрібна?
3. Як створити кнопку в мові програмування Python?

4. Які властивості можна їй надати?
5. Як розмістити кнопку у вікні модуля tkinter?
6. Що називають методом у програмуванні?
7. Що таке place, та як з ним працювати?

Практичні завдання

1. Створіть новий файл Python та нове вікно із кнопкою. Розміри вікна 400x300, заголовок "Вікно №1". Кнопка розміщена в 100 пікселях від лівого краю та в 90 пікселях від правого краю, текст кнопка "Розфарбуй". Колір кнопки та тексту довільний. При натисканні лівою клавішею миші на кнопку розміри вікна зміняться на 650x560, колір фону вікна стане зеленим а кнопки блакитним.
2. Створіть новий файл Python та нове вікно із кнопкою. Розміри вікна 500x800, заголовок "Вікно №2". Кнопка розміщена в 200 пікселях від лівого краю та в 390 пікселях від правого краю, текст кнопка "Ок". Колір кнопки та тексту довільний. При натисканні лівою клавішею миші на кнопку колір фону вікна стане фіолетовим а кнопки жовтим та створиться вікно із повідомленням "Завдання виконано!".
3. Створіть новий файл Python та нове вікно із заголовком "Це вікно Python". Створіть нову кнопку та розмістіть її приблизно по центру даного вікна. Колір кнопки рожевий, текст кнопки "Змінити", колір тексту блакитний. При натисканні правою клавішею миші по даній кнопці розміри вікна стануть 560x435 пікселів, а колір вікна стане жовтим. Також буде виводитися вікно повідомлення із заголовком "Виконано" та із текстом "Зміни застосовані!".

Лекція 12. Написи та їх властивості

Ми ознайомилися з таким елементом, як кнопка, який зазвичай є елементом управління і відповідає за динаміку в програмах написаних мовою Python. Але існують і статичні об'єкти, які не впливають на динаміку в програмі, і одним із них ми зараз познайомимося. Найпростіший об'єкт вікна – це *напис*, тобто звичайний рядок тексту, або декілька рядків. Звичайно, як і інші об'єкти, напис має свої властивості, і про всі ці властивості буде зараз йти мова.

***Напис** — це елемент вікна за допомогою якого можна виводити текстові повідомлення.*

Не будемо гаяти час і почнемо одразу з того, як розмістити напис в нашому вікні. Робиться це дуже просто: аналогічно, як із кнопкою, кожному рядку потрібно дати ім'я та вказати функцію для його створення. **Label()** – функція для створення напису:

назва_напису=**Label**(*вікно*, *атрибут1*...)

Так само у дужках ми вказуємо місцезнаходження напису (вікно) та атрибути. Атрибути залишилися майже ті ж самі:

- **bg=“**колір**”** - колір напису;
- **text=“**текст**”** - текст напису;
- **fg=“**колір**”** - колір тексту напису;
- **font=“**шрифт та розмір шрифту**”** - шрифт тексту напису та його розміри;

Для виведення на екран користуємося тим самим методом *place()*, вказуючи при цьому відступ від лівого краю вікна та відступ від верхнього краю вікна (*x* та *y* відповідно).

Отже, давайте створимо для прикладу напис зеленого кольору з текстом чорного кольору, шрифтом Calibri 12, а зміст повідомлення буде "Привіт світ!". Дамо назву нашому напису label та розмістимо його в 30 пікселях від лівого краю і в 40 пікселях від правого краю вікна. Маємо наступний програмний код:

label1=Label(Window, text="Привіт світ!", bg="green", fg="black", font="Calibri 12")

label1.place(x=30, y=40)

Напис представляє собою стрічку із тексту, тому для того, щоб перенести якийсь текст на наступний рядок в текст вставляється код переносу. Виглядає він так "**\n**". Він ставиться перед словом або символом, після якого всі слова переносяться на наступний рядок. Наприклад, для переносу слова "світ" у нашій програмі на наступний рядок потрібно перед цим словом поставити ці символи. Тобто ми маємо текст напису "Привіт \n світ!".

Так як напис — це статичний об'єкт, то до нього не часто прив'язують події, але його створення або змінення властивостей може слугувати командою функції, яка буде виконуватися при виконанні події.

Тож давайте для прикладу створимо вікно та кнопку, а от встановлювати властивості ми не будемо, лише розмістимо кнопку, вказавши відступ в 30 пікселів від лівого краю та 40 пікселів від верхнього краю, розміри вікна встановимо 300x200 пікселів:

```
from tkinter import *
Window=Tk()
Window.geometry("300x200")
but=Button(Window)
but.place(x=30, y=40)
Window.mainloop()
```

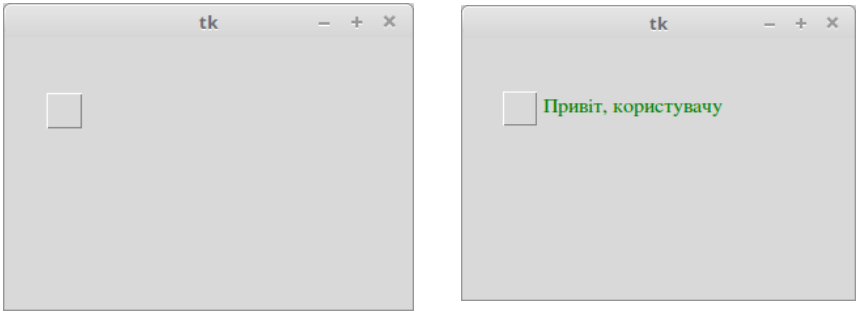
Тепер при натисканні на кнопку лівою клавішею миші буде створюватися та виводитися на екран напис з текстом "Привіт, користувачу" поряд з кнопкою (x=60, y=40), зеленого кольору та шрифтом Times 12.

