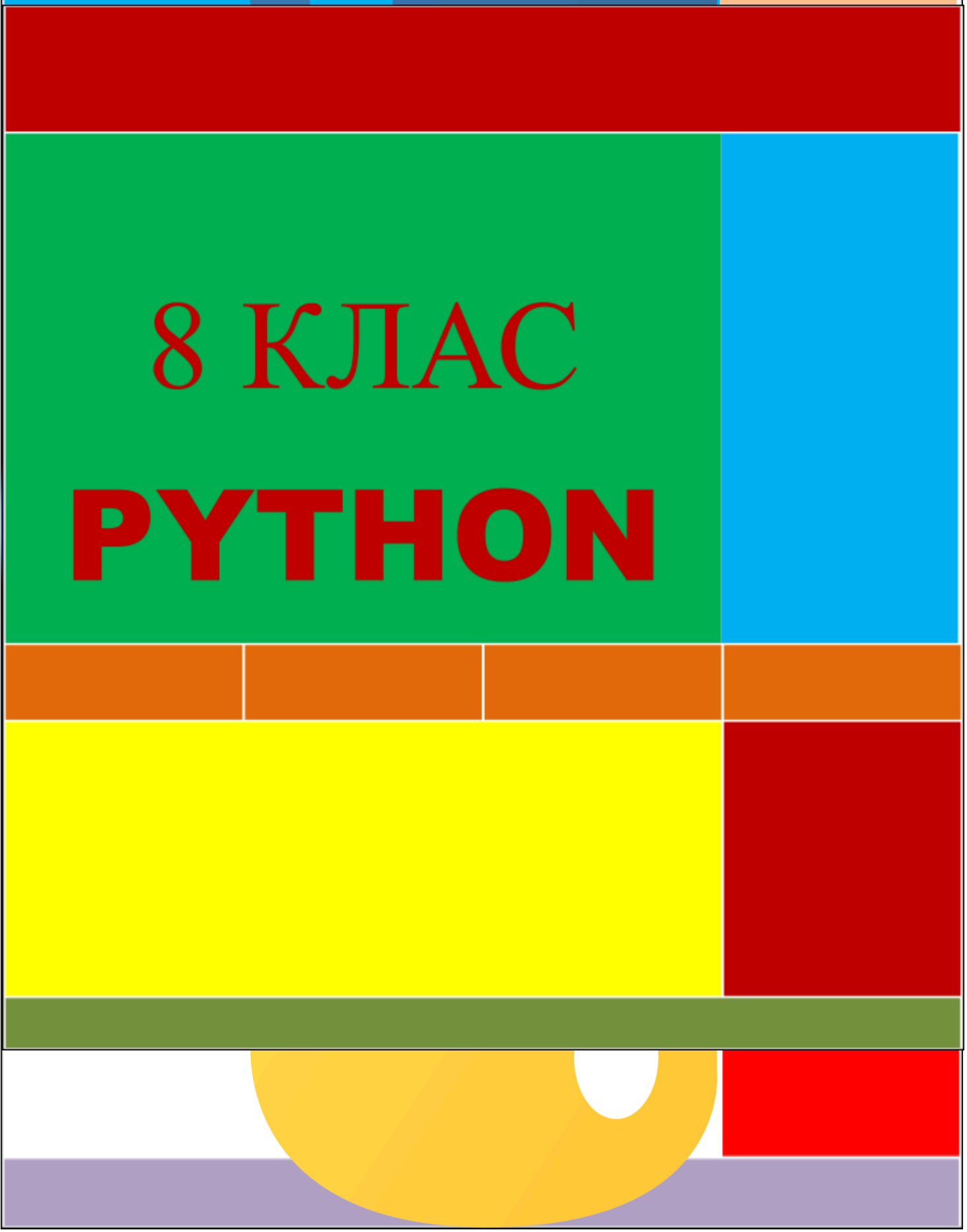




8 КЛАС

PYTHON



§ 1. Сучасні мови та середовища програмування

Комп'ютерна програма — це алгоритм опрацювання даних, записаний спеціальною мовою та призначений для виконання комп'ютером.

У процесі своєї роботи програма опрацьовує **дані**.

Дані, які потрапляють до програми від певних пристроїв (наприклад, з клавіатури або від датчика температури) або від іншої програми, або з іншого джерела (наприклад, з текстового файлу), називають **вхідними (початковими) даними**.

Під час виконання програми утворюються і опрацьовуються й інші дані, які називають **проміжними даними**.

Якщо метою виконання програми є отримання певних даних, то ці дані називають **вихідними (результуючими) даними**.

Мова програмування - це мова, яку використовують для запису алгоритмів, призначених для виконання комп'ютером.

Мова програмування, як і будь-яка інша мова, має такі складові:

- 1) *набір символів, з яких утворюють слова та речення цієї мови - **алфавіт**;*
- 2) *сукупність спеціальних слів, які мають однозначне пояснення та застосування – **словник**;*
- 3) *систему правил складання базових конструкцій мови – **синтаксис**;*
- 4) *правила **семантики**, що пояснюють, яке смислове значення має опис кожної з вказівок програми і які дії повинен виконати комп'ютер під час виконання кожної з таких вказівок.*

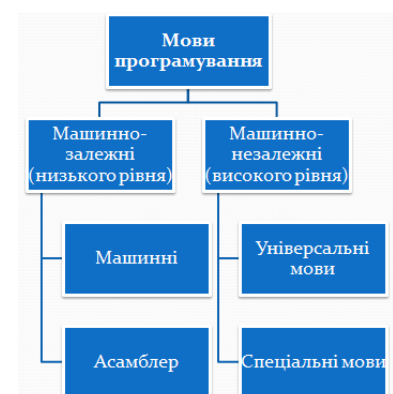
Використання символів, що не входять до алфавіту мови програмування, неправильне написання словникових слів, порушення синтаксичних правил призводять до неможливості виконання комп'ютером відповідної команди. Такі порушення називають **синтаксичними помилками**.

Чим відрізняються мови програмування?

Мови програмування можна поділити на дві групи: мови *низького рівня* та мови *високого рівня*.

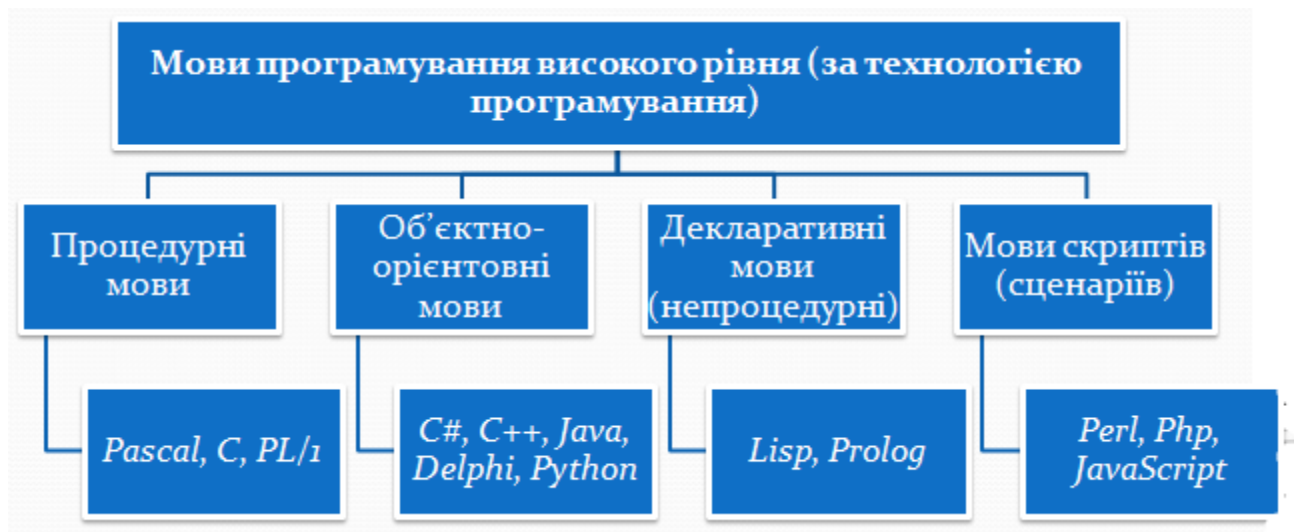
Машинна мова — набір команд, які виконуються безпосередньо центральним процесором. Працювати людині-програмісту з такою мовою важко через велику кількість команд у двійковій формі.

До мов низького рівня належать **мова асамблер**. У мові асамблера виконуються символічні позначення команд, які легше зрозуміти й запам'ятати.



Програма, записана мовою програмування **високого рівня**, містить команди, що схожі на звичайні слова, записані англійською мовою. *Наприклад, програма, записана мовою програмування Python, містить команди print – друкувати, while – поки, if – якщо тощо.*

За технологією програмування розрізняють:



- ❑ **процедурні** мови – програма, складається із набору підзадач (процедур), які реалізують завдання;
- ❑ **об'єктно-орієнтовні** – головними елементами алгоритму є *клас* – новий тип даних, що розширює мову, й *об'єкт*, що має властивості та методи – дії, які виконуються об'єктом;
- ❑ **декларативні** – мови створення програм зі штучним інтелектом;
- ❑ **мова скриптів** (мова сценаріїв) – мова, розроблена для запису «сценаріїв» – послідовностей команд, які користувач може виконувати на комп'ютері.

Які засоби необхідні для виконання створених програм?

Транслятор – програма, яка записує команди програми, описаної деякою мовою програмування, машинною мовою

Компілятор зчитує одразу всю програму й переписує її машинним кодом або мовою асамблер. Процес трансляції, який при такому підході називають *компіляцією*, відбувається один раз, і результат перекладу зберігається в окремому файлі.

Інтерпретатор зчитує вихідний код програми по одній команді й одразу намагається їх «перекладати» та виконувати. Це дає змогу програмісту швидше перевіряти правильність виконання програми та знаходити помилки в коді.

Середовище програмування – це комплекс програм, що містить засоби автоматизації процесів підготовки та виконання програм користувача, а саме:

- 1) **редактор текстів програм** – у ньому можна створювати й редагувати текст програми;
- 2) **довідково-інформаційну систему** про мову програмування та середовище;
- 3) **бібліотеки**, у яких зберігаються найбільш уживані фрагменти програм або цілі програми;
- 4) **компілятор чи інтерпретатор**, використання якого дає змогу швидко знайти в програмі помилку та виправити її;
- 5) **покроковий «виконавець»** програми.

Для написання програм використовують **текстові редактори** або **інтегровані середовища розробки**, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python

- **IDLE** — стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакетом для користувачів Linux.
- **Notepad++** - безкоштовний текстовий редактор вихідного коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.
- **PyScripter** - інтегроване середовище розробки для мови програмування Python, працює під Windows. Поширюється безкоштовно.
- **Wing IDE 101** - вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і Mac OS X. Поширюється безкоштовно.
- **Geany** - вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Mac OS X і Windows.
- **Sublime Text 3** - кросплатформенний текстовий редактор вихідних текстів програм та інтегроване середовище розробки. Підтримує плагіни, розроблені за допомогою мови програмування Python. Sublime Text не є вільним чи відкритим програмним забезпеченням, але деякі його плагіни розповсюджуються з вільною ліцензією, розробляються і підтримуються спільнотою розробників. Проте, можна використовувати вільно, хоча часто з'являється повідомлення про придбання ліцензії. Для користувачів Windows, Mac OS X, Linux.
- **PyCharm** — інтегроване середовище розробки для мови програмування Python. Підтримує веб-розробку на Django. PyCharm є власницьким програмним забезпеченням. Присутні безкоштовна версія **Community** з усіченим набором можливостей і безкоштовна версія **Edu** для навчальних закладів. PyCharm працює під операційними системами Windows, Mac OS X і Linux.

Завдання для самостійного виконання

1. Дайте відповіді на запитання.

- 1) Що таке мова програмування? Чим вона відрізняється від мови людського спілкування?
- 2) Чим відрізняються мови програмування низького та високого рівнів?
- 3) Назвіть приклади мов програмування високого рівня. Для яких завдань їх використовують?
- 4) Для чого використовують програми-транслятори та якими вони бувають?
- 5) Що таке середовище програмування та які його складові?

2. Заповніть таблицю.

Завдання 3.1. Наведіть приклади мов і систем програмування та заповніть таблицю.	Мови програмування		Інтегровані середовища програмування	Системи візуального програмування
	низького рівня	високого рівня		

§ 2. Типи даних у програмуванні

Можна помітити, що все, що ми робимо, ми робимо над чимось - якимись предметами або об'єктами. Ми міняємо властивості об'єктів та їх можливості. Програми для комп'ютерів також маніпулюють якимись об'єктами (назовемо їх поки даними).

Очевидно, дані бувають різними. Дуже часто комп'ютерній програмі доводиться працювати з числами і рядками. Числа у свою чергу також бувають різними: цілими, дробовими, можуть мати величезне значення або дуже довгу дробову частину.

***Тип даних** – це набір значень та операцій, які над ними можна виконувати.*

Наприклад, числа можна множити бо ділити. Але ці операції неможливо виконати над словами чи літерами.

Змінні можуть набувати значень різних типів. Розглянемо деякі стандартні типи мови Python.

Назва типу	Позначення у Python	Приклади значень
цілий	int	5 1225 -4587
дробовий	float	3.45 -0.145 9.0
рядковий	str (string)	'mother' "Python" 'In 1985 year'
логічний	bool	Лише 2 значення: True False

Ви вже знаєте, що в Python для присвоювання змінній певного значення використовується символ «=». У багатьох мовах програмування, включаючи Python, цей символ використовується для позначення «присвоювання» .

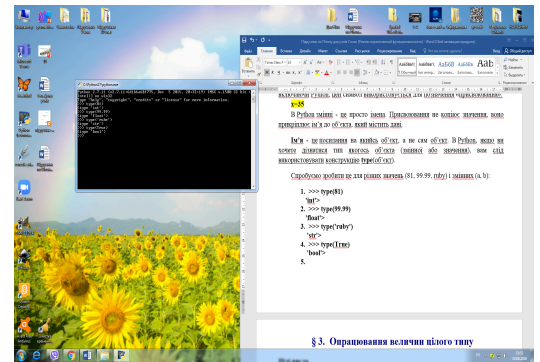
x=35

В Python змінні - це просто імена. Присвоювання не копіює значення, воно прикріплює ім'я до об'єкта, який містить дані.

Ім'я - це посилання на якийсь об'єкт, а не сам об'єкт. В Python, якщо ви хочете дізнатися тип якогось об'єкта (змінної або значення), вам слід використовувати конструкцію **type(об'єкт)**.

Спробуємо зробити це для різних значень (81, 99.99, ruby) і змінних (a, b):

1. `>>> type(81)`
`'int'>`
2. `>>> type(99.99)`
`'float'>`
3. `>>> type('ruby')`
`'str'>`
4. `>>> type(True)`
`'bool'>`



Операції

Можна сказати, що **операція** - це виконання якихось дій над даними (операндами). Для виконання конкретних дій потрібні спеціальні інструменти — оператори.



Наприклад, символ "+" по відношенню до чисел виконує операцію додавання, а по відношенню до рядків - конкатенацію (з'єднання). Парний знак ** зводить перше число в ступінь другого.

Вираз	Значення
<code>34.907 + 320.65</code>	<code>355.55699999999996</code>
<code>"Hi, " + "world :)"</code>	<code>'Hi, world :)'</code>
<code>"Hi, " * 10</code>	<code>'Hi, Hi, Hi, Hi, Hi, Hi, Hi, Hi, Hi, Hi, '</code>

Зміна типу даних

Що буде, якщо ми спробуємо виконати в одному виразі операцію над різними типами даними? Наприклад, скласти ціле і дробове число, число і рядок. Однозначну відповідь дати не можна: так, при складанні цілого числа і числа з плаваючою крапкою, виходить число з плаваючою крапкою, а якщо спробувати скласти будь-яке число і рядок, то інтерпретатор Python видасть помилку.

Вираз	Результат виконання
<code>1 + 0.65</code>	<code>1.65</code>
<code>"Hi," + 15</code>	П о м и л к а

Однак, бувають випадки, коли програма отримує дані у вигляді рядків, а оперувати повинна числами (або навпаки). У такому випадку використовуються спеціальні функції (особливі оператори), що дозволяють перетворити один тип даних в інший.

Так функція **int ()** перетворює переданий їй рядок (або число з плаваючою крапкою) в ціле число, функція **str ()** перетворює переданий їй аргумент в рядок, **float ()** - в дробове число.

Вираз	Результат виконання
int ("56")	56
int (4.03)	4
int ("comp 486")	П о м и л к а
str (56)	'56 '
str (4.03)	'4.03 '
float (56)	56.0
float ("56")	56.0

Змінні

Дані зберігаються в комірках пам'яті комп'ютера. Коли ми вводимо число, воно поміщається в пам'ять. Але як дізнатися, куди саме? Як надалі звертатися до цих даних? Раніше, при написанні програм на машинній мові, звернення до комірок пам'яті здійснювали за допомогою вказівки регістрів. Але вже з появою асемблерів, при зверненні до даних стали використовувати так звані змінні. Механізм зв'язку між змінними та даними може розрізнятися в залежності від мови програмування і типу даних. Поки досить запам'ятати, що дані зв'язуються з яких-небудь ім'ям і надалі звернення до них можливо з цього імені.

У програмі на мові Python зв'язок між даними і змінними встановлюється за допомогою знака **=**. Така операція називається **присвоєнням**. Наприклад, вираз **a = 4** означає, що на об'єкт (дані) у певній області пам'яті посилається ім'я **a** і звертатися до них тепер слід за цим іменем.



Імена змінних можуть бути будь-якими. Однак є кілька загальних правил їх написання:

- Бажано давати змінним осмислені імена, що говорять про призначення даних, на які вони посилаються.
- Ім'я змінної не повинно збігатися з командами мови (зарезервованими ключовими словами).

- Ім'я змінної має починатися з букви або символу підкреслення (_).

Щоб дізнатися значення, на яке посилається змінна, перебуваючи в режимі інтерпретатора, достатньо її викликати (написати ім'я і натиснути Enter).

Приклад роботи зі змінними в інтерактивному режимі:

```
>>> apples = 100
>>> eat_day = 5
>>> day = 7
>>> apples = apples - eat_day * day
>>> apples
65
```

Завдання для самостійного виконання

1. Заповнити таблицю в зошиті

Практичне завдання				
Заповніть в зошиті таблицю без використання IDLE				
Вхідні дані (a)	Типи даних (a)	Результат (b)	Типи даних (b)	Конструкція перетворення
a='45.56'		b=45		
a=78*33/15	int	b='652'		
a=(78-89)*56			str	
a=78		b=78.0		b=float(a)

2. Практична робота

- Змінній **var_int** надайте значення 10, **var_float** - значення 8.4, **var_str** - "No".
- Змініть значення, збережене у змінній **var_int**, збільшивши його в 3.5 рази, результат зв'яжіть зі змінною **big_int**.
- Змініть значення, збережене у змінній **var_float**, зменшивши його на одиницю, результат зв'яжіть з тією ж змінною.
- Розділіть **var_int** на **var_float**, а потім **big_int** на **var_float**. Результат даних виразів не прив'язуйте ні до яких змінних.
- Змініть значення змінної **var_str** на "NoNoYesYesYes". При формуванні нового значення використовуйте операції конкатенації (+) і повторення рядка (*).
- Виведіть значення всіх змінних.

§ 3. Структура програми. Введення й виведення даних

Структура програми

Розглянемо, з чого складаються програми на мові Python. Будь-яка програма на мові Python складається з модулів. Модуль на мові Python являє собою ряд пов'язаних між собою операцій. Модулі зберігаються в окремих файлах з розширенням *.py. Збережені модулі можна використовувати в створюваних програмах.

Модулі, в свою чергу, складаються з більш простих структурних одиниць. У модулях міститься код на мові Python, що складається з інструкцій. **Інструкції** - це вказівки комп'ютера. Вони визначають, які операції виконає комп'ютер з даними. Інструкції в мові Python поділяються на прості і складові. **Прості** інструкції описуються одним рядком коду, **складові** ж - містять вкладені інструкції.

Інструкції можуть містити вирази. І якщо інструкції визначають, які дії будуть виконані над інформацією, то вирази в складі інструкцій визначають, над якими саме даними будуть виконані дії, описані в інструкції.

Разом з мовою Python поставляється безліч стандартних модулів, які надають програмісту велику кількість інструментів і можливостей для написання самих різних програм.

Введення та виведення даних

В Python існує вбудована функція *input()*, призначена для отримання даних від користувача. Вона отримує від користувача рядок тексту. Зазвичай дані просто вводяться з клавіатури.

Ось приклад застосування функції введення:

```
yourName = input()
```

Ця команда дає користувачеві можливість ввести рядок, якому буде присвоєно ім'я *yourName*. Вставимо її в програму:

```
print("Як тебе звати?")  
yourName = input()  
print("Привіт, ", yourName)
```

Якщо зберегти і запустити цю програму на виконання, отримаємо приблизно такий результат:

```
Як тебе звати?  
Тарас  
Привіт, Тарас
```

Тут було використано дві функції – функція *print()* для виведення на екран пояснюючого тексту, а функція *input()* – для зчитування з клавіатури введеної

користувачем відповіді.

Для виведення даних використовується функція **print()**, яка може виводити не тільки значення змінних, але і значення будь-яких виразів. Наприклад:

```
>>> print(2+8/23-45*2+12**3)
1640.3478260869565
```

Також за допомогою функції **print** можна виводити значення не одного, а декількох виразів, для цього потрібно перерахувати їх через кому:

```
>>> print(14+8,12+36, 8*9)
22 48 72
```

Використовуючи функцію **print()** можна виводити на екран будь які скомпоновані рядки тексту:

```
>>> a = 15
>>> b = 23
>>> print(a, '+', b, '=', a+b)
15 + 23 = 38
```

За замовчуванням, замість коми між значеннями, які виводяться, вставляється пробіл. Це можна легко змінити використавши додатковий параметр **sep** функції **print()**:

```
>>> print(1, 2, 3, 4, 5, 6, sep='-')
1-2-3-4-5-6
```

Фактично, атрибут **sep** замінить стандартний пробіл на вказаний вами символ (або послідовність символів, але не забуваємо про використання лапок).

А якщо спробувати вивести кожне значення з нової стрічки. Хіба для цього потрібно буде писати щоразу нову функцію **print()** для кожного значення? Звичайно Ні. Достатньо в якості атрибуту **sep** вказати символ перенесення рядка “**\n**”:

```
>>> print(1, 2, 3, 4, 5, 6, sep='\n')
1
2
3
4
5
6
```

Кожна наступна функція **print()** буде виводити задане значення з нового рядка. Фактично **print()** виводить вказане значення а після цього переводить курсор на наступний рядок.

Введення чисел

Отже, тепер ви знаєте, як за допомогою функції *input ()* здійснити введення рядка. А що робити, якщо вам потрібно ввести число? Можна взяти функцію *int ()* або *float ()* і перетворити наданий вам функцією *input ()* рядок в число. Це буде виглядати так:

```
a = input()
a = float(a)
```

Спочатку за допомогою функції *input ()* ми отримуємо дані у вигляді рядка. А потім, застосувавши функцію *float ()*, ми перетворюємо її в число. Втім, все це можна зробити і коротше, всього за один крок:

```
a = float(input())
```

Цей рядок виконує ті ж самі дії. Отримує у користувача рядок і перетворює його в число. Але тепер для цього потрібно менше коду.

Якщо ви хочете, щоб всі дані, які вводяться користувачем були цілими числами (без десяткових знаків), для перетворення можна задіяти функцію *int ()*:

```
a = int(input())
```

Але тут може виникнути помилка введення, коли користувач замість цілого числа введе дробове:

```
>>> a = int(input())
```

```
12.3
```

```
Traceback (most recent call last):
```

```
File "<pyshell#30>", line 1, in <module>
```

```
a = int(input())
```

```
ValueError: invalid literal for int() with base 10: '12.3'
```

Це сталося тому, що **Python** не знає як перетворити рядок з десятковою крапкою в ціле число. Вирішити дану проблему, можна спочатку перетворивши рядок в дійсне число, а вже потім в ціле, тобто:

```
a = float(input())
a = int(a)
```

Ту ж саму операцію ми можемо виконати в одному рядку коду:

```
>>> a = int(float(input()))
```

```
12.365
```

```
>>> a
```

```
12
```

Тепер помилка відсутня, оскільки, ми спочатку виконали перетворення в дійсне

число, а вже потім – в ціле.

Використовуючи отриманні знання, напишемо невелику програму-калькулятор, яка виведе на екран результат виконання семи основних операцій над цілими числами. Числа запитуються в користувача з відповідним повідомленням.

Приклад програми:

```
a = int(float(input("Введіть перше ціле число:\n")))\nb = int(float(input("Введіть друге ціле число:\n")))\nprint(a, '+', b, '=', a+b)\nprint(a, '-', b, '=', a-b)\nprint(a, '*', b, '=', a*b)\nprint(a, '/', b, '=', a/b)\nprint(a, '**', b, '=', a**b)\nprint(a, '//', b, '=', a//b)\nprint(a, '%', b, '=', a%b)
```

Результат виконання:

Введіть перше ціле число:

12

Введіть друге ціле число:

32

12 + 32 = 44

12 - 32 = -20

*12 * 32 = 384*

12 / 32 = 0.375

*12 ** 32 = 34182189187166852111368841966125056*

12 // 32 = 0

12 % 32 = 12

Завдання для самостійного виконання

1. Створіть в інтерактивному режимі дві змінні - одну для вашого імені, а другу для вашого прізвища. Потім за допомогою однієї інструкції *print* виведіть їх на екран в один рядок.
2. Напишіть програму, яка спочатку запитує ваше ім'я, потім ваше прізвище, а потім виводить на екран повідомлення з вашими особистими даними.
3. Напишіть програму, яка запитує розміри прямокутної кімнати (в сантиметрах) і виводить на екран площу килима, який закриває всю підлогу.
4. Напишіть програму, яка робить все перераховане в завданні 3, запитуючи вартість килима за квадратний метр. Після виведіть на екран наступні дані:
 - загальну площу килима в квадратних сантиметрах;
 - загальну площу килима в квадратних метрах (1 квадратний метр = 10 000 квадратних сантиметрів);
 - підсумкову ціну килима.
5. Напишіть програму для підрахунку дрібних грошей. Вона повинна питати:
 - «Скільки у вас монет по 1 гривні?»;
 - «Скільки у вас монет по 2 гривні?»;
 - «Скільки у вас монет по 5 гривень?».
 - Після цього на екрані повинна з'явитися загальна сума.
6. Запишіть букву 'A' (латинську, велику) 100 разів поспіль.
7. Запишіть слово 'Python' 100 раз поспіль.
8. Дано: рядок `s = 'інформатика'`. Програма має створити новий рядок `ns` зі значенням 'торт' присвоюванням необхідних зрізів рядка `s`.
9. Дано рядок `S`. Програма має створити новий рядок, в якому міститимуться подвоєні символи першого рядка. Наприклад: вхідний рядок = 123, вихідний = 112233.
10. Напишіть програму, яка вітає користувача, виводячи слово Hello, введене ім'я і розділові знаки за зразком (Hello, Harry!). Програма повинна зчитувати в текстову змінну значення і писати відповідне вітання. Зверніть увагу, що після коми повинен обов'язково стояти пробіл, а перед знаком оклику пробілу немає.

§ 4. Вирази. Складання програм на опрацювання виразів

Числові та буквені вирази

Вирази, що містять числа і знаки дій, називають **числовими**.

Приклад:

Вартість одного хліба — 5 грн., а вартість плитки шоколаду — 15 грн.

Вартість двох хлібів і двох плиток шоколаду разом $2 \cdot 5 + 2 \cdot 15$ грн.

Вираз $2 \cdot 5 + 2 \cdot 15$ — числовий.

Виконавши в числовому виразі названі дії, знайдемо його значення.

Змінимо умову попереднього прикладу.

Приклад:

Нехай вартість одного хліба — x грн., а вартість плитки шоколаду — y грн.

Вартість трьох хлібів і двох плиток шоколаду разом $3 \cdot x + 2 \cdot y$ грн.

Вирази, що містять не тільки числа і знаки дій, але і букви, називають **буквеними**.

Вираз $3 \cdot x + 2 \cdot y$ — буквений.

Числові і буквені вирази, утворені з яких-небудь даних, - це переклад звичайної мови на математичну — мову цифр, знаків дій та інших символів.

Приклад:

Число m на 8 більше числа n математичною мовою запишеться так: $m = n + 8$.

Складання програм на опрацювання виразів

Мова Python має можливість розв'язувати математичні задачі. Обчислення здійснюються за допомогою арифметичних виразів. Арифметичні вирази будуються з імен змінних, констант, математичних операторів (знаків арифметичних операцій) і круглих дужок так, як це прийнято в математиці.

Розглянемо математичні операції і відповідні їм символи операторів:

Операція	Символ оператора	Приклад	Результат
Додавання	+	$Res = 15 + 3$	$Res = 18$
Віднімання	-	$A = Res - 10$	$A = 8$
Множення	*	$A = A * 2$	$A = 16$
Ділення	/	$Res = 5 / 2$	$Res = 2.5$
Обчислення неповної частки від ділення	//	$Res = 5 // 2$	$Res = 2$
Обчислення остачі	%	$Res = 5 \% 2$	$Res = 1$
Піднесення до степеня	**	$A = 4 ** 2$	$A = 16$

Порядок виконання (пріоритет) математичних операцій:

- 1) піднесення до степеня (**);
- 2) множення (*) і ділення (/), цілочисельне ділення (//), одержання остачі від цілочисельного ділення (%);
- 3) додавання (+) і віднімання (-).

Зверніть увагу!

Якщо операції мають однаковий пріоритет, то вони виконуються зліва направо по черзі.

Правила лінійного запису арифметичних виразів

1. Вираз повинен бути записаний у вигляді лінійного ланцюжка символів.
2. Не можна опускати знак операції множення.
3. Порядок виконання операцій одного пріоритету регулюється дужками.
4. Аргументи функцій записуються в круглих дужках.

Запишемо арифметичний вираз за правилами лінійного запису.

Вираз виглядатиме так:

$$(2*x - 5) / (3+x)$$

Приклад:

*Першого дня туристи подолали **a** км маршруту, другого дня – вдвічі більше. Третього дня вони пройшли на **b** км менше, ніж першого. Скільки кілометрів подолали туристи за три дні??*

Математичний розв'язок:

1. Скільки кілометрів подолали туристи другого дня? **2a**
2. Скільки кілометрів подолали туристи третього дня? **a – b**
3. Скільки кілометрів подолали туристи всього? **a+2a+a – b**
4. Спростимо вираз: **a+2a+a – b=4a – b**

Програма буде мати вигляд:

```
a=input ()  
b= input ()  
L=5*a-b  
print L
```

Завдання для самостійного виконання

1. Дано довжину ребра куба a . Написати програму, яка обчислює об'єм куба $V=a^3$ і площу його поверхні $S=6\cdot a^2$.
2. Дано довжини ребр a , b , c прямокутного паралелепіпеда. Написати програму, яка обчислює його об'єм $V=a\cdot b\cdot c$ і площу поверхні $S=2\cdot(a\cdot b+b\cdot c+a\cdot c)$.
3. Дано два числа a і b . Написати програму, яка обчислює їх середнє арифметичне: $(a + b)/2$.
4. У хлопчика було x книг, а у дівчинки - y . Всі книги діти поставили на 2 полиці, порівну на кожную. Скільки книг на кожній полиці?
5. У Андрійка було s марок, а у Тараса - у 5 разів більше. Хлопці всі марки розклали у 3 альбоми, порівну у кожний. Скільки марок у кожному альбомі?
6. За перший день велосипедист проїхав a км, за другий - b км, а за третій день - на 15 км менше, ніж за перший і другий день разом. Скільки кілометрів проїхав велосипедист за третій день?
7. З пристані Київ до пристані Кременчук вийшов теплохід, і одночасно йому назустріч з пристані Кременчук вийшов катер. Теплохід ішов зі швидкістю a км/год, а катер — b км/год. Через c год вони зустрілися. Яка відстань між пристанями?
8. Два літаки одночасно вилетіли з аеродрому в протилежних напрямках. Через d годин після вильоту відстань між ними була S км. Перший літак летів зі швидкістю a км/год.. З якою швидкістю летів другий літак?
9. Відстань між двома пристанями по річці S км. За який час катер пройде цю відстань туди й назад, якщо швидкість катера в стоячій воді a км/год, а швидкість течії річки b км/год?
10. Ширина саду a м, а довжина у 2 рази більша. Третину площі саду займають фруктові дерева, а решту — ягідні кущі. Яку площу займають ягідні кущі?