Створимо подію *Button-1* та функцію *fun1* і додамо відповідний код:

```
from tkinter import *
def fun1(event):
      label1=Label(Window, text="Привіт, користувачу",
          font="Times 12", fg="green")
      # створення напису
      label1.place(x=60, y=40)
      # розміщення напису
Window=Tk()
Window.geometry("300x200")
```

```
but=Button(Window)
but.place(x=30, y=40)
but.bind("<Button-1>", fun1)
# присвоєння події до кнопки
Window.mainloop()
```

Запустимо нашу програму (мал. 23)



Мал. 23

Контрольні питання

1. Що таке напис?
2. Як створити напис?
3. Які властивості має напис?
4. Як перенести текст на наступний рядок у написі?

Практичні завдання

1. Створіть новий файл Python та вікно із заголовком "Вікно №1" та з розмірами 500x500. Додайте до вікна напис із текстом "Це вікно було створено в середовищі IDLE". Колір напису повинен бути синім, а колір тексту білий, шрифт тексту "Arial 14". Розмістити його довільно.
2. Створіть новий файл Python та вікно із заголовком "Вікно №2" та з розмірами 500x500. Додайте до вікна напис із текстом "Функція виконана", який буде з'являтися після виконання події KeyPess відносно вікна. Колір напису повинен бути зеленим, а колір тексту білий, шрифт тексту "Calibri 14". Розмістити його довільно.
3. Створіть новий файл Python та вікно із заголовком "Вікно №3" та з розмірами 500x500. Створіть кнопку із текстом "Розфарбуй", розміщену на 200 пікселів від лівого та верхнього краю. Додайте до кнопки подію Button-3, при виконанні якої колір вікна стане блакитним та буде

створюватися напис із текстом "Це вікно було створено в середовищі IDLE". Колір напису повинен бути синім, а колір тексту білий, шрифт тексту "Arial 14". Напис буде розміщуватися на 200 пікселів від лівого краю та на 250 пікселів від верхнього краю.

Практична робота №4

Тема: Створення програми з кнопками та написами.

Для виконання даної практичної роботи необхідно знати:

- основні функції для створення кнопок та написів;
- принципи роботи з методом *bind*;
- атрибути, які можна застосувати для кнопки або напису.

Завдання 1

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. Створіть нове вікно, не вказуючи жодних властивостей. Також створіть кнопку із текстом "Змінити". Розмістіть її в довільному положенні. При натисканні на дану кнопку лівою клавішею миші заголовок вікна зміниться на "Вікно зі змінами", а колір на рожевий.

Завдання 2

Відкрийте середовище розробки програмного забезпечення IDLE та створіть новий файл Python. Створіть нове вікно з розмірами 600x400 пікселів блакитного кольору з забороною змінювати розміри вікна. Створіть дві кнопки із текстами "Кнопка №1" та "Кнопка №2" відповідно. Розмістіть їх в довільних місцях. При натисканні на кнопку №1 колір вікна стане світло блакитним, а при натисканні на кнопку №2 внизу вікна з'явиться напис з текстом "Ви натиснули на Кнопку 1".

Лекція 13. Текстове поле його функції та властивості

Все, що ми робили до цього, було націлене лише на створення об'єктів, надання їм властивостей та їх зміну. Тепер же ми почнемо працювати із даними та способами їх отримання.

Ми пам'ятаємо, що для введення даних в командному режимі використовується функція *input()*, а у модулі tkiner існують спеціальні об'єкти для введення даних, такі як текстове поле, прапорець та перемикачі.

13.1 Текстове поле

Тексове поле - це об'єкт, що використовується при побудові графічного інтерфейсу користувача. *Мета текстового поля - дозволити користувачеві ввести текстову інформацію, яка буде використовуватися програмою.*

Модуль tkinter має можливість створити текстове поле у програмі на мові Python за допомогою функції ***Entry***():

назва_поля=**Entry**(*назва_вікна*, *атрибут1*...)

До текстового поля застосовуються наступні атрибути:

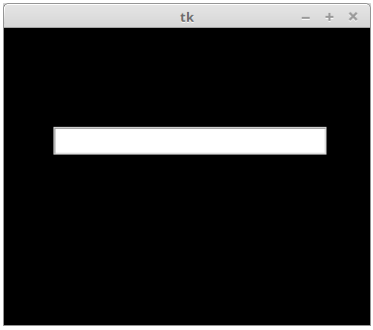
- **bg=**"колір" - колір поля;
- **fg=**"колір" - колір тексту поля;
- **font=**"шрифт та розмір шрифту" - шрифт тексту (встановлює висоту поля);
- **width=***число* – ширина кнопки, зазначається у кількості знаків;
- **bd=***число* — ширина контуру поля, зазначається у пікселях.

Наприклад, ми маємо вікно Window1, розміри вікна 370x300, колір нехай буде чорний. Створимо в ньому текстове поле білого кольору з шириною у 30 знаків, а шрифтом *Times 13*. Розмістимо його в 50 пікселів від лівого краю та в 100 від верхнього краю. Ось як це буде виглядати у програмному коді:

```
from tkinter import *
Window1=Tk()
Window1.geometry("370x300")
Window1["bg"]="black"
entry1=Entry(Window1, bg="white", width=30, font="Times 13")
```

```
# створення текстового поля
entry1.place(x=50, y=100)
# розміщення текстового поля
Window1.mainloop()
```

Запустимо програму та побачимо що з цього вийде (мал. 24).



Мал. 24

13.2 Отримання даних та створення повноцінних програм

Саме по собі текстове поле не може використовуватися для отримання даних, поле – лише допоміжний об'єкт. Тобто, *entry1 – це лише назва об'єкта, а не назва змінної, яка містить дані введені у це поле.* Для отримання даних з поля використовується спеціальний метод ***get()***.