§ 5. Логічні вирази та змінні. Операції над ними

Часто в реальному житті ми погоджуємося або заперечуємо те чи інше твердження, подію, факт. Наприклад, "Сума чисел 3 та 5 більша за 7" є правдивим твердженням, а "Сума чисел 3 та 5 менша за 7" - хибним. Можна помітити, що з точки зору логіки подібні фрази припускають тільки два результати: "Так" (правда) і "Ні" (неправда). Подібне використовується в програмуванні: якщо результатом обчислення виразу може бути лише "Так" або "Ні", то такий вираз називається **логічним**.

На попередніх уроках були описані три типи даних: цілий, дробовий і рядковий. Також виділяють логічний тип даних. У цього типу всього два можливих значення: **True** (правда) - 1 і **False** (неправда) - 0. Тільки ці значення можуть бути результатом логічних виразів.

Логічні оператори

Говорячи природною мовою (наприклад, українською) ми позначаємо порівняння словами "дорівнює", "більше", "менше". У мовах програмування використовуються спеціальні знаки, подібні тим, які використовуються в математичних виразах: > (більше), < (менше), >= (більше або дорівнює), <= (менше або дорівнює). Новими для вас можуть виявитися позначення рівності: == (два знаки "дорівнює"); а також нерівності != (знак оклику і знак "дорівнює"). Тут слід звернути увагу на наступне: не плутайте операцію присвоювання, що позначається у мові Python одиночним знаком "дорівнює", і операцію порівняння (два знаки "дорівнює"). Присвоєння і порівняння - зовсім різні операції.

Операції над логічними виразами

Над об'єктами типу "**bool**" можна виконувати наступні операції:

Операція	Результат
$a == b$	True якщо значення a дорівнює значенню b, False в інших випадках
$a != b$	True якщо значення a не дорівнює значенню b, False в інших випадках
$a > b$	True якщо значення a більше ніж значенню b, False в інших випадках
$a >= b$	True якщо значення a більше ніж значенню b або дорівнює йому, False в інших випадках
$a < b$	True якщо значення a менше ніж значенню b, False в інших випадках
$a <= b$	True якщо значення a менше ніж значенню b або дорівнює йому, False в інших випадках

Приклади роботи з логічними виразами на мові програмування Python

```
x == 4      # x дорівнює 4  
x == 7      # x дорівнює 7  
x != 7      # x не дорівнює 7  
x != 4      # x не дорівнює 4  
x > 5       # x більше 5  
x < 5       # x менше 5  
x >= 6      # x більше або дорівнює 6  
x <= 6      # x менше або дорівнює 6
```

Прості логічні операції:

```
>>> 5 == 5  
True  
>>> 5 >= 5  
True  
>>> 5 != 'five'  
True
```


§ 6. Умовні оператори (коротка та повна форма).

Хід виконання програми може бути лінійним, тобто таким, коли вирази виконуються, починаючи з першого і закінчуючи останнім рядком, по порядку, не пропускаючи жодного рядка коду. Але частіше буває зовсім не так. При виконанні програмного коду деякі його ділянки можуть бути пропущені.

Щоб краще зрозуміти чому, проведемо аналогію з реальним життям. Припустимо, людина живе за розкладом (можна сказати, розклад - це своєрідний "програмний код", який слід виконати). У її розкладі о 18.00 запланований похід до басейну. Проте людині надходить інформація, що воду злили, і басейн не працює. Цілком логічно скасувати своє заняття з плавання. Тобто однією з умов відвідування басейну має бути його функціонування, інакше повинні виконуватися інші дії.

Схожа нелінійність дій може бути передбачена і в комп'ютерній програмі. Наприклад, частина коду повинна виконуватися лише при певному значенні конкретної змінної. Такі алгоритми називаються алгоритмами з розгалуженням.

Розгалуження

Згадаємо вивчений раніше теоретичний матеріал про повне та неповне розгалуження.

Для того, щоб представити умову у програмному коді Python, використовуються наступний оператор: **if** (з англ. «якщо»). Він застосовується у вигляді наступної конструкції:

if *a* **логічний оператор** *b* :
операції, що здійснюються при виконанні
умови (тобто у випадку True)

1. Неповне розгалуження

if <логічний вираз>:

дії, які будуть виконуватися якщо логічний вираз буде мати значення **True** *else* (з англ. «інакше»), є необов'язковою частиною конструкції **if**.

Приклад її реалізації на мові програмування Python:

```
if n < 100:  
    c = a ** b
```

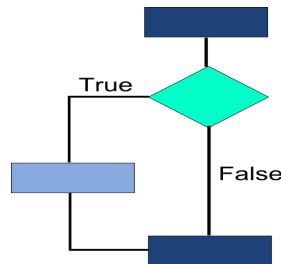
Розшифруємо цей вираз: якщо змінна **n** менша за **100** то виконується дія **c=a**b**

Перший рядок конструкції **if** - це заголовок, в якому перевіряється умова виконання рядків коду після двокрапки (тіла конструкції). У вищевказаному прикладі тіло містить всього лише один вислів, однак частіше їх буває набагато більше. Про Python говорять, що це мова програмування з досить ясним



і легко читаним кодом. Це пов'язано з тим, що в ньому зведені до мінімуму допоміжні елементи (дужки, крапки з комою), а для поділу синтаксичних конструкцій використовуються відступи від початку рядка. Враховуючи це, в конструкції **if** код, який виконується при дотриманні умови, повинен обов'язково мати відступ вправо. Решта коду (основна програма) повинна мати той же відступ, що й слово **if**. Зазвичай відступ робиться за допомогою клавіші Tab.

Можна зобразити блок-схему програми, яка містить інструкцію **if**, в такому вигляді:



Зустрічається і більш складна форма розгалуження: **if-else**. Якщо умова при інструкції **if** виявляється хибною, то виконується блок коду при інструкції **else**.

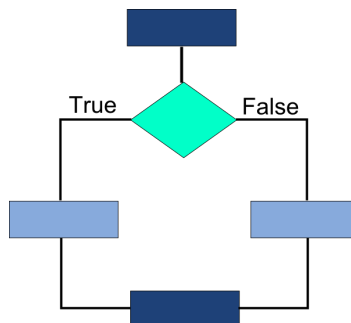
2. Повне розгалуження

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**



Приклад коду з гілкою **else** на мові програмування Python:

```
товар1 = 50  
товар2 = 32  
if товар1 + товар2 > 99:  
    print("Сума не достатня")  
else:  
    print("Чек сплачено")
```

Задача

На одному маленькому квадратному безлюдному острові зі стороною a метрів перебували k Робінзонів. Чи не порушені їх права на житло, якщо на кожного Робінзона повинно припадати n м² площі острова?

```
a=int(input("Сторона острова ="))
s=a*a
k=int(input("Кількість Робінзонів ="))
n=int(input("Норма площі ="))
m=s//k
if (m>=n):
    print ("Місця достатньо")
else:
    print ("Місця не достатньо")
```

Завдання для самостійного виконання

1. Напишіть програмний код, в якому у випадку, якщо значення якоїсь змінної більше 0, виводилося б спеціальне повідомлення (використовуйте функцію print). Один раз виконайте програму при значенні змінної більше 0, другий раз - менше 0.
2. Вдоскональте попередній код за допомогою гілки else так, щоб залежно від значення змінної, виводилося або 1, або -1.
3. Самостійно придумайте програму, в якій би використовувалася інструкція if (бажано з гілкою else). Вкладений код повинен містити не менше трьох виразів.
4. Іван Петрович у нових штанах сів на щойно пофарбовану табуретку. На його штанах з'явилась квадратна пляма з довжиною сторони a см. Виявилось, що в хімчистку беруть одяг, плями на якому не більші n см². Визначити, чи вдалось Іванові Петровичу врятувати свої штани?

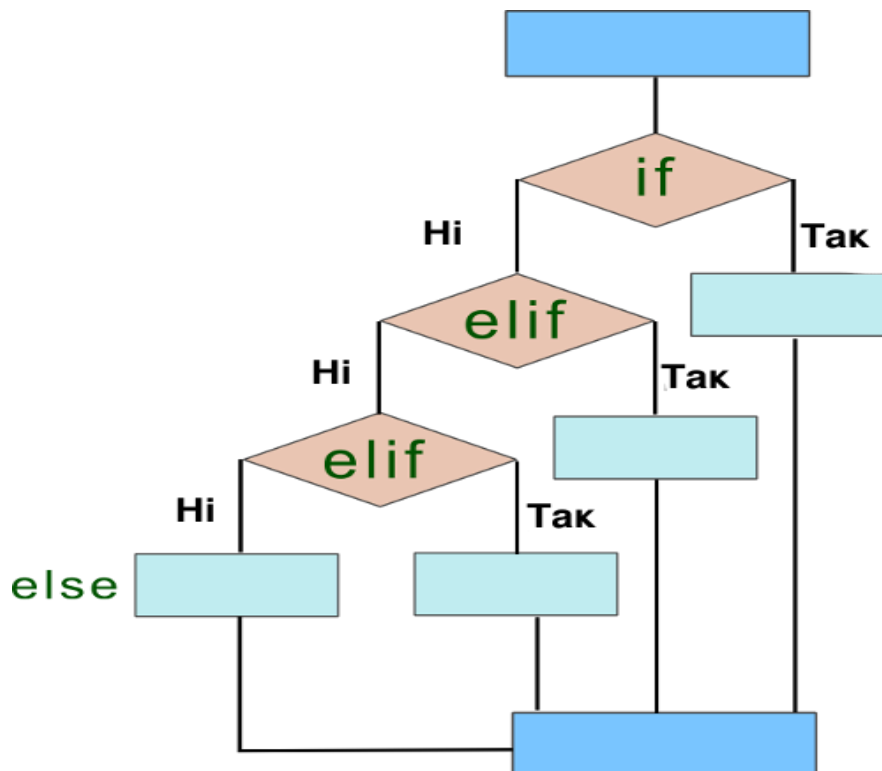
§ 7. Множинне розгалуження

Логіка виконуваної програми може бути складніша, ніж вибір однієї з двох гілок. Наприклад, залежно від значення тієї чи іншої змінної, може виконуватися одна з трьох (або більше) гілок програми.

Як організувати таке множинне розгалуження? Напевно, можна використовувати декілька інструкцій **if**: спочатку перевіряється умовний вираз у першій інструкції **if** (якщо він True, то буде виконуватися вкладений в неї блок коду), потім у другій інструкції **if** і т.д. Однак при такому підході перевірка наступних інструкцій буде продовжуватися навіть тоді, коли перша умова була True, і блок коду при даній гілці був виконаний. Перевірка наступних умов може виявитися безглуздою.

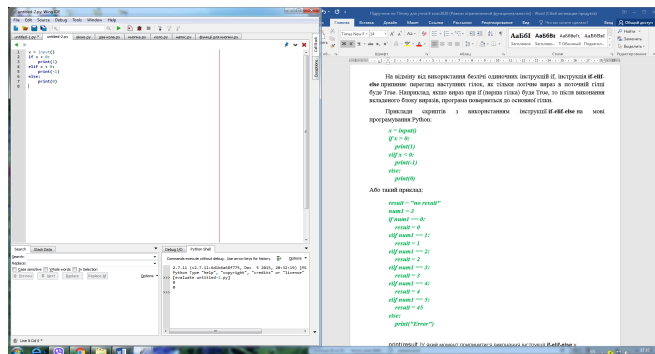
Зазвичай таку проблему можна вирішити за допомогою вкладених конструкцій **if-else**. Однак при цьому часто з'являється проблема правильного трактування коду: незрозуміло, до якого **if** відноситься **else** (хоча в Python така плутанина не можлива через обов'язкові відступи).

З іншого боку, в ряді мов програмування, в тому числі і Python, передбачено спеціальне розширення інструкції **if**, що дозволяє направити потік виконання програми з однієї гілки. Дана розширена інструкція, крім необов'язкової частини **else**, містить ряд гілок **elif** (скорочення від "**else if**" - "ще якщо") і виглядає приблизно так, як показано на блок-схемі. Частин **elif** може бути як завгодно багато (в межах розумного, звичайно).

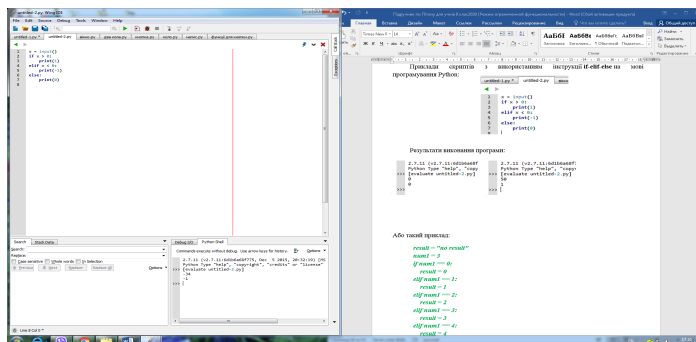
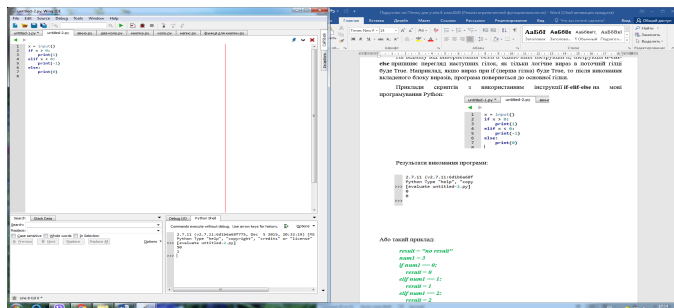
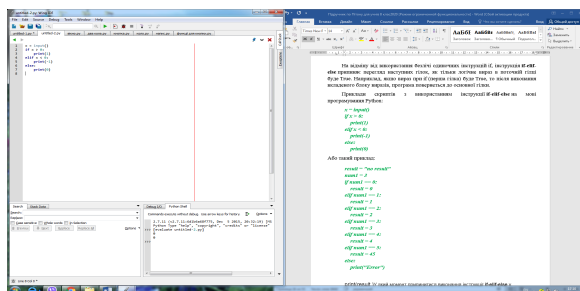


На відміну від використання безлічі одиночних інструкцій `if`, інструкція **`if-elif-else`** припиняє перегляд наступних гілок, як тільки логічне вираз в поточній гілці буде `True`. Наприклад, якщо вираз при `if` (перша гілка) буде `True`, то після виконання вкладеного блоку виразів, програма повернеться до основної гілки.

Приклади скриптів з використанням інструкції **`if-elif-else`** на мові програмування Python:



Результати виконання програми:



Завдання для самостійного виконання

1. Напишіть програму за наступним описом:

- двом змінним присвоюються числові значення;
- якщо значення першої змінної більше другої, то знайти різницю значень змінних (відняти від першої другу), результат присвоїти третій змінній;
- якщо перша змінна має менше значення, ніж друга, то третю змінну пов'язати з результатом суми значень двох перших змінних;
- у всіх інших випадках, присвоїти третій змінній значення першої змінної;
- вивести значення третьої змінної на екран.

2. Придумайте програму, в якій би використовувалася інструкція **if-elif-else**. Кількість гілок повинна бути як мінімум чотири.

3. Скласти програму, яка за введеною кількістю років визначає одну з категорій населення:

- ще не школяр — до 6 років;
- початкова школа — 6–9 років;
- середня школа — 10–15 років;
- старша школа — 16–17 років;
- вже не школяр — понад 17 років.