Застосовується він так:

назва поля/прапорця/перемикача.**get()**

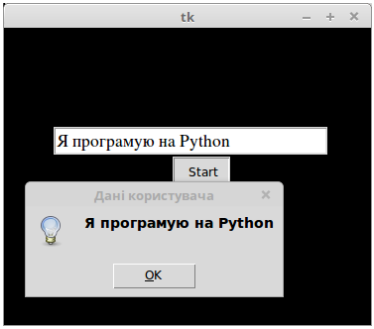
Тобто, entry1 – це назва поля, а entry1.get() - це значення цього поля. Я думаю, що на прикладі буде зрозуміліше.

Отже, нехай ми маємо те ж вікно і те ж поле. Тепер створимо кнопку з текстом "Start" та розмістимо її в 170 пікселях від лівого краю та в 130 від верхнього. Присвоїмо їй подію *Button-1*, при виконанні якої буде створюватися вікно повідомлення із даними, які ввів користувач. Заголовком вікна повідомлення зробимо "Дані користувача". Ось, як це буде виглядати:

```
from tkinter import *
def start(event):
    messagebox.showinfo("Дані користувача", entry1.get())
    # виведення вікна повідомлення із даними користувача
Window1=Tk()
Window1.geometry("370x300")
Window1["bg"]="black"
entry1=Entry(Window1, bg="white", width=30, font="Times 13")
# створення текстового поля
```

```
entry1.place(x=50, y=100)
# розміщення текстового поля
but=Button(Window1, text="Start")
but.place(x=170, y=130)
but.bind("<Button-1>", start)
Window1.mainloop()
```

Як ви можете бачити замість повідомлення в функції *messagebox.showinfo()* було встановлено дані отримані з текстового поля. **Зверніть увагу!** *Лапки в даному випадку ставити зовсім не потрібно, так як за замовчуванням дані текстового поля є рядком тексту. Якщо потрібно отримати числа (цілі або дійсні) — потрібно використати функції int() або float() відповідно.*



Мал. 25

Маємо наступне вікно (мал. 25).

Ну і звичайно давайте розглянемо щось більш складніше. Нехай ми матимемо те саме вікно з тими самими параметрами. Лише додамо до нього напис із початковим текстом "Введіть дані" який буде розміщено в 150-ти пікселях від лівого краю та в 50-ти від верхнього. В залежності від того, що введе користувач, воно буде відображатися замість напису "Введіть дані" (звичайно після натискання кнопки). Так це повинно виглядати у коді:

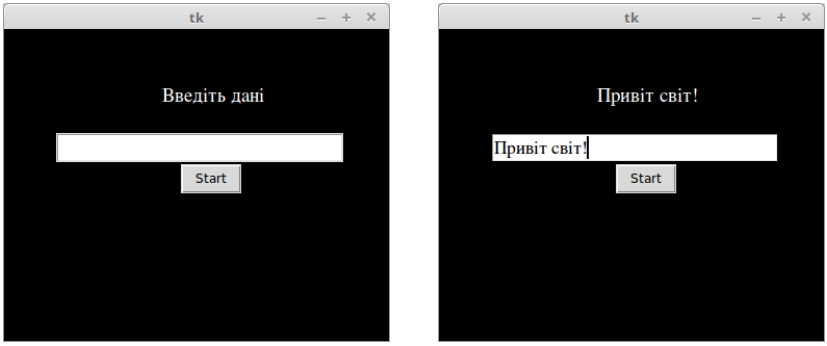
```
from tkinter import *
def start(event):
        label["text"]=entry1.get()
        # зміна тексту напису на введений користувачем
Window1=Tk()
Window1.geometry("370x300")
Window1["bg"]="black"
entry1=Entry(Window1, bg="white", width=30, font="Times 13")
# створення текстового поля
```

```
entry1.place(x=50, y=100)
# розміщення текстового поля
label=Label(Window1, text="Введіть дані", bg="black", fg="white",
   font="Times 14")
# створення напису
label.place(x=150, y=50)
# розміщення напису
but=Button(Window1, text="Start")
but.place(x=170, y=130)
but.bind("<Button-1>", start)
Window1.mainloop()
```

Як ви бачите, в даному випадку ми плануємо вносити змінну тексту напису. Зміна, або присвоєння одного атрибуту об'єкту застосовується так:

ім'я об'єкта **[“**назва атрибуту**”]**=*значення*

Отже, після натискання кнопки, текст напису буде дорівнювати даним отриманим з текстового поля. Для прикладу запустимо програму і введемо "Привіт світ!" (мал. 26)

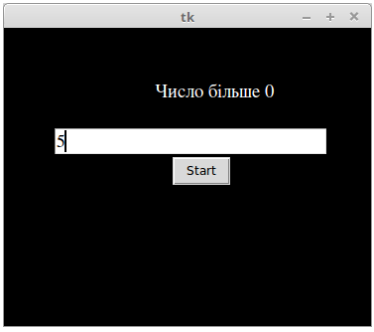


Мал. 26

Для створення більш складних задач нам буде потрібно згадати програмування у командному інтерфейсі, адже всі дії з даними та конструкції можна застосувати до даних у графічному інтерфейсі.

Давайте вдосконалимо нашу програму, щоб вона була цікавішою. Нехай користувач буде вводити числа в наше поле і якщо це число більше 0 то буде виводитися напис "Число більше 0", якщо менше то "Число менше 0" і відповідно "Число дорівнює 0". Отже, маємо:

```
from tkinter import *
def start(event):
       if float(entry1.get())>0:
              label["text"]="Число більше 0"
       elif float(entry1.get())==0:
              label["text"]="Число дорівнює 0"
       else:
              label["text"]="Число менше 0"
      # зміна тексту напису в залежності від значення, яке ввів
      # користувач
Window1=Tk()
Window1.geometry("370x300")
Window1["bg"]="black"
entry1=Entry(Window1, bg="white", width=30, font="Times 13")
# створення текстового поля
entry1.place(x=50, y=100)
# розміщення текстового поля
label=Label(Window1, text="Введіть дані", bg="black", fg="white",
   font="Times 14")
# створення напису
label.place(x=150, y=50)
# розміщення напису
but=Button(Window1, text="Start")
but.place(x=170, y=130)
but.bind("<Button-1>", start)
Window1.mainloop()
```



Мал. 27

Як ми бачимо, в залежності від того яким буде значення дійсного числа, яке введе користувач атрибут text напису label буде змінюватися по різному (мал. 27).

Контрольні питання

1. Дайте визначення поняттю текстове поле.
2. Які атрибути можна застосувати для поля?
3. Який метод слугує для отримання даних з текстового поля?

Практичні завдання

1. Створіть новий файл Python та підключіть відповідний модуль. Створіть нове вікно світло червоного кольору із розмірами 600 на 700 пікселів із забороною змінювати розміри вікна. Створити текстове поле (всі атрибути встановити за власним бажанням), кнопку поруч із текстовим полем (текст "Ок", колір білий, шрифт "Calibri 13"), та напис із початковим текстом «Ви нічого не ввели». Після натискання на кнопку текст напису повинен змінитися на введений користувачем.

2. Створіть вікно жовтогарячого кольору з розмірами 600x600 та з заголовком "Перше вікно". Створіть текстове поле будь-якого кольору. Ширина поля 30 знаків та шрифт "Calibri 12". Створіть напис, колір тексту темно-синій, шрифт аналогічний полю. При натисканні правою клавішею по області вікна, текст напису буде змінюватися на введений користувачем та виводитися на екран вікно повідомлення з заголовком "Підтвердження події" та з змістом "Дія виконана!".

3. Створіть вікно з розмірами 875x578 з заголовком «Це є вікно :)». Створіть текстове поле з шириною у 35 знаків, та розмістити його на 100 пікселів від лівого краю та на 100 пікселів від верхнього краю. В поле користувач буде вводити ціле число. При натисканні правою клавішею миші по області поля на екран буде виводитися вікно повідомлення із добутком введеного числа на нього самого. Заголовок вікна повідомлення та колір фону головного вікна встановити самостійно.

Лекція 14. Перемикачі та прапорці

Ще одним із способів введення даних є перемикачі та прапорці. Вони представлені у вигляді невеликих списків із можливістю вибору одного (перемикачі) або декількох варіантів (прапорці).

14.1 Перемикачі

Почнемо із перемикачів, їх ще називають радіоточками (у формі позначаються так ). Так, як перемикачі слугують для вибору одного варіанта серед запропонованих, то вони представляються групою, в якій активним може бути тільки один перемикач. Якщо перемикач обрано, то всередині нього з'являється позначка  .

Але їх створити трішки складніше, ніж будь-який об'єкт, який ми створювали раніше. Просто перемикач — це не єдиний об'єкт, а група об'єктів, тому спочатку створюється група перемикачів, а потім самі перемикачі за допомогою функції ***Radiobutton()***:

ім'я_групи_перемикачів=**IntVar()**

назва_ перемикача1=**Radiobutton(***назва_вікна*, **text**="Текст перемикача1", **variable**=*ім'я_групи_перемикачів*, **value**=*значення_перемикача1***)**

назва_перемикача2=**Radiobutton(***назва_вікна*, **text**="Текст перемикача2", **variable**=*ім'я_групи_перемикачів*, **value**=*значення_перемикача2***)**

…

value – значення перемикача, яке буде набувати група перемикачів в залежності від вибору користувача. Зазвичай значення перемикача – це натуральні числа, починаючи від 1, хоча ви можете встановити і інші. Та головне щоб для всіх перемикачів вони були різними.

Переходимо далі, і як ви вже зрозуміли ім'я групи перемикачів – це цілочисельна (IntVar) змінна із результатом вибору користувача програми. Та для того, щоб отримати значення, яке набула група перемикачів потрібно застосувати метод *get(),* з яким ми вже знайомі.

Нехай, ми маємо групу перемикачів *grupa1*, перший перемикач має значення 1, а другий – 2. Отже, якщо користувач вибере перший перемикач, то *grupa1.get()* буде мати значення 1.

Аргументи *variable, value* та *text* є обов'язковими, без них перемикачі не будуть представляти із себе нічого. Але є й необов'язкові атрибути:

- **bg="**колір**"** - колір фону;
- **fg="**колір**"** - колір тексту;
- **font="**шрифт та розмір шрифту**"** - шрифт тексту кнопки.

І звичайно, кожен перемикач потрібно розмістити за допомогою методу *place*(). Але це ми вже робити вміємо.

Для прикладу створимо вікно із розмірами 400x400, та створимо в нього групу перемикачів, а саме три перемикачі із варіантами кольорів (жовтий, рожевий та світло зелений). І перший розмістимо в 100 пікселях від лівого краю та в 200 пікселях від верхнього краю, а кожен наступний на 20 пікселів нижче. Програмний код буде виглядати так:

```
from tkinter import *
Window=Tk()
Window.geometry("400x400")
perem=IntVar()
# створення групи перемикачів perem
perem1=Radiobutton(Window, text="Жовтий", variable=perem, value=1)
perem1.place(x=100, y=200)
# створення та розміщення першого перемикача perem1
perem2=Radiobutton(Window, text="Рожевий", variable=perem,
   value=2)
perem2.place(x=100, y=220)
# створення та розміщення першого перемикача perem2
perem3=Radiobutton(Window, text="Світло зелений", variable=perem,
   value=3)
perem3.place(x=100, y=240)
# створення та розміщення першого перемикача perem3
Window.mainloop()
```