4. Написати програму, яка в залежності від віку користувача рекомендує один з фільмів:

3-6 років – «Маша і Ведмідь»;

7-12 років – «Поліанна»;

13-16 років – «Загадковий острів»;

16 і старші – «Марсіанин»

§ 8. Складені умови

Логічні вирази типу $x \geq 1023$ є простими. Однак, на практиці нерідко використовуються більш складні. Може знадобитися отримати відповіді "Так" або "Ні" залежно від результату виконання двох простих виразів. Наприклад, "на вулиці йде сніг або дощ", "змінна new більша 12 і менша 20" і т.п.

У таких випадках використовуються спеціальні оператори, що об'єднують два і більше простих логічних вирази.

1) **Оператор AND (з англ. «і»)**. Результат буде **True** або **False**. Цей оператор застосовується наступним чином: [лог.вираз] and [лог.вираз] and ... Такий складений логічний вираз буде набувати значення **True** тільки у тому випадку, якщо всі змінні 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.

Наприклад: $1 > a \text{ and } a > 5$

2) **Оператор OR (з англ. «або»)**. Результатом виразу буде **True** в тому випадку, якщо хоча б один із складових цього виразу буде мати значення **True**.

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 $\text{not } 2 > 1$ буде приймати значення *False* (протилежний до значення виразу $2 > 1$). Іноді не дуже часто використовують цей оператор окремо, так як вираз $\text{not } 2 > 1$ можна замінити на протилежний логічний вираз (в даному випадку $2 \leq 1$). Та все ж його часто застосовують із іншими операторами. Для прикладу вираз $\text{not } 1 > 2 \text{ and } 4 > 3$ буде мати значення *True* ($\text{not } 1 > 2 = \text{True}$, $4 > 3 = \text{True}$).

Випадки, всі логічні оператори можна об'єднувати у різні вирази. Наприклад: $\text{not } 2 > 1 \text{ or } (3 > 2 \text{ and } 4 > 3)$.

3) **Оператор NOT (з англ. «не»)**. Застосовується трішки по іншому: він ставиться на початку, тобто перед логічним виразом, надаючи йому при цьому протилежного значення. Тобто, вираз $\text{not } 2 > 1$ буде приймати значення **False** (протилежний до значення виразу $2 > 1$).

Наприклад: $\text{Not } 1 > 2 \text{ and } 4 > 3$ буде мати значення *true* ($\text{not } 1 > 2 = \text{True}$, $4 > 3 = \text{True}$).

Приклади роботи зі складними логічними виразами на мові Python

$x = 8$

$y = 13$

$x == 8 \text{ and } y < 15$

x дорівнює 8 та y менше 15

$x > 8 \text{ and } y < 15$

x більше 8 та y менше 15

$x != 0 \text{ or } y > 15$

x не дорівнює 0 або y більше 15

$x < 0 \text{ or } y > 15$

x менше 0 або y більше 15

Операції над логічними виразами

Операція	Результат
<i>a and b</i>	True якщо обидва a і b дорівнюють True, False в інших випадках
<i>a or b</i>	True якщо хоча б один з a і b дорівнює True, False в інших випадках
<i>not b</i>	True якщо b дорівнює False, False в інших випадках
<i>a is b</i>	True якщо a і b посилаються на один об'єкт, False в інших випадках
<i>a is not b</i>	True якщо a і b посилаються на різні об'єкти, False в інших випадках
<i>a in b</i>	True якщо послідовність b включає у себе об'єкт a, False в інших випадках
<i>a not in b</i>	True якщо послідовність b не включає у себе об'єкт a, False в інших випадках

Складні логічні операції:

1. Операція **not** (логічне "не") має вищий пріоритет над **and**.
2. Операція **and** (логічне "і") має вищий пріоритет над **or** (логічним "або").

>>> not False or True and False

True

3. Оператори **"is"** та **"is not"** перевіряють не по значенню, а на приналежність до одного і того ж об'єкту:

>>> a = [1, 2, 3]

>>> b = [1, 2, 3]

>>> a == b

True

>>> a is b

False

>>> b = a

>>> a is b

True

4. За допомогою **"in"** можна перевірити чи входить об'єкт до послідовності:

>>> a = ['one', 2, 'three', 4]

>>> 2 in a

True

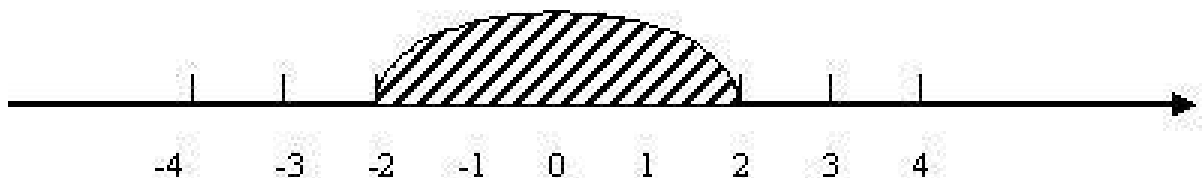
>>> 1 in a

False

Завдання для самостійного виконання

1. Присвойте двум змінним будь-які числові значення.
2. Складіть чотири складних логічних вирази за допомогою оператора `and`, два з яких повинні давати `True`, а два інших - `False`.
3. Аналогічно виконайте п. 2, але вже використовуючи оператор `or`.
4. Спробуйте використовувати у складних логічних виразах роботу зі змінними рядкового типу.
5. Дано: ціле число x . Код програми має перевірити чи належить число x діапазону $(-5, 3)$.
6. Дано три цілих числа a, b, c . Знайти найбільше серед них.
7. Сторони трикутника a, b, c – вводяться з клавіатури.
Програма має визначити тип трикутника: рівносторонній, рівнобедрений, прямокутний, це звичайний трикутник, або взагалі не трикутник.
8. Дано: точка з координатою x на числовій прямій.

Програма має визначити: чи належить вона заштрихованій області.



§ 9. Цикли. Оператор циклу *while*

Цикли - це інструкції, що виконують одну і ту ж саму послідовність дій, поки діє задана умова.

У реальному житті ми досить часто стикаємося з циклами. Наприклад, ходьба людини - цілком циклічне явище: крок лівою, крок правою, знову лівою-правою і т.д., поки не буде досягнута певна мета (наприклад, школа чи магазин). У комп'ютерних програмах поряд з інструкціями розгалуження (тобто вибором шляху дії) також існують інструкції циклів (повторення дії). Якби інструкцій циклу не існувало, довелося б багато разів вставляти в програму один і той же код підряд стільки разів, скільки потрібно виконати однакову послідовність дій.

Універсальним організатором циклу в мові програмування Python (як і в багатьох інших мовах) є конструкція **while**. Слово "**while**" з англійської мови перекладається як "доки" ("доки логічний вираз має значення True, доти будуть виконуватись певні операції"). Конструкцію **while** мовою Python можна описати наступною схемою:

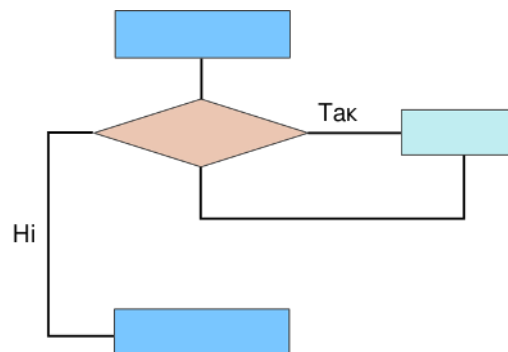
while **a** *логічний оператор* **b**:

дії (або дія)

зміна **a**

Ця схема приблизна, тому що логічний вираз в заголовку циклу **while** може бути більш складним, а змінюватися може змінна (або вираз) **b**.

Може виникнути питання: "Навіщо змінювати **a** або **b**?". Коли виконання програмного коду доходить до циклу **while**, виконується логічний вираз в заголовку, і, якщо було отримано True, виконуються вкладені вирази. Після потік виконання програми знову повертається в заголовок циклу **while**, і знову перевіряється умова. Якщо умова ніколи не буде змінюватись, то не буде причин зупинки циклу і програма зациклиться. Щоб цього не сталося, необхідно передбачити можливість виходу з циклу. Таким чином, змінюючи значення змінної в тілі циклу, можна довести логічний вираз до False і цикл завершиться.



Цю змінювану змінну, яка заголовку циклу **while**,

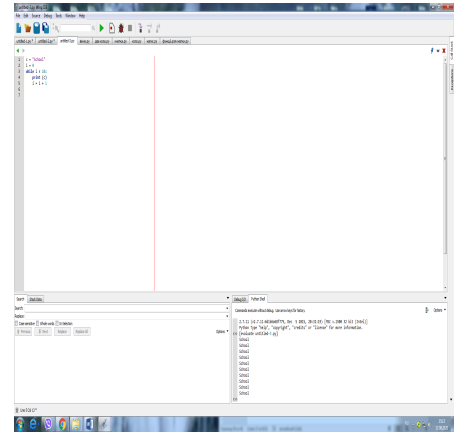
називають лічильником. Як і всякій змінній їй можна давати довільні імена, однак

використовується в зазвичай

дуже часто використовують букви *i* та *j*. Найпростіший цикл на мові програмування Python може виглядати так:

```
c = "School"
i = 0
while i < 10:
    print (c)
    i = i + 1
```

В останньому рядку коду відбувається збільшення значення змінної *i* на одиницю, тому з кожним колом циклу її значення збільшується. Коли буде досягнуте число **10**, логічний вираз ***i*<10** дасть **False**, виконання тіла циклу буде припинено, а потік виконання програми перейде на наступні команди, що слідують після циклу. Результатом виконання вищевказаного скрипту є виведення на екран десяти слів “School” в стовпчик. Якщо збільшувати лічильник в тілі циклу не на одиницю, а на 2, то буде виведено тільки п'ять знаків, так як цикл зробить лише п'ять обертів.



Задача. Знайти суму перших 5 чисел.

1. Пояснення

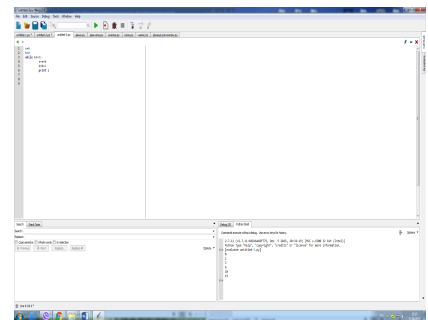
На початку сума (деяка величина *s*) дорівнює нулю, оскільки не було додано жодного числа. Лічильник *k* теж дорівнює нулю. Ми будемо збільшувати значення лічильника до тих пір, поки воно не стане рівним 5. Після кожного збільшення значення лічильника відбувається збільшення значення суми на величину *k*.

2. Складемо таблицю перевірки умови

k	Перевірка умови	s
0	+	0
1	+	1
2	+	3
3	+	6
4	+	10
5	+	15
6	-	-

3. Напишемо програму розв'язання даної задачі мовою Python.

```
s=0  
k=0  
while k<=5:  
    s=s+k  
    k=k+1  
print s
```



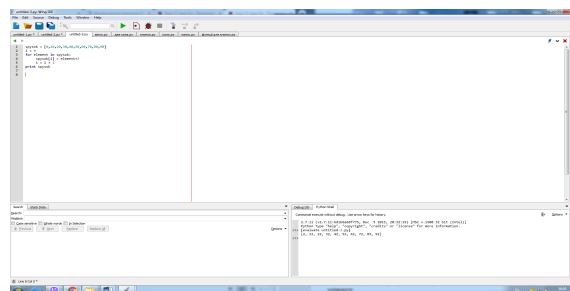
Завдання для самостійного виконання

1. Напишіть програму, яка виводить на екран ваше ім'я 20 разів.
2. Змініть її так, щоб на екран виводилася лише половина цих слів.
3. Змініть її, щоб на екран виводилося одне слово.
4. Знайдіть суму перших 100 натуральних чисел.
5. Знайдіть суму перших 20 парних натуральних чисел.
6. Знайдіть суму перших 20 непарних натуральних чисел.
7. Знайдіть добуток перших 10 натуральних чисел.
8. Знайдіть добуток перших 5 парних натуральних чисел.
9. Знайдіть добуток перших 5 непарних натуральних чисел.
10. Змініть програму так, щоб вона працювала для будь-якої кількості чисел.

§ 10. Цикли. Оператор циклу *for*

У попередньому параграфі було розглянуто цикл **while**. Однак це не єдиний спосіб організації повторення групи виразів у мові Python. У програмах, написаних на Python, широко застосовується цикл **for**, який являє собою цикл обходу заданої множини елементів (символів рядка, об'єктів списку або словника) та виконання в своєму тілі різних операцій над ними. Наприклад, якщо є список чисел, і необхідно збільшити значення кожного елемента на дві одиниці, то можна перебрати список за допомогою циклу **for**, виконавши над кожним його елементом відповідну дію.

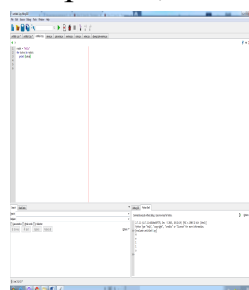
```
spysok = [0,10,20,30,40,50,60,70,80,90]  
i = 0  
for element in spysok:  
    spysok[i] = element+2  
    i = i + 1  
print spysok
```



У прикладі змінна *i* потрібна для того, щоб записати змінене значення елемента в список. У ній зберігається значення індексу чергового елемента списку. У той час, як змінна *element* пов'язується зі значенням чергового елемента даних. Узаголовку циклу **for** відбувається звернення до чергового елемента списку. У тілі циклу елементу з індексом *i* присвоюється сума значення поточного (оброблюваного) елемента і двійки. Далі індекс збільшується на одиницю, а потік виконання програми переходить знову в заголовок циклу **for**, де відбувається звернення до іншого елемента списку. Коли всі елементи оброблені цикл **for** припиняє свою роботу. Відсутність чергового елемента є умовою завершення роботи циклу **for** (для порівняння: в циклі **while** умовою завершення служить результат **False** логічного виразу в заголовку).

З таким само успіхом перебирати можна й рядки, якщо не намагатися їх при цьому змінювати:

```
radok = "Hello"
```



*for bukva in radok:
print (bukva)*

Спробуйте змінити вищевказаний приклад, написавши, наприклад, *print (bukva, '*')*, або *print(bukva+' *')* і ви побачите, як зміниться результат.

Цикл **for** широко використовується в мові програмування Python, тому що є важливим інструментом при обробці структур даних. Також слід запам'ятати, що цикл **for** в Python особливий. Він не є аналогом циклів **for** в багатьох інших мовах програмування, де представляє собою, так званий, цикл з лічильником.

Завдання для самостійного виконання

1. Створіть список, що складається з чотирьох рядків. Потім, за допомогою циклу **for**, виведіть рядки по черзі на екран.
2. Змініть попередню програму так, щоб в кінці кожної букви рядка додавалося тире. (Підказка: цикл **for** може бути вкладений в інший цикл.)
3. Створіть список, що містить елементи цілочисельного типу, потім за допомогою циклу перебору змініть тип даних елементів на числа з плаваючою крапкою. (Підказка: використовуйте вбудовану функцію `float()`.)

§ 11. Вкладені цикли

При розв'язанні деяких задач необхідно у тілі циклу робити допоміжні обчислення або перевірки. Іноколи для їх здійснення достатньо зробити арифметичну дію, а в інших випадках потрібно застосувати ще один цикл.

Вкладеним циклом називають конструкцію, в якій один цикл виконується всередині другого. Внутрішній цикл виконується повністю під час кожної ітерації зовнішнього циклу.

Вкладені цикли

Цикл називають вкладеним, якщо він міститься в тілі іншого циклу (його також називають **внутрішнім**), а цикл, у якому він міститься, — **зовнішнім**.

Вкладені цикли організовані таким чином: внутрішній цикл повністю вміщується в тілі зовнішнього циклу.

Тут А — **зовнішній** цикл, В — **внутрішній**.

Як внутрішній, так і зовнішній цикли можуть бути циклами

з параметром або **з умовою**

Вкладені цикли

Порядок виконання вкладених циклів:

1. При першій ітерації (повторенні) зовнішнього циклу викликається внутрішній, який виконується до свого завершення.
2. Після цього керування передається в тіло зовнішнього циклу.
3. При другій ітерації зовнішнього циклу знову викликається внутрішній.
4. І так триватиме доти, поки не завершиться зовнішній цикл.