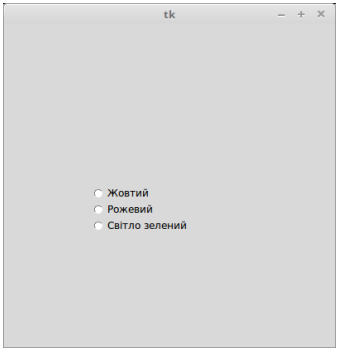
При запуску вікно буде виглядати так (мал. 28) Як ми бачимо, з самого початку жоден перемикач не вибраний, якщо ж ми хочемо, щоб на початку був одразу вибраний один із перемикачів, нам потрібно розмістити після оголошення змінної групи перемикачів наступну конструкцію:

*ім'я_групи_перемикачів***.set(***значення_перемикача***)**

Примітка. Метод *get()*, з яким ми вже працювали призначений для одержання значення різних об'єктів, а метод ***set()*** для встановлення значення.

Тобто, якщо ми маємо групу перемикачів *perem*, та два перемикача із значеннями 1 та 2, і ми хочемо аби одразу був вибраний перший перемикач, ми застосуємо наступний рядок коду:

```
perem.set(1)
```



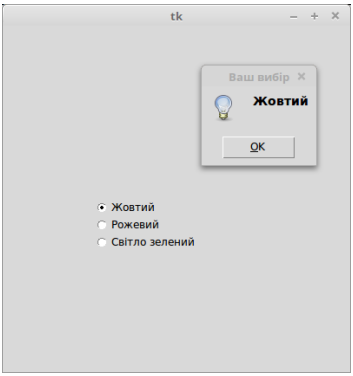
Мал. 28

Ну і тепер створимо динамічну програму, яка буде, при натисканні на праву клавішу миші по області вікна, виводити вікно повідомлення з заголовком "Ваш вибір" та з варіантом, який обрав користувач. Створимо нову подію та функцію. Але перемикачі та прапорці працюють не так, як текстове поле, тому ми зможемо отримати лише номер вибраного перемикача, а отже будемо користуватися розгалуженням:

```
from tkinter import *
def result(event):
      if perem.get()==1:
            pov="Жовтий"
      # значення змінної pov, якщо користувач обере жовтий колір
      elif perem.get()==2:
            pov="Рожевий"
      # значення змінної pov, якщо користувач вибере рожевий колір
      else:
            pov="Світло зелений"
      # значення змінної pov, якщо користувач вибере зелений колір
      massegebox.showinfo("Ваш вибір", pov)
      # виведення вікна повідомлення із даними які містяться в
      # змінній pov
Window=Tk()
Window.geometry("400x400")
perem=IntVar()
# створення групи перемикачів perem
perem1=Radiobutton(Window, text="Жовтий", variable=perem, value=1)
perem1.place(x=100, y=200)
```

```
# створення та розміщення першого перемикача perem1
perem2=Radiobutton(Window, text="Рожевий", variable=perem,
   value=2)
perem2.place(x=100, y=220)
# створення та розміщення першого перемикача perem2
perem3=Radiobutton(Window, text="Світло зелений", variable=perem,
   value=3)
perem3.place(x=100, y=240)
# створення та розміщення першого перемикача perem3
Window.bind("<Button-3>", result)
Window.mainloop()
```

Зверніть увагу! Для більш зручного виведення інформації створено додаткову змінну *pov*. *pov* - змінна, яка містить повідомлення для виведення, і в залежності від вибору користувача, значення цієї змінної буде змінюватися.



Мал. 29

Запустимо нашу програму та поспостерігаємо за її виконанням (мал. 29).

14.2 Прапорці

Що стосується прапорців, то ці елементи управління не об'єднуються в групи, тому, що активних прапорців може бути не обмежена кількість. Виглядають вони так ☐ (у неактивному стані) і так ☑ (в активному стані). Прапорці слугують для вибору декількох варіантів і тому кожен прапорець має свою окрему змінну із значенням.

Для створення прапорців призначена функція ***Checkbutton()***, і застосовується вона так:

назва_змінної=**IntVar**()

ім'я_прапорця=**Checkbutton**(*назва_вікна*, **text**="Текст прапорця", **variable**=*назва+змінної*, **onvalue**=*значення_коли_прапорець_ввімкнутий*, **offvalue**=*значення_коли_прапорець_вимкнутий*)

Примітка! Зазвичай в якості значення прапорця встановлюють 1 та 0 (в активному та неактивному стані), але вибір все одно залишається за вами.

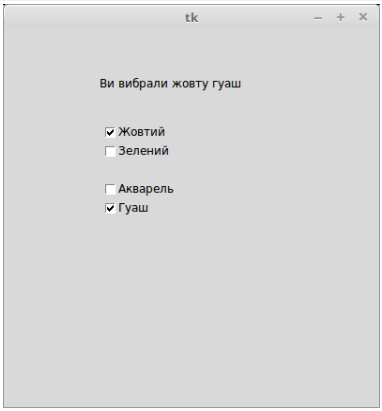
Ну і для прикладу візьмемо вікно з розмірами 500x500 пікселів та створимо напис із початковим текстом "Не вибрано". Також створимо чотири прапорці (перші два із варіантами кольорів та інші двоє із варіантами типів фарб). В залежності від нашого вибору текст напису буде змінюватися (у якості прикладу розглянемо лише 2 комбінації). Щоб реалізувати зміну тексту напису створимо подію **Button-1** для напису. Ось що ми маємо:

```
from tkinter import *
def newtext(event):
	if prap1.get()==1 and prap4.get()==1:
		label["text"]="Ви вибрали жовту гуаш"
	# якщо перший і четвертий прапорці вибрані, то вивести
	# відповідне повідомлення
	elif prap2.get()==1 and prap3.get()==1:
		label["text"]="Ви вибрали зелену акварель"
	# якщо другий і третій прапорці вибрані, то вивести
	# відповідне повідомлення
Window=Tk()
Window.geometry("400x400")
label=Label(Window, text="Не вибрано")
label.place(x=100, y=50)
prap1=IntVar()
prap2=IntVar()
prap3=IntVar()
prap4=IntVar()
# створення змінних прапорців
prapor1=Checkbutton(Window,	text="Жовтий",	variable=prap1,
onvalue=1, offvalue=0)
prapor1.place(x=100, y=100)
# створення та розміщення prapor1
prapor2=Checkbutton(Window,	text="Зелений",	variable=prap2,
onvalue=1, offvalue=0)
prapor2.place(x=100, y=120)
# створення та розміщення prapor2
```

```
prapor3=Checkbutton(Window,    text="Акварель",    variable=prap3,
onvalue=1, offvalue=0)
prapor3.place(x=100, y=160)
# створення та розміщення prapor3
prapor4=Checkbutton(Window,      text="Гуаш",      variable=prap4,
onvalue=1, offvalue=0)
prapor4.place(x=100, y=180)
# створення та розміщення prapor4
label.bind("<Button-1>", newtext)
Window.mainloop()
```