Новая вкладка x Множественное ветвление: if-else x Урок 12. Циклы for - Программу x Презентация PowerPoint x 8 Цикл з п лінійним кроком.pdf x

uvk14.at.ua/informatika_3/prezentacija_6_klas_urok_26.pdf

Вкладені цикли

Щоб створити такий малюнок, потрібно малювання п'ятикутника повторити 10 разів.

Зовнішній цикл

Внутрішній цикл

```

for m in range(10):
    down()
    for n in range(5):
        color(colors[n])
        forward(20)
        left(360/5)
    up()
    forward(30)
  
```

Тіло зовнішнього циклу (за m)

Тіло внутрішнього циклу (за n)

Вкладені цикли

Цикл називають вкладеним, якщо він міститься в тілі

111183.zip FotoCollage.exe

17:35 13.08.2020

Новая вкладка x Множественное ветвление: if-else x Урок 12. Циклы for - Программу x Презентация PowerPoint x 8 Цикл з п лінійним кроком.pdf x

uvk14.at.ua/informatika_3/prezentacija_6_klas_urok_26.pdf

кількість кроків (наприклад, 5) і припинити роботу циклу.

Вкладені цикли

Нехай нам потрібно засобами «черепашкої» графіки відтворити малюнок, наведений на рисунку.



Завантажимо команди для роботи з Черепашкою і встановимо ширину сліду Черепашки в 3 пікселі:

```

from turtle import *
width(3)
up() # Перо підняте
colors = ['yellow', 'blue', 'green', 'purple', 'red']
# Список кольорових констант
  
```

111183.zip FotoCollage.exe

17:36 13.08.2020

Новая вкладка x Множественное ветвление: if-else x Урок 12. Циклы for - Программу x Урок 8. Циклы. Циклы с лічильником x 8 Цикл з лічильником.py - x

drive.google.com/file/d/1ZeFshzmpV4kZiZFw4bTWLXbR4fXK/view

Вкладені цикли

Приклад 1. Дано натуральне число n ($n \leq 9$). Вивести числові рядки, у яких i -ий рядок складається із чисел від 1 до i .

Вхід	Вихід
4	1
	12
	123
	1234

```
n=int(input())
for i in range(1,n+1):
    for j in range(1,i+1):
        print(j, end="")
    print()
```

У прикладі використовується вкладений цикл з параметром j , який змінюється в залежності від параметра i у зовнішньому циклі.

Вкладені цикли

111183.zip FotoCollage.exe

Показать все

17:10 13.08.2020

Новая вкладка x Множественное ветвление: if-else x Урок 12. Циклы for - Программу x Урок 8. Циклы. Циклы с лічильником x 8 Цикл з лічильником.py - x

drive.google.com/file/d/1ZeFshzmpV4kZiZFw4bTWLXbR4fXK/view

print(j, end="")
print()

вкладений цикл з параметром j , який змінюється в залежності від параметра i у зовнішньому циклі.

Вкладені цикли

Приклад 2. Дано натуральне число n ($n \leq 100$). Вивести всі різні піфагорові трійки у порядку зростання чисел, що не перевищують числа n . Вхід

Вхід	Вихід
10	3 4 5
	6 8 10

```
n=int(input())
for a in range(1,n+1):
    for b in range(a,n+1):
        for c in range(b,n+1):
            if a**2+b**2==c**2:
                print(a,b,c)
```

Натуральні числа a , b і c , що задовольняють умову $a^2+b^2=c^2$, називають піфагоровими. Наприклад, 3, 4 і 5 є піфагоровою трійкою, бо $3^2+4^2=5^2$.

У прикладі використовується потрійний цикл.

Завдання для самостійного виконання

Дано натуральне число n . Вивести кількість додатних рядків, у яких сума чисел дорівнює n .

Сторінка 9 / 11

Показать все

17:11 13.08.2020

Завдання для самостійного виконання

1. Дано натуральне число n . У наступних n рядках дано по одному цілому числу. Виведіть кількість додатних серед цих чисел.

Вхід	Вихід
4 12 -4 0 20	2

2. Дано натуральне число n . Виведіть у один рядок всі дільники цього числа у порядку зростання.

Вхід	Вихід
6	1 2 3 6

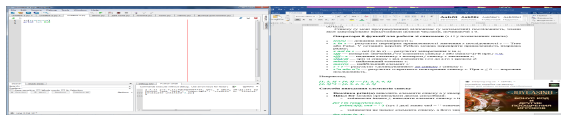
§ 12. Робота зі списками. Пошук максимального та мінімального значення

У мові Python відповідні зручні структури даних називають *послідовностями*. Це *рядки* (символів), *списки*, *масиви*, *значення функції range* і ще деякі інші об'єкти.

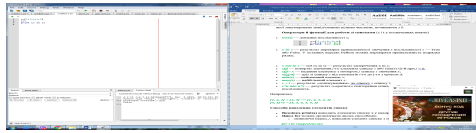
Списку (у мові програмування) відповідає (у математиці) послідовність, члени якої занумеровано невід'ємними цілими числами, починаючи з 0.

Оператори й функції для роботи зі списками (s і t у позначеннях нижче)

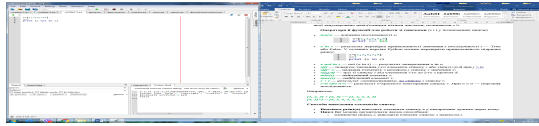
- ***len(s)*** — довжина послідовності s ;



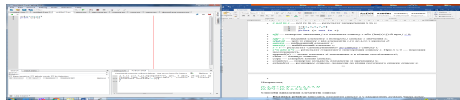
- ***x in s*** — результат перевірки приналежності значення x послідовності s — True або False. У останніх версіях Python можна перевіряти приналежність підрядка рядку;



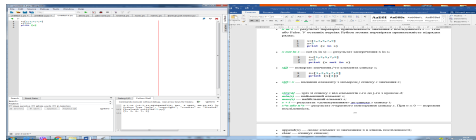
- $x \text{ not in } s$ — not (x in s) — результат заперечення x in s;



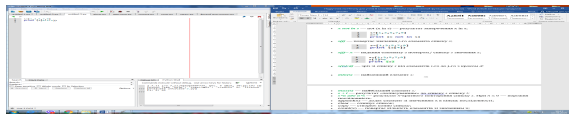
- $s[j]$ — повертає значення j-го елемента списку s;



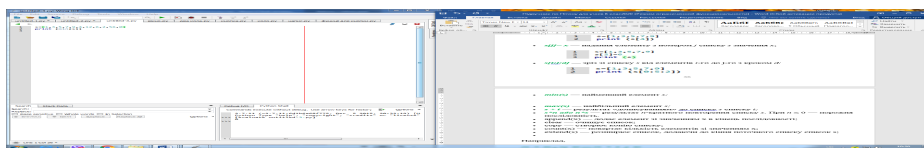
- $s[j] = x$ — надання елементу з номером j списку s значення x;



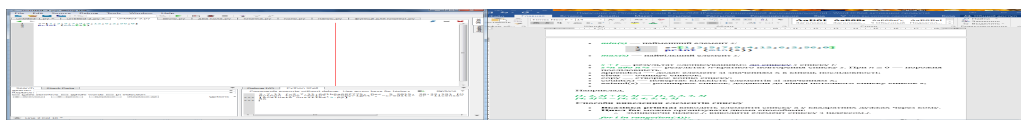
- $s[i:j:d]$ — зріз зі списку s від елементів i-го до j-го з кроком d;



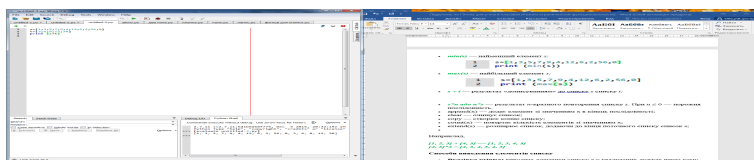
- $\min(s)$ — найменший елемент s;



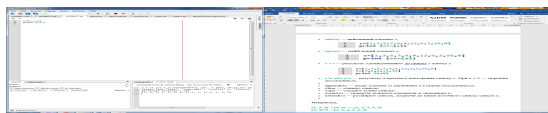
- $\max(s)$ — найбільший елемент s;



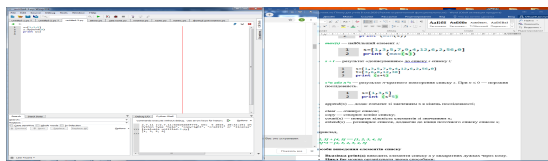
- $s + t$ — результат «дописуванням» до списку s списку t;



- $s*n$ або $n*s$ — результат n -кратного повторення списку s . При $n \leq 0$ — порожня послідовність.



- `append(x)` — додає елемент зі значенням x в кінець послідовності;



Завдання для самостійного виконання

1. Вивести на екран список з 10 елементів.
2. Знайти довжину даного списку.
3. Визначити, чи належить елемент даному списку. Розглянути випадки, коли елемент належить і не належить списку.
4. Розв'язати попередню задачу з використанням заперечення.
5. Вивести на екран порядковий номер заданого елемента.
6. Вивести на екран список [23,45,12,78,567,3,90]. Замінити всі непарні елементи списку на парні.
7. Вивести на екран список з 10 елементів. Отримати зріз цього списку з кроком 3.
8. Вивести на екран список з 10 елементів. Знайти найбільший елемент цього списку.

9. Вивести на екран список з 10 елементів. Знайти найменший елемент цього списку.
10. Знайти суму списків, які містять ваше прізвище, ім'я та по батькові.
11. Продублювати тричі список, який містить 5 елементів.
12. Вивести на екран список, який містить число та місяць вашого народження.
Додати до нього значення року народження.

§ 13. Поняття об'єкта в мові програмування, його властивостей і методів

Клас - це абстрактний тип даних. За допомогою класу описується деяка сутність (її характеристики і можливі дії). Наприклад, клас може описувати студента, автомобіль і т.д. Описавши клас, ми можемо створити його примірник - Об'єкт.

Об'єкт - це вже конкретний представник класу.

Клас	Об'єкт
------	--------

Студент – людина, яка здобуває знання у вищому навчальному закладі.	
Автомобіль – транспортний засіб, оснащений двигуном.	

Крім назви кожний об'єкт має низку **властивостей** (параметрів), значення яких повністю описує об'єкт.

Властивості поділяють на:

кількісні — такі властивості можна подати числом у певних одиницях виміру (метрах, секундах, грамах);

якісні — такі властивості можна лише назвати. Наприклад, характер учня може бути спокійним або неврівноваженим.

Значення властивостей об'єкта у певний момент часу визначають його **стан**.

Завдання для самостійного виконання

1. Заповни таблицю.

Об'єкт	Властивості	Методи (події)
	Який він? Чим особливий?	Що він робить? Що з ним можна робити?
Яблуко		
Ручка		
Цукерка		

Мишка		
Лампочка		

2. Заповни таблицю, вказавши щонайменше по три властивості об'єкта та їх значення.

Об'єкт	Властивості	Значення властивості
Яблуко	1. Колір 2. Смак 3.	Червоний Солодкий

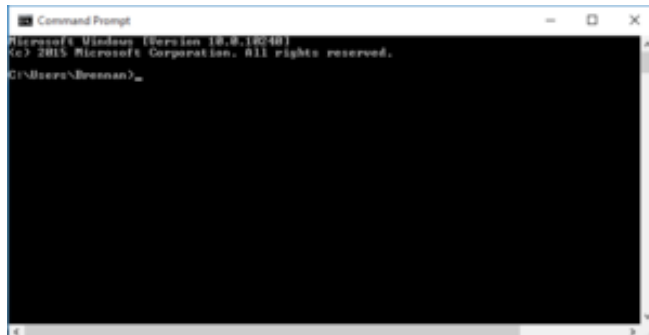
§ 14. Графічний інтерфейс, основні компоненти програми з графічним інтерфейсом

Інтерфейс користувача — засіб зручної взаємодії користувача з інформаційною системою.

Графічний інтерфейс



Текстовий інтерфейс



При створенні програм з графічним інтерфейсом користувача потрібно передбачити взаємодію з програмою за допомогою різних:

- кнопок,
- меню,
- піктограм,
- введення даних у спеціальні поля,
- списки для вибору значень
- тощо.

Відповідні візуальні форми називають **віджетами** (window gadget — англійською штучка). Для мови програмування Python такі віджети включено у спеціальну бібліотеку tkinter. Потрібно імпортувати бібліотеку у програму, щоб використовувати її складові, створюючи графічний інтерфейс і вікна повідомлень.

Tkinter (від англ. tk interface) - це графічна бібліотека, яка дозволяє створювати програми з віконним інтерфейсом.

Tk є базовим класом будь-якого Tkinter-додатку. При створенні об'єкта цього класу запускається інтерпретатор та створюється базове вікно додатку.

Етапи створення програми з графічним інтерфейсом користувача

1. **Імпорт модуля Tkinter** можна здійснити таким чином:

*from Tkinter import **

Надалі ми будемо користуватися цим способом, бо це дає можливість не вказувати щоразу назву модуля при зверненні до об'єктів, які він містить. У версії Python 3 назву модуля пишуть з малої літери (tkinter), хоч у попередніх версіях використовують велику (Tkinter).

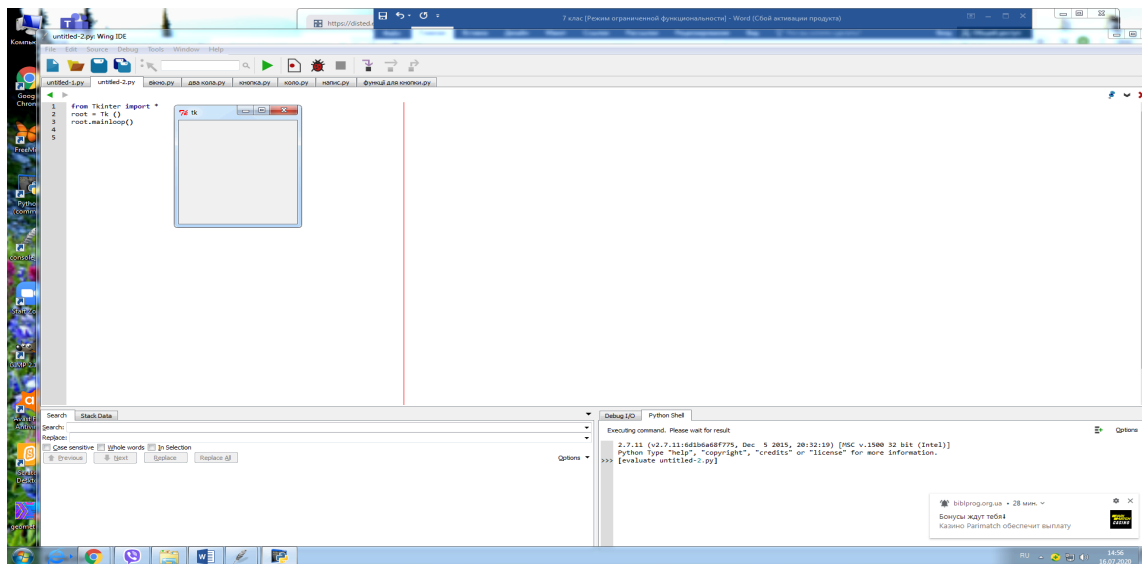
2. Створення головного вікна. В сучасних операційних системах будь-яку клієнтську програму подано вікном. Його можна назвати головним, бо в ньому розташовують всі інші віджети. Об'єкт вікна верхнього рівня створюють при зверненні до класу Tk модуля Tkinter. Змінну, пов'язану з об'єктом-вікном, зазвичай називають **root** (корінь), хоча обмежень на назву немає. Другий рядок програми має такий вигляд:

```
root = Tk()
```

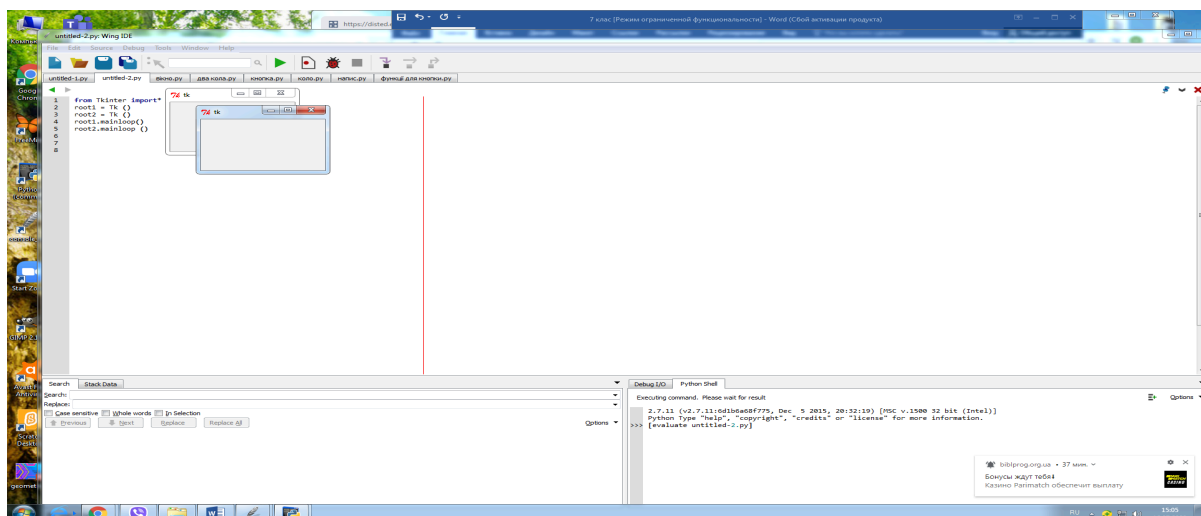
3. Запуск головного вікна здійснюється за допомогою команди

```
root.mainloop()
```

Отже, програма для створення вікна буде мати вигляд:

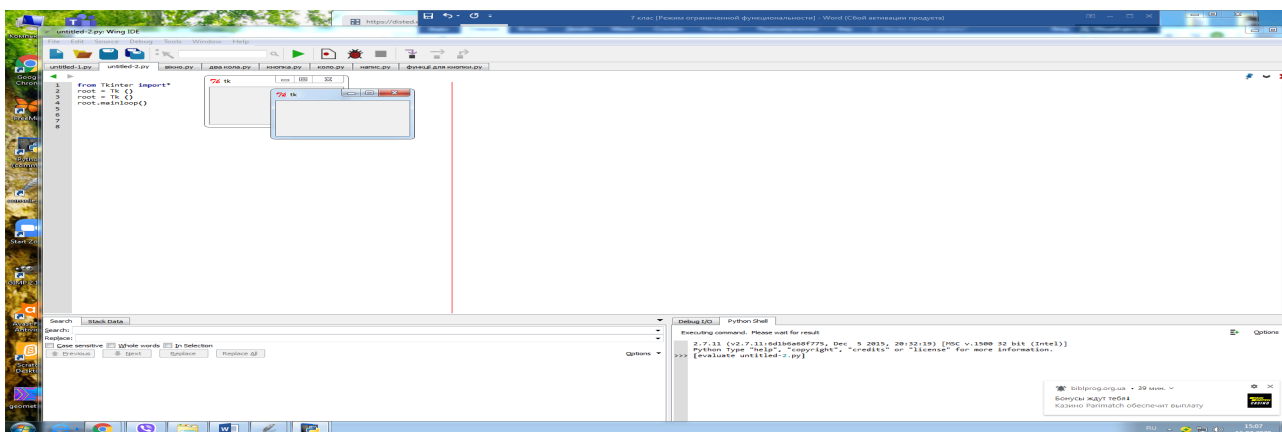


Можна використовувати кілька викликів інтерпретатора Tk.



1.

2.



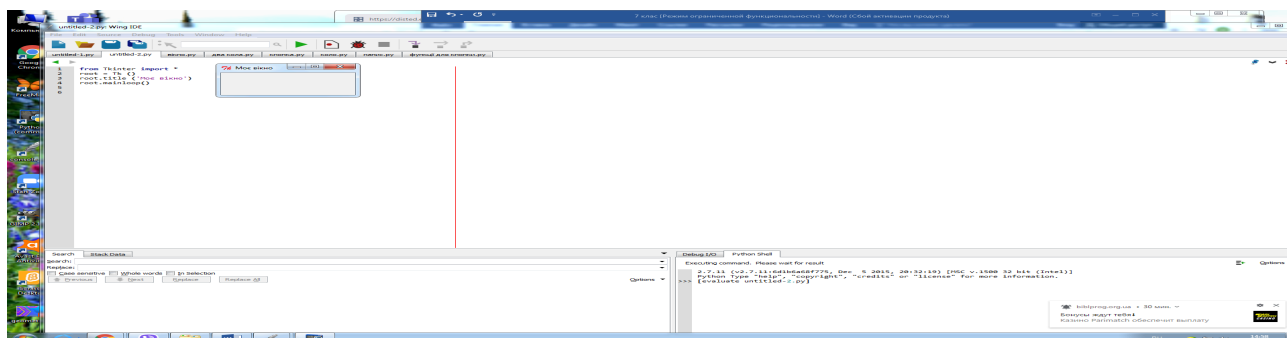
При використанні двох чи більше викликів інтерпретатора *необхідно, щоб об'єкти, створені в одному циклі `mainloop`, було використано лише в ньому*. Необхідність використання кількох циклів у одному застосунку виникає вкрай рідко.

4. Назва головного вікна

Далі створюємо заголовок вікна. Команда для створення заголовка:

`root.title ('Моє вікно')`

А програма, яка створює вікно з назвою «Моє вікно» має вигляд:



5. Параметри головного вікна

При створенні вікна можна задавати його параметри: розміри, колір, тип та товщину рамки. Для цього використовується команда:

`root.config (bg = "red", width = 200, height = 100, relief = FLAT, bd = 10)`

Параметри вікна	Пояснення
<i><code>bg</code></i>	Колір вікна
<i><code>width</code></i>	Ширина вікна

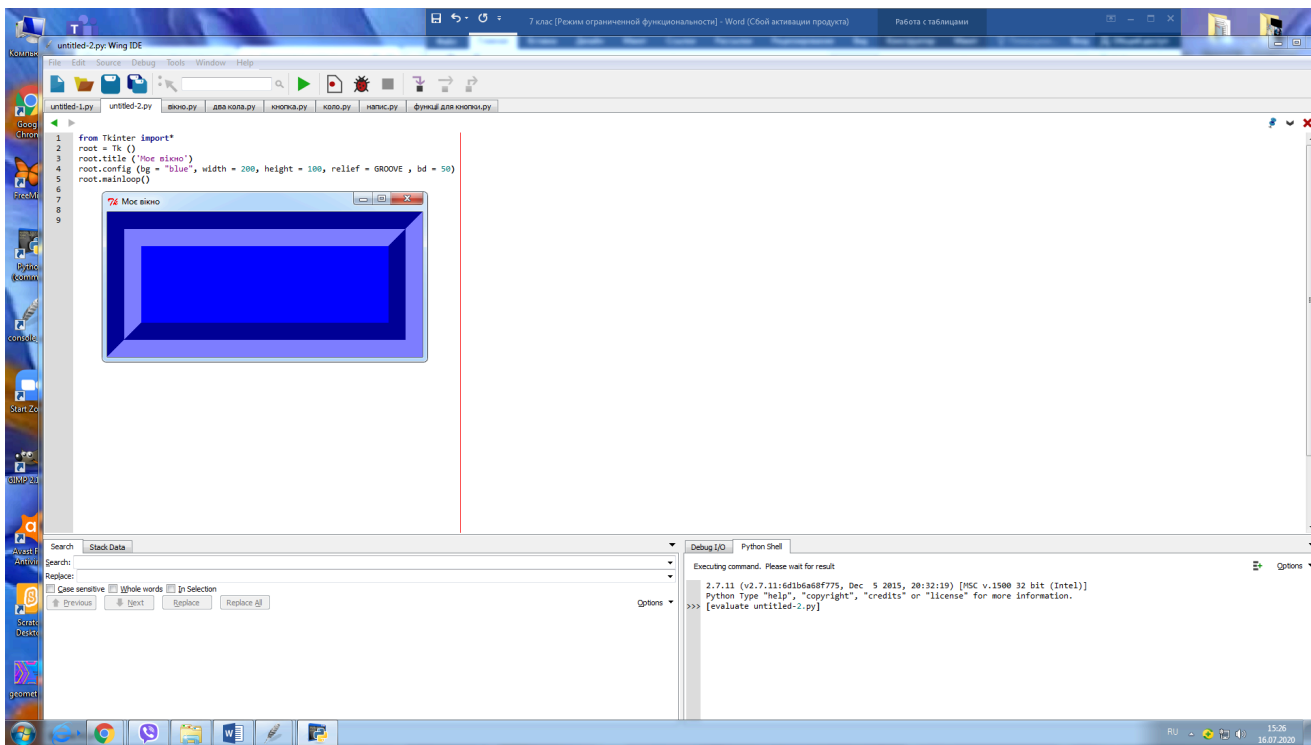
<i>height</i>	Висота вікна
<i>relief</i>	Тип рамки
<i>bd</i>	Товщина рамки

6. Типи рамок:

FLAT (плоска рамка)	SUNKEN (увігнута рамка)	RAISED (опукла рамка)	GROOVE (у вигляді жолобу)	RIDGE (у вигляді хребта)
				

7. Побудова вікна

Рядки програми	Пояснення
<i>from Tkinter import *</i>	підключення бібліотеки Tkinter
<i>root = Tk()</i>	створення вікна
<i>root.title ('...')</i>	назва вікна
<i>root.config (bg = "колір_фону", width = ширина_вікна, height = висота_вікна, relief = тип_рамки, bd = товщина_рамки)</i>	Задання конфігурації (параметрів вікна)
<i>root.mainloop()</i>	Запуск вікна



Завдання для самостійного виконання

1. Знайди слова, що позначають елементи графічного інтерфейсу.

Й	Ї	В	Й	М	П	Т	Ц	У	Р	Д
Т	И	А	Л	В	І	К	Н	О	Е	Ч
Ї	Д	Г	З	Щ	К	Н	О	П	К	А
В	І	Д	Ж	Е	Т	Ь	Є	І	Я	Х
Ф	И	Д	Н	С	О	Я	Ю	Е	П	Ж
Ф	К	З	Г	Є	Г	Щ	О	Я	П	Ж
К	У	Р	С	О	Р	П	Б	Ф	В	Л
С	Щ	Н	Ь	Ж	А	О	Ц	Ї	Ї	Л
Ч	А	Ї	Ж	Ч	М	Е	Н	Ю	П	Ї
Р	Ц	Є	Д	О	А	В	Щ	Х	Є	В

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____

2. Встанови відповідність між командою та її значенням:

- | | |
|----------------------|-----------------------|
| А. root = Tk() | 1. Конфігурація вікна |
| Б. root.title('...') | 2. Запуск вікна |
| В. root.mainloop() | 3. Заголовок вікна |
| Г. root.config | 4. Створення вікна |

3. Запиши програму, яка створює та запускає вікно з назвою My first window.

§ 15. Поняття елемента керування. Обробники подій, пов'язаних з елементами керування

Всі віджети в Tkinter мають деякі спільні властивості. Розглянемо деякі з них перед вивченням конкретних випадків. Віджети створюються викликом конструктора відповідного класу.

Віджет = клас(**master** , **options** =)

Де:

- **master** - батьківське вікно.
- **options** - список найбільш часто використовуваних властивостей для цього віджета. Це може бути шрифт (font=...), колір віджета (bg=...), команда, що виконується при активації віджета (command=...) тощо.

В таблиці 1 вказано основні властивості віджетів та їх значення:

	Варіант і опис
1	bd Ширина межі в пікселях. За замовчуванням - 2.
2	bg Колір фону.
3	Cursor Курсор, який використовується на полотні, наприклад <i>стрілка, коло, точка тощо</i>
4	Height Розмір полотна в Y-вимірі.
5	highlightcolor Колір, зображений у фокусі.
6	Relief Рельєф визначає тип межі. Деякі значення є SUNKEN, RAISED, GROOVE і RIDGE.
7	Width Розмір полотна в розмірності X.

Кожен об'єкт, окрім властивостей, має набір подій, які можуть з ним статися. Так, наприклад, користувач під час роботи з програмою може натиснути кнопку, підвести курсор миші до напису, вибрати один з елементів списку, ввести в текстове поле деяку інформацію тощо – все це події. Завдання програміста – написати обробники для подій, що потрібні для розв'язання поставленої задачі; керувати подіями шляхом написання програмного коду для обробки таких подій.

Отже, програму можна розглядати як набір подій та обробників цих подій. Програмний код, який обробляє ту чи іншу подію об'єкта, називають методом цього об'єкта.

Найпростішим прикладом обробника події є зміна властивості деякого об'єкта.

Завдання для самостійного виконання

1. Спробуй заповнити таблицю, полиставши цей посібник.

Об'єкт	Властивості	Події
Вікно		
Прапорець		

2. Напиши програму, яка зображає вікно синього кольору з назвою “My second window”, розмірами 200x50, товщиною рамки 10 пікселів.
3. Напиши програму, яка відображає вікно, використовуючи різні властивості віджетів з таблиці 1 та їх значення.

§ 16. Текстовий напис (вікно повідомлень). Властивості та методи елементів керування

У вікні можна розміщувати різні об'єкти. Наприклад, для розміщення тексту у вікні використаємо клас **Label** (з англ. ярлик).

Створимо конкретний об'єкт цього класу з іменем **label**.

```
label = Label(root, text='Hello!')
```

У дужках необхідно вказати, у якому вікні буде розташований текст і який саме текст. Тому першою опцією є ім'я створеного вікна **root**, другий параметр **text** приймає значення 'Hello!'.

До об'єкту **label** треба застосувати метод **pack()**.

```
label.pack()
```

Якщо цей метод не використати, текст у вікні не буде відображеним (не з'явиться).

Щоб додати текст до вікна, потрібно виконати певну послідовність дій.

1. Підключити бібліотеку Tkinter.
2. Створити вікно.
3. Назвати його.
4. Задати конфігурацію (параметри).
5. Створити надпис (текст) і задати його параметри.
6. Розмістити текст у вікні.
7. Запустити вікно.

Параметри:

1. **width** – ширина;
2. **height** – висота;
3. **background(bg)** - колір фону;
4. **foreground(fg)** – колір символів;
5. **font = ('Arial', 40)** – шрифт та розмір тексту;

Розглянемо приклади.

Задача № 1

Написати програму, яка зображає вікно синього кольору з назвою “My first window”, розмірами 200x50, товщиною рамки 10 пікселів. Додати надпис “Hello!!!” жовтого кольору, шрифт – Arial, розмір - 40.

```
from Tkinter import *  
win=Tk ()
```



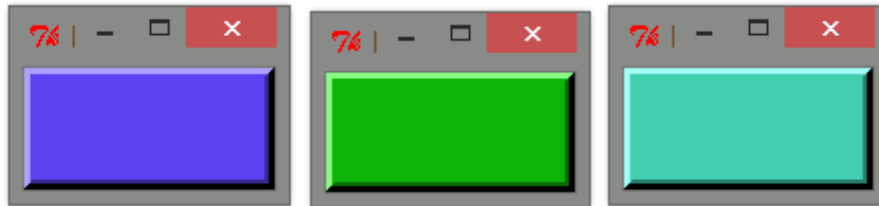

```

win.title ('My first windows')
win.config (width=200, height=50,bg= "blue", relief=RAISED, bd=10)
lab1=Label (win, text='Hello!!!', fg='yellow', font = ('Arial', 40))
lab1.pack()
win.mainloop()

```

Задача № 2

Написати програму зображення вікна розмірами 40x60, назвою “My windows” з товщиною рамки 5, колір якого генерується програмою.