Зверніть увагу! Ми не повинні забувати, що для того, щоб отримати значення прапорця ми повинні користуватися методом *get()*.

Отже запустимо нашу програму та поспостерігаємо за її виконанням та результатом (мал. 30).



Мал. 30

Контрольні питання

1. Що представляють собою прапорці та перемикачі?
2. Як називаються функції для створення прапорців і перемикачів?
3. Чому для створення перемикачів та прапорців потрібно створювати додаткові змінні?
4. Що означає IntVar?
5. Для чого слугує метод set?

Практичні завдання

1. Вдосконалити останній приклад, додавши до програми всі можливі комбінації вибору.
2. Створіть новий файл Python та нове вікно із заголовком "Магазин морозива". Створіть дві групи перемикачів. Над першою групою створіть напис "Оберіть тип морозива". В першій групі буде три перемикачі із типами морозива: ванільне, шоколадне, фруктове. Над другою групою створіть напис "Оберіть розмір ріжку". В цій групі буде також три перемикачі із розмірами ріжку: маленький, середній та великий. В залежності від вибору користувача на екран буде виводитися відповідне

вікно з повідомленням. Наприклад, якщо користувач вибере фруктове морозиво у маленькому ріжку, то на екран виведеться вікно з повідомленням "Ви вибрали фруктове морозиво у маленькому ріжку".

3. Створіть новий файл Python та нове вікно із заголовком "Магазин морозива 2". Створіть групу перемикачів. Над першою групою створіть напис "Оберіть тип морозива". В групі буде три перемикачі із типами морозива: ванільне, шоколадне, фруктове. Поруч створити два прапорці із типами присипки: шоколадна присипка, кокосова стружка. Також над ними напис "Оберіть тип присипки". В залежності від вибору користувача буде виводитися вікно повідомлення із вибором користувача. **Примітка.** Розглянути всі можливі випадки.

Лекція 15. Зображення основних графічних об'єктів у Python

Малювання у Python – це, мабуть, найцікавіша частина у всьому курсі програмування. Отже, і ми попрактикуємося над створенням графічних примітивів, які є складовою частиною векторної графіки.

15.1 Полотно

Але почнемо ми із розміщення полотна. **Полотно для малювання** *— частина вікна (або все вікно) у якій може бути здійснене малювання об'єктів.*

Для створення полотна існує функція ***Canvas()*** і застосовується вона таким чином:

назва_полотна=**Canvas**(*назва_вікна*, *атрибут1*…)

До полотна можна застосувати наступні атрибути:

- **width**=*число_у_пікселях* — ширина полотна;
- **height**=*число_у_пікселях* — висота полотна;
- **bg**=**"**колір**"** — колір фону.

Зверніть увагу! Також нам потрібно обов'язково розмістити наше полотно у вікні за допомогою методу *place()*. Як його застосовувати ми вже знаємо.

15.2 Графічні примітиви

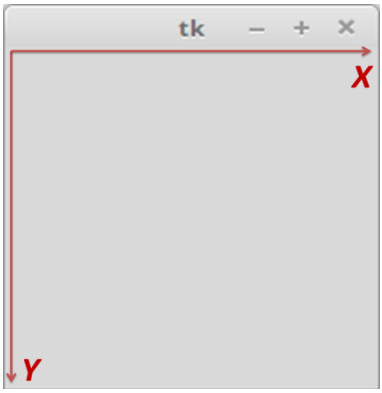
До графічних примітивів у Python відносяться лінія, прямокутник, еліпс та многокутник.

Для їх побудови створимо вікно розмірами 500x500 та полотно світло-блакитного кольору з такими ж розмірами:

```
from tkinter import *
Window=Tk()
Window.geometry("500x500")
canv=Canvas(Window, width=500, height=500, bg="light blue")
canv.place(x=0, y=0)
Window.mainloop()
```

При побудові кожного об'єкту ми повинні перед собою малювати уявні координатні промені. Від верхнього лівого краю буде починатися відлік по координатним променям *x* та *y* (мал. 31). Значення *x* та *y*

вимірюються в пікселях. **Зверніть увагу!** Осі координат розміщені не так, як ви звикли їх бачити. Про це не можна забувати.



Мал. 31

Отже, існують наступні 4 функції для зображення основних графічних примітивів:

- **create_line()** – для побудови лінії;
- **create_rectangle()** – для побудови прямокутника;
- **create_oval()** – для побудови еліпса;
- **create_polygon()** – для побудови довільного многокутника.

Тепер про кожну із функцій окремо.

15.2.1 Лінія

Функція *create_line()* застосовується наступним чином:

назва_полотна.**create_line**([x1,y1], [x2,y2], *аргумент1*…)

x1 та **y1** – координати початкової точки;

x2 та **y2** – координати кінцевої точки.