Для генерування кольору слід задати “кількість” червоного, зеленого та синього кольорів (числом у межах 1 байта). Потім змішати кольори.

```

red = randint (0, 255)
green = randint (0, 255)
blue = randint (0, 255)
col = "#%02x%02x%02x" %(red, green, blue)

```

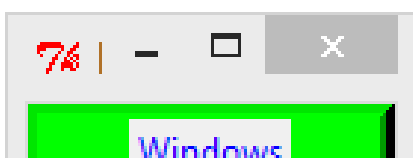
```

from Tkinter import *
from random import *
win=Tk ()
win.title ('My windows')
red = randint (0, 255)
green = randint (0, 255)
blue = randint (0, 255)
col = "#%02x%02x%02x" %(red, green, blue)
win.config (width = 40, height = 60, bg = col, relief = RAISED, bd = 5)
win.mainloop()

```

Задача № 3

Написати програму, яка зображуватиме вікно із однією з 3-ох описаних надписів “Microsoft Word”, “Windows” або “Python” (вибір надпису здійснює програма). Параметри кожного тексту задати на вибір.



```
from Tkinter import *
from random import *
win=Tk()
win.title ('My windows')
win.config (width = 40, height=60, bg='green', relief=RAISED, bd=5)
lab1 = Label (win, text='Microsoft Word', fg='red', font = ('Arial',40))
lab2 = Label (win, text='Windows', fg='blue')
lab3 = Label (win, text='Python', fg='yellow', bg='green', font='Centaur')
lab = choice ([lab1, lab2, lab3])
lab.pack()
win.mainloop()
```

Завдання для самостійного виконання

1. Встанови відповідність між параметром та його значенням..

A. foreground(fg)	1. Колір фону
Б. font = ('Arial', 40)	2. Колір символів
В. background(bg)	3. Розмір тексту
Г. height	4. Шрифт тексту

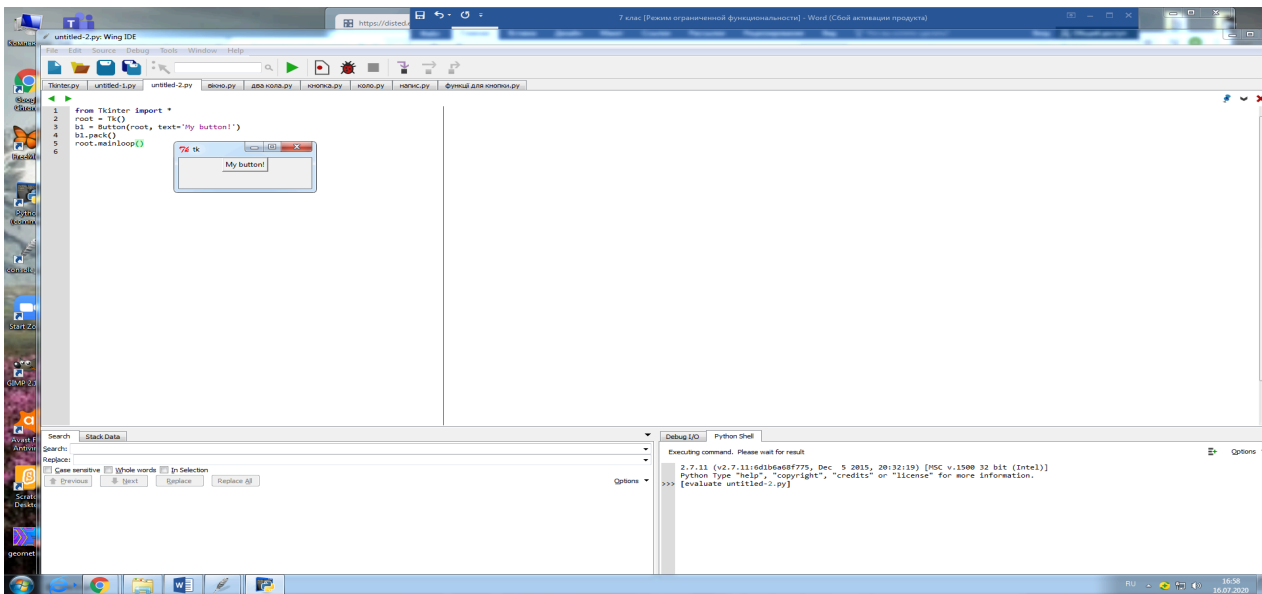
2. Напиши програму створення вікна розмірами 100x100, з назвою “My class” з товщиною рамки 10, колір якого генерується програмою.
3. Напиши програму зображення вікна розмірами 200x50, з назвою “Принцип ІВМ” з товщиною рамки 10 та написом у вікні «Машини повинні працювати. Люди повинні думати».
4. Напиши програму створення вікна розмірами 200x200, з назвою “My window” з товщиною рамки 10 та трьома написами у вікні «Прізвище», «Ім'я», «По батькові» .
5. Напиши програму створення трьох вікон розмірами 100x150 з товщиною рамки 5 та трьома написами у них: «Скоро залишаться лише дві групи працівників: ті, хто контролює комп'ютери, і ті, кого контролюють комп'ютери. Постарайтеся потрапити в першу», «Людина в ХХІ столітті, яка не буде вміти користуватися ЕОМ, буде подібна людині ХХ століття, що не вміє ні читати, ні писати», «Комп'ютер одного разу переміг мене в шахи, але в кікбоксингу він виявився просто слабак». У назвах вікон вкажи авторів висловлювань. (Google в допомогу ☺)

§ 17. Кнопка. Властивості та методи елементів керування

Створимо вікно, в якому розмістимо кнопку. Для створення кнопок використовується клас **Button**. Команда, яка відповідає за створення кнопки, має вигляд:

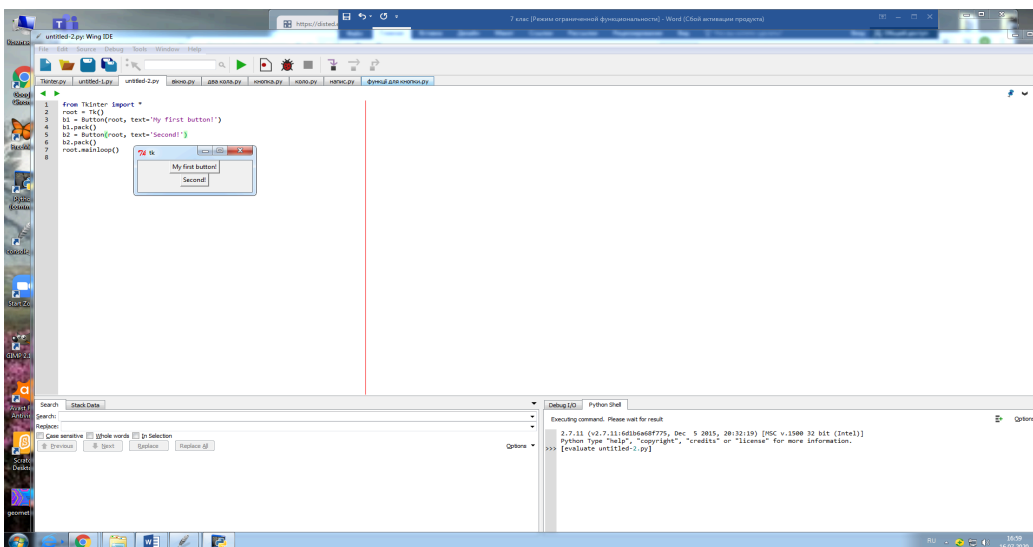
```
b1 = Button (root, text='My button!', bg='red')  
b1.pack()
```

Напишемо програму, яка створює кнопку у вікні.

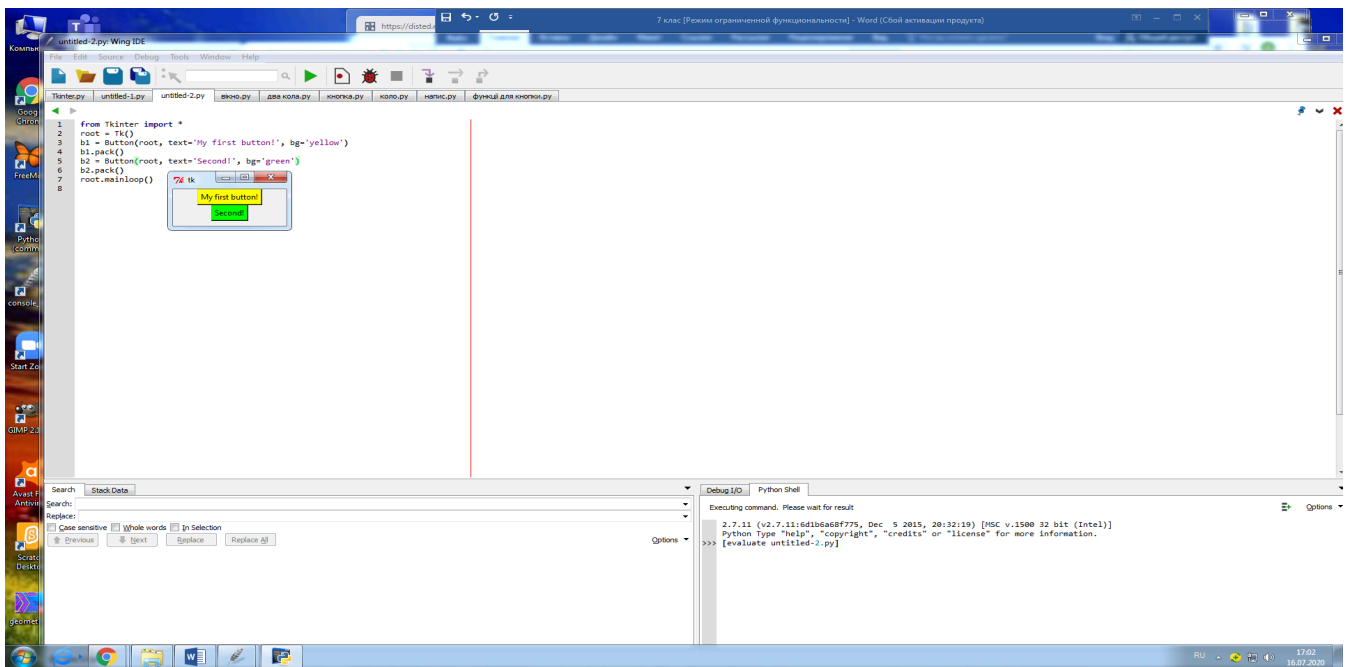


Якщо розміри кнопки не задані, то вони визначаються текстом на ній.

Створимо вікно з двома кнопками, що будуть розташовуватися одна під одною.



Задамо колір для наших кнопок.



Завдання для самостійного виконання

1. Записати команду, яка дозволяє створити кнопку з назвою «Кнопка».
2. Скільки кнопок можна створити у вікні?
3. Записати програму, яка створює кнопку «Кнопка» у вікні.
4. Записати програму, яка створює у вікні кнопку жовтого кольору.
5. Записати програму, яка створює у вікні дві кнопки різних кольорів.
6. Записати програму, яка створює у вікні три кнопки різних кольорів з назвами, що позначають ваше прізвище, ім'я та по батькові.
7. Записати програму, яка створює три вікна різних кольорів. У першому вікні додати кнопку «Прізвище», у другому – «Ім'я», у третьому – «По батькові».
8. Записати програму, яка створює вікно будь-якого кольору з назвою «Довільна кнопка». У вікні розмістити одну з кнопок «Прізвище», «Ім'я», «По батькові», використовуючи модуль випадкових чисел.

§ 18. Функції у програмуванні

Що таке функція?

Функція – це невелика автономна програма (шматок коду), що виконує певне завдання і яку можна використовувати в різних місцях основної більшої програми. Після того як функція створена, її можна використовувати її у будь-який час, у будь-якому місці і скільки завгодно разів.

Таким чином функції економлять час і зусилля, допомагаючи уникати повторювання коду кожен раз, коли потрібно запрограмувати одне і те ж завдання.

Вони дають змогу надати ім'я для блоку інструкцій, і тоді можна буде виконувати цей іменований блок програми будь-де і будь-коли. Для цього потрібно просто вказати ім'я функції. Це називається **викликом функції**.

Python має багато вже готових (вбудованих) функцій, які ми можемо використовувати одразу ж після інсталяції цієї мови, просто 'викликавши' її.

Але, що робити, коли нам самим потрібно написати власну функцію, щоб наприклад заощадити трохи часу, уникнути дублікації коду, а також щоб мати можливість використати її у наших інших програмах?

В цьому випадку функції визначаються за допомогою ключового слова *def*. Для того щоб створити функцію потрібно розмістити ключове слово *def* перед ідентифікатором функції (її ім'ям), потім вказати пару дужок всередині яких можуть міститися імена змінних (аргументи функції), в кінці рядка дві крапки. Після двох крапок (в новому фізичному ряду) йде блок інструкцій, який є тілом функції.

```
def Count():  
    a = input()  
    b = input()  
    P = a+b  
    print P
```

Це приклад визначення функції. Функція складається із заголовка і тіла. Команда закінчується двокрапкою і переходом на новий рядок. Тіло має відступ.

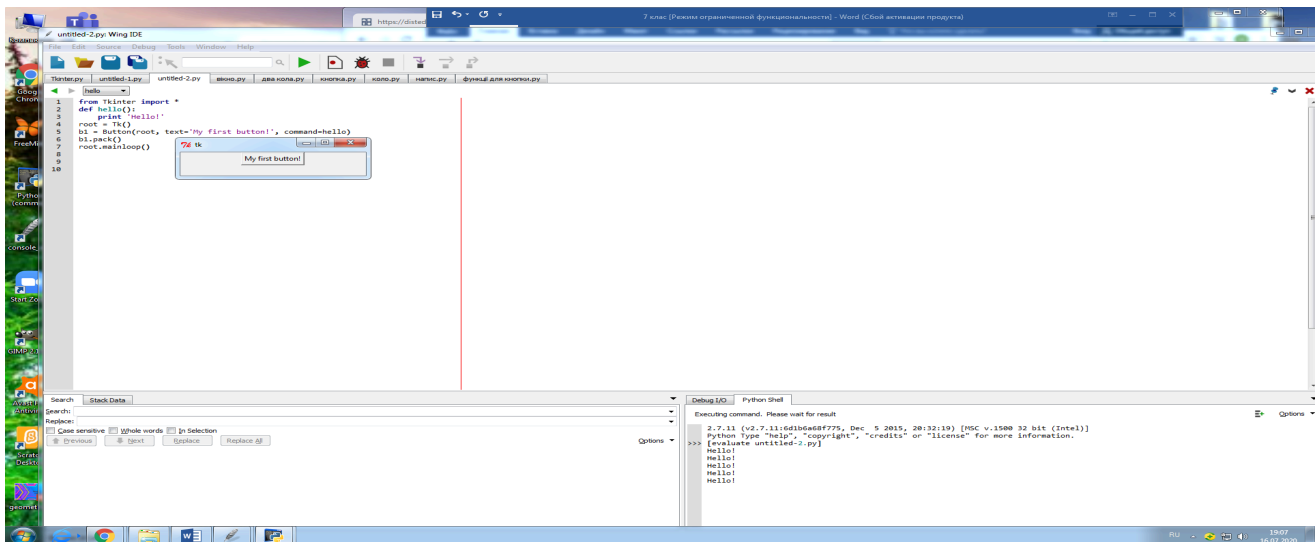
Ключове слово *def* повідомляє інтерпретатору, що перед ним визначення функції. За *def* слідує ім'я функції. Воно може бути будь-яким. У програмуванні дуже бажано давати всьому осмислені імена. Після двокрапки слідує тіло, що містить інструкції, які виконуються при виконанні функції.

§ 19. Функції для кнопок

Ми уже навчилися створювати кнопки. Але при натисненні на них, крім клацання, нічого не відбувається. Це тому, що нашим кнопкам не присвоїли жодних функцій.