Як ви вже помітили, то координати точок записуються у квадратних дужках через кому.

Зверніть увагу! Координати початку та кінця лінії повинні знаходитися у межах полотна, щоб вона відображалася.

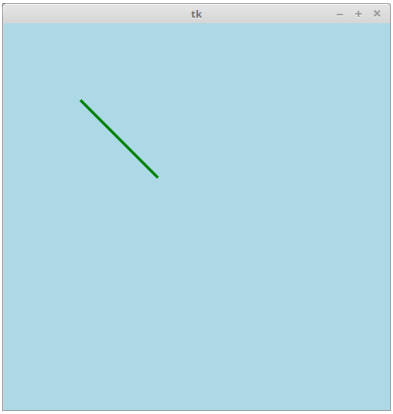
Також для лінії притаманні такі атрибути:

- **fill=**"колір" - колір лінії;
- **width=***число_у_пікселях* - товщина лінії.

Для прикладу побудуємо лінію зеленого кольору у нашому полотні із товщиною 3 пікселі, початок буде в точці [100, 100], а кінець в точці [200, 200]:

```
from tkinter import *
Window=Tk()
Window.geometry("500x500")
```

```
canv=Canvas(Window, width=500,
height=500, bg="light blue")
canv.place(x=0, y=0)
canv.create_line([100,100],[200,200],
   width=3, fill="green")
Window.mainloop()
```



Мал. 32

Подивимося, як це буде виглядати при запуску програми (мал. 32)

15.2.2 Прямокутник

Побудова прямокутника, як і лінії буде відбуватися лише за двома точками, а саме за координатами точки верхнього лівого кута та точки нижнього правого кута за допомогою функції *create_rectangle():*

назва_полотна.**create_rectangle**([x1,y1], [x2,y2], *аргумент1*…)

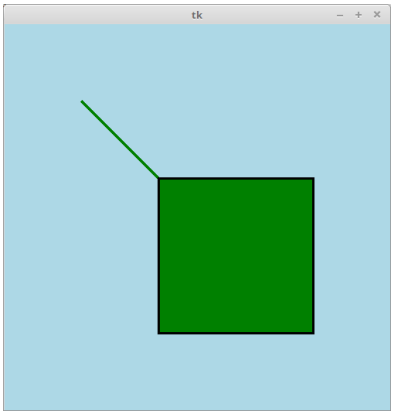
x1 та **y1** – координати точки верхнього лівого кута;

x2 та **y2** – координати точки правого нижнього кута.

Для прямокутника має такі атрибути:

- **fill=**"колір" - колір фону об'єкта;
- **outline=**"колір" - колір контуру;
- **width=***число_у_пікселях* - товщина контуру.

Отже, добудуємо до нашого малюнка прямокутник, а саме квадрат із координатами [200,200], [400, 400] зеленого кольору із товщиною контуру — 3 пікселі (мал.):



Мал. 33

```
from tkinter import *
Window=Tk()
Window.geometry("500x500")
canv=Canvas(Window, width=500,
height=500, bg="light blue")
canv.place(x=0, y=0)
canv.create_line([100,100],
[200,200], width=3, fill="green")
canv.create_rectangle([200,200], [400,
```

```
   400], width=3, fill="green")
Window.mainloop()
```

15.2.3 Еліпс

З еліпсом складніше, адже при створенні еліпсу потрібно вказувати координати уявного прямокутника описаного навколо нього:

*назва_полотна***.create_oval**([x1,y1], [x2,y2], *аргумент1*…)

x1 та **y1** – координати точки верхнього лівого кута уявного прямокутника;

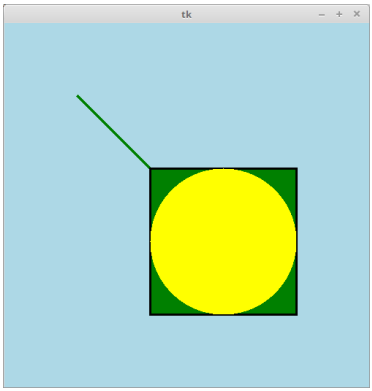
x2 та **y2** – координати точки правого нижнього кута уявного прямокутника.

Для еліпса притаманні такі ж атрибути, як і до прямокутника: колір фону, товщина та колір контуру.

Щоб зрозуміти принцип побудови еліпса, побудуємо його із такими ж координатами як і квадрат. Встановимо жовтий колір фону та приберемо контур:

```
from tkinter import *
Window=Tk()
Window.geometry("500x500")
canv=Canvas(Window, width=500, height=500, bg="light blue")
canv.place(x=0, y=0)
canv.create_line([100,100], [200,200], width=3, fill="green")
canv.create_rectangle([200,200], [400,400], width=3, fill="green")
canv.create_oval([200,200], [400, 400], width=0, fill="yellow")
Window.mainloop()
```

Маємо наступне вікно (мал. 34).



Мал. 34

Зверніть увагу! Так як квадрат був створений раніше, він розміщений за кругом. За таким принципом розміщуються всі об'єкти.

15.2.4 Довільний многокутника

Тобто окрім чітких фігур можна будувати довільні фігури, які складаються із 3-х та більше вершин:

*назва_полотна***.create_polygon**([x1,y1],

[x2,y2]… [xn,yn], *аргумент1*…)

x1 та **y1** – координати першої вершини многокутника;

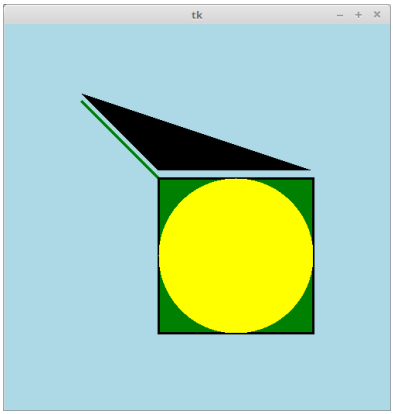
x2 та **y2** – координати другої вершини многокутника;

...

xn та **yn** – координати n-ї вершини многокутника.

Для довільного многокутника притаманні такі атрибути:

- **fill=**"колір" - колір фону об'єкта;
- **outline=**"колір" - колір контуру;
- **width=***число_у_пікселях* - товщина контуру;
- **smooth=***True/False* – згладжування контурів.



Мал. 35

Ну і для прикладу добудуємо трикутник по контуру вже сформованої фігури (мал. 35):

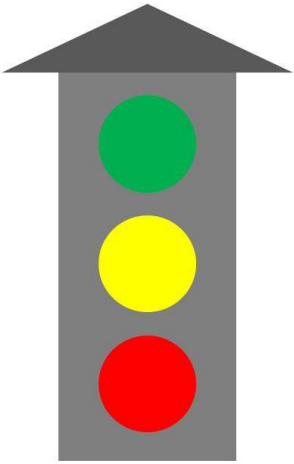
```
from tkinter import *
Window=Tk()
Window.geometry("500x500")
canv=Canvas(Window, width=500, height=500, bg="light blue")
canv.place(x=0, y=0)
canv.create_line([100,100], [200,200], width=3, fill="green")
canv.create_rectangle([200,200],      [400,      400],      width=3,
fill="green")
canv.create_oval([200,200], [400,400], width=0, fill="yellow")
canv.create_polygon([100, 90], [200,190], [400, 190],
   smooth=False)
Window.mainloop()
```

Контрольні питання

1. Що таке полотно та яка його головна функція?
2. Які функції для побудови графічних примітивів ви знаєте?
3. Скільки точок потрібно для побудови: а) лінії; б) прямокутника; в) еліпса; г) довільного многокутника?

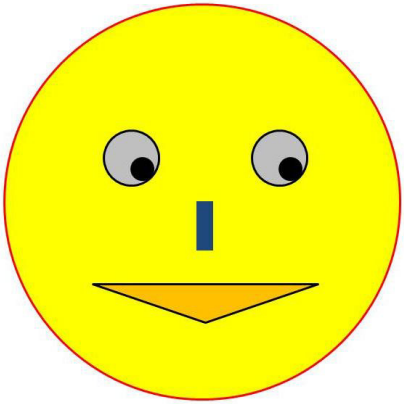
Практичні завдання

Створіть новий файл Python та нове вікно і полотно. Побудуйте наступні малюнки засобами мови Python (мал. 36).

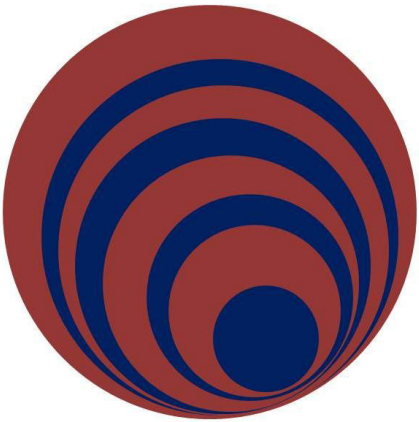


1.

2.



3.



4.

Мал. 36

Додаток

Основні функції модуля math

math.copysign(x, y**)** - повертає число, що має модуль такий же, як і у числа *x*, а знак - як у числа *y*;

math.fabs(x**)** - модуль *x*;

math.factorial(x**)** - факторіал числа *x*;

math.isfinite(x**)** - перевіряє чи є *x* число;

math.modf (x**)** - повертає дробову і цілу частину числа *x*. Обидва числа мають той же знак, що і *x*;

math.sqrt(x**)** - квадратний корінь з *x*;

math.acos(x**)** - арккосинус *x* (*x* вказується в радіанах);

math.asin(x**)** - арксинус *x* (*x* вказується в радіанах);

math.atan(x**)** - арктангенс *x* (*x* вказується в радіанах);

math.cos(x**)** - косинус *x* (*x* вказується в радіанах);

math.sin(x**)** - синус *x* (*x* вказується в радіанах);

math.tan(x**)** - тангенс *x* (*x* вказується в радіанах);

math.hypot (x, y**)** - обчислює гіпотенузу трикутника з катетами *x* і *y*;

math.degrees (x**)** - конвертує радіани в градуси;

math.radians (x**)** - конвертує градуси в радіани;

math.pi - π = 3,1415926…;

math.e - e = 2,718281…

ЛІТЕРАТУРА

1. Мова програмування [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Мова_програмування

2. Python [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Python

3. Тип даних [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Тип_даних

4. Модуль math [Електронний ресурс]. – Режим доступу: https://pythonworld.ru/moduli/modul-math.html

5. Логіка [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Логіка

6. Інтерфейс [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Інтерфейс

7. Курс по библиотеке Tkinter языка Python [Електронний ресурс]. – Режим доступу: https://ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python

8. Подія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Подія

9. Метод (програмування) [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Метод_(програмування)

10. Інформатика: підруч. для 8-го кл. загальноосвіт. І-74 навч. закл. / Й.Я. Ривкінд [та ін.]. – Київ: Генеза, 2016. – 288 с.: іл.

КОЗОЛУП Євгеній Вікторович

Програмування в школі. Мова Python

Навчальний посібник

8 клас

Рецензенти:

Бабенко І.В. – вчитель інформатики Сумської спеціалізованої школи І-ІІІ ступенів №17, м. Суми, Сумської області, спеціаліст І категорії.

Мироненко І.О. – вчитель інформатики Сумської спеціалізованої школи І-ІІІ ступенів №17, м. Суми, Сумської області, спеціаліст ІІ категорії.

ДЛЯ ЗАМІТОК