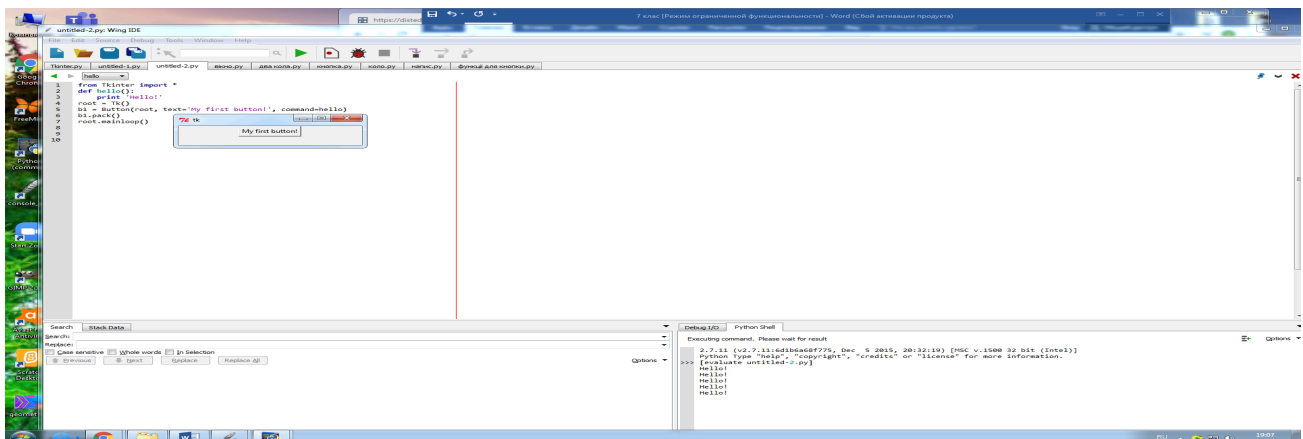
Ускладнимо завдання і створимо кнопку, при натисненні на яку виводиться повідомлення "Hello!".

Подія пов'язана з кнопкою, тому кнопка має містити атрибут, що обробляє цю подію. Цей атрибут носить назву *command*. Яке ж може бути його значення? Яка нам знайома структура виконується лише тоді, коли ми її викликаємо? Це функція. Напишемо функцію, яка виводить дане повідомлення. Потім значенням атрибуту **command** поставимо ім'я функції. Вона буде викликатися лише при натисненні на кнопку.

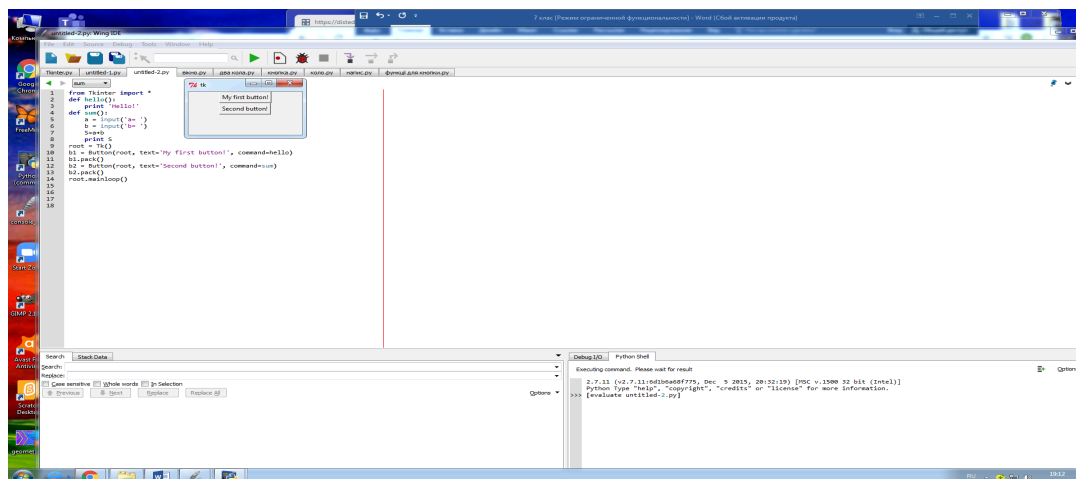


Текст 'Hello!' виводиться не у нашому вікні, а у вікні Python Shell.

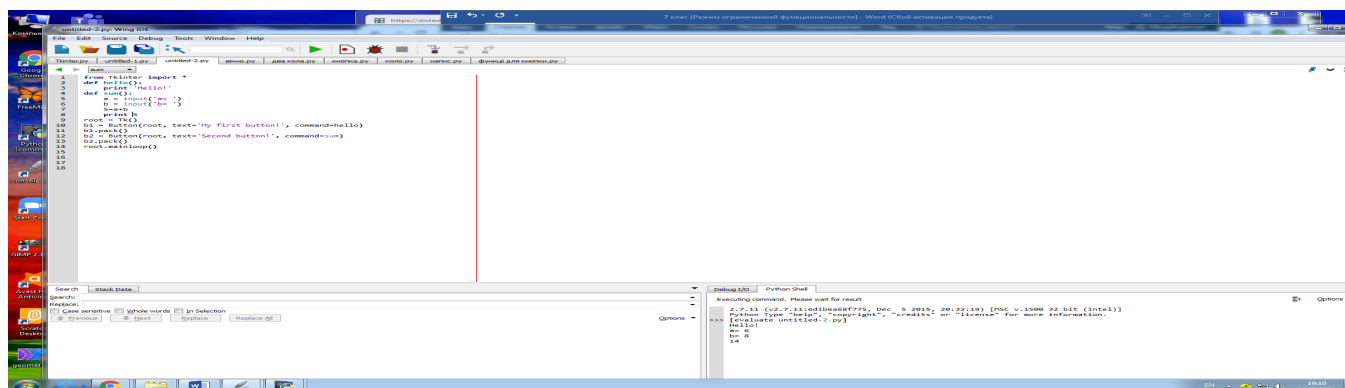
Результат запуску програми і п'яти натискувань кнопки:



Розглянемо ще один приклад, у якому при натисненні на першу кнопку виводиться "Hello!", друга кнопка дозволяє обрахувати суму двох чисел.

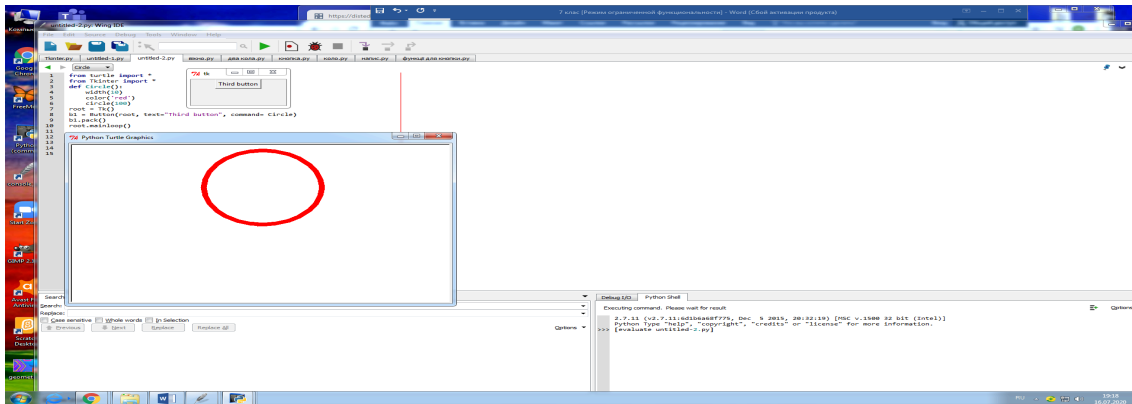
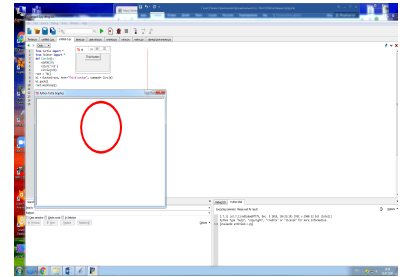


Результат запуску програми і натискування спочатку на першу, потім на другу кнопку (при цьому вводяться два числа і виводиться результат):



Використаємо бібліотеку **turtle** для задання функції малювання при натисненні на кнопку. Нехай при цьому в окремому вікні малюється коло.

При підключенні модуль **turtle** повинен обов'язково йти перед модулем Tkinter, бо вікно turtle написане за допомогою Tkinter.

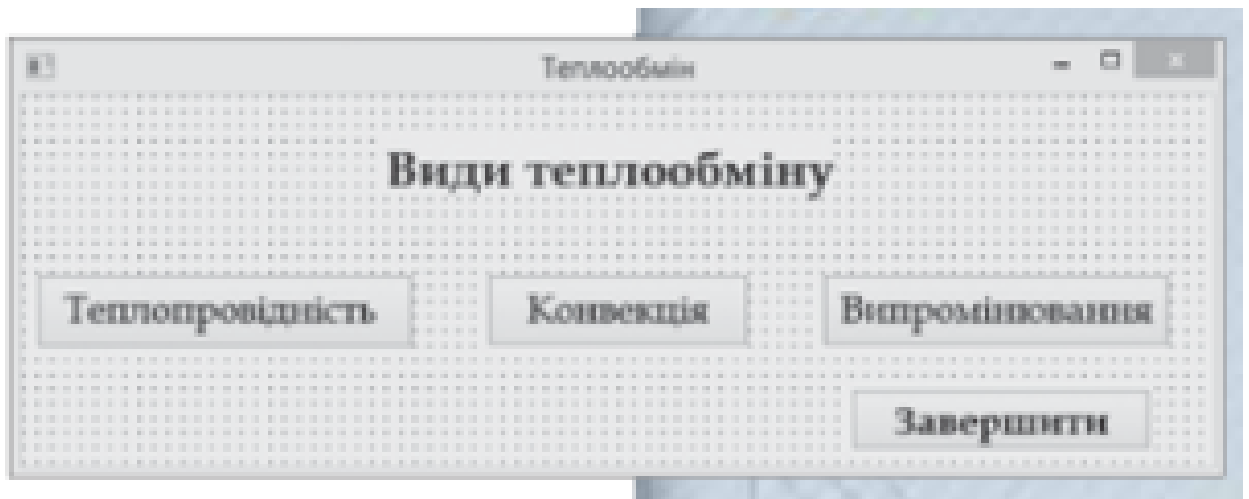


Завдання для самостійного виконання

1. Яка команда відповідає за введення функції?
 - a. Button
 - b. def
 - c. pack
 - d. command
2. Записати програму, яка створює у вікні дві кнопки різних кольорів, одна з яких рахує суму чисел, а інша – різницю.
3. Записати програму, яка створює у вікні дві кнопки різних кольорів, одна з яких малює коло довільного радіуса, а інша – трикутник.
4. Записати програму, яка створює у вікні дві кнопки різних кольорів, одна з яких малює квадрат довільного кольору і розмірів, а інша – рахує добуток трьох чисел.
5. Створити вікно з 4-ма кнопками однакового розміру. При натисненні на першу у вікні turtle черепашка малює червоний квадрат, при натисненні на другу - черепашка малює зелений трикутник, третя кнопка дозволяє черепашці намалювати жовтий круг, четверта - пурпуровий шестикутник.

Творче завдання

Довідник з фізики. Розроби довідник з фізики таким чином, щоб кожна кнопка викликала вікно з необхідною інформацією.



Вікно повинно містити напис «Види теплообміну» та три кнопки: Теплопровідність, Конвекція, Випромінювання. При натисненні на кожну кнопку відкривається вікно з текстом по даній темі.

§ 20. Властивості та методи елементів керування Список.

Для створення списків використовується клас **Listbox**. Це віджет, який має вигляд списку, з елементів якого користувач може вибирати один чи кілька пунктів. В Tkinter спочатку створюється `listbox`, після цього він заповнюється елементами за допомогою команди *list* і реалізується за допомогою команди *insert*:

listbox.insert(END,i)

Перший аргумент вказує на індекс місця, куди буде вставлено елемент. Якщо елемент буде вставлено в кінець списку, то індекс позначається *END*. Другий аргумент відповідає за елемент, що вставляється.

За замовчуванням в **Listbox** клацанням мишки можна вибрати лише один елемент.

Якщо необхідно забезпечити вибір одного елемента зі списку, то для властивості *selectmode* встановлюється значення **SINGLE**:

selectmode=SINGLE

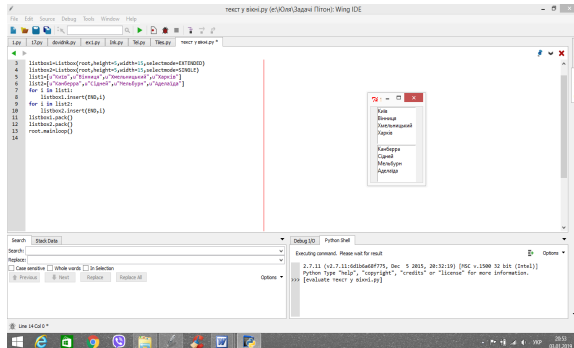
Якщо необхідно забезпечити вибір декількох елементів, то для властивості *selectmode*

можна встановити значення **EXTENDED**:

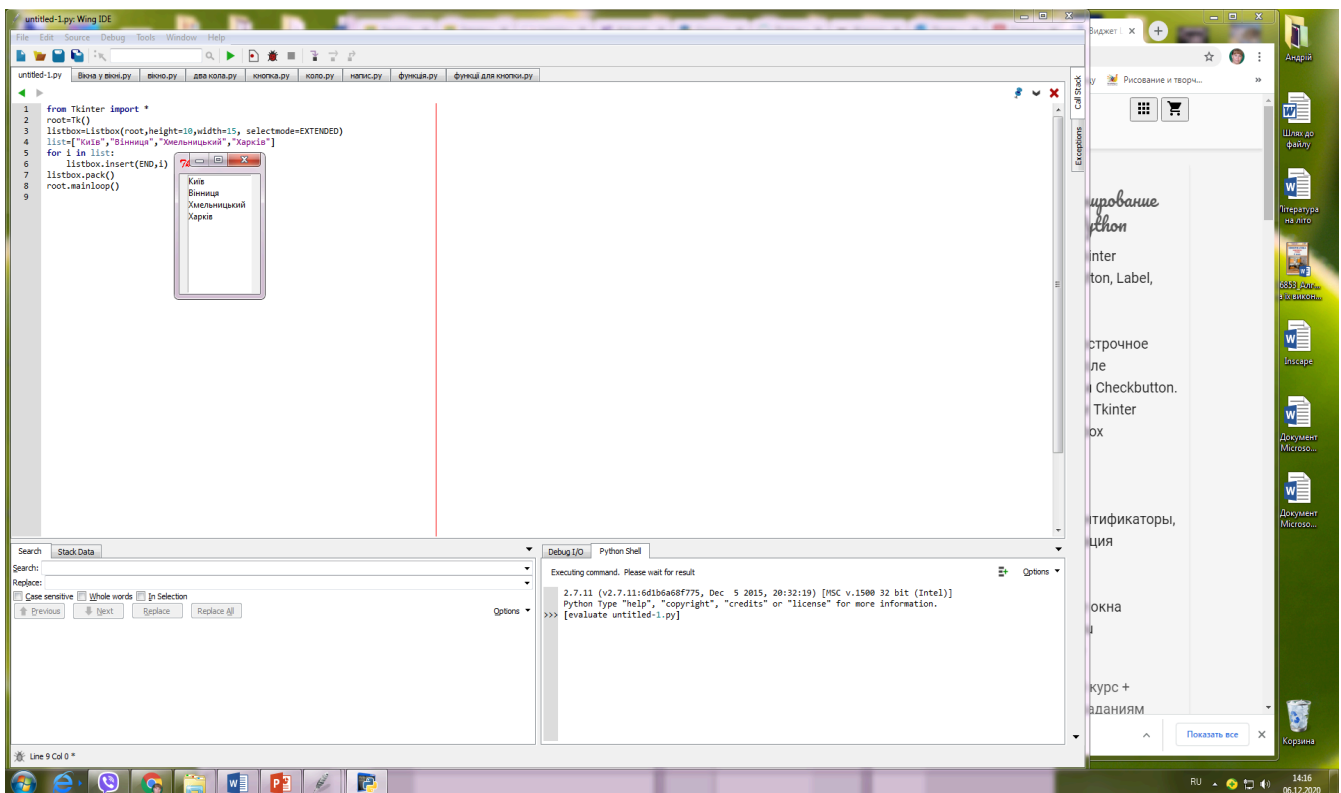
selectmode=EXTENDED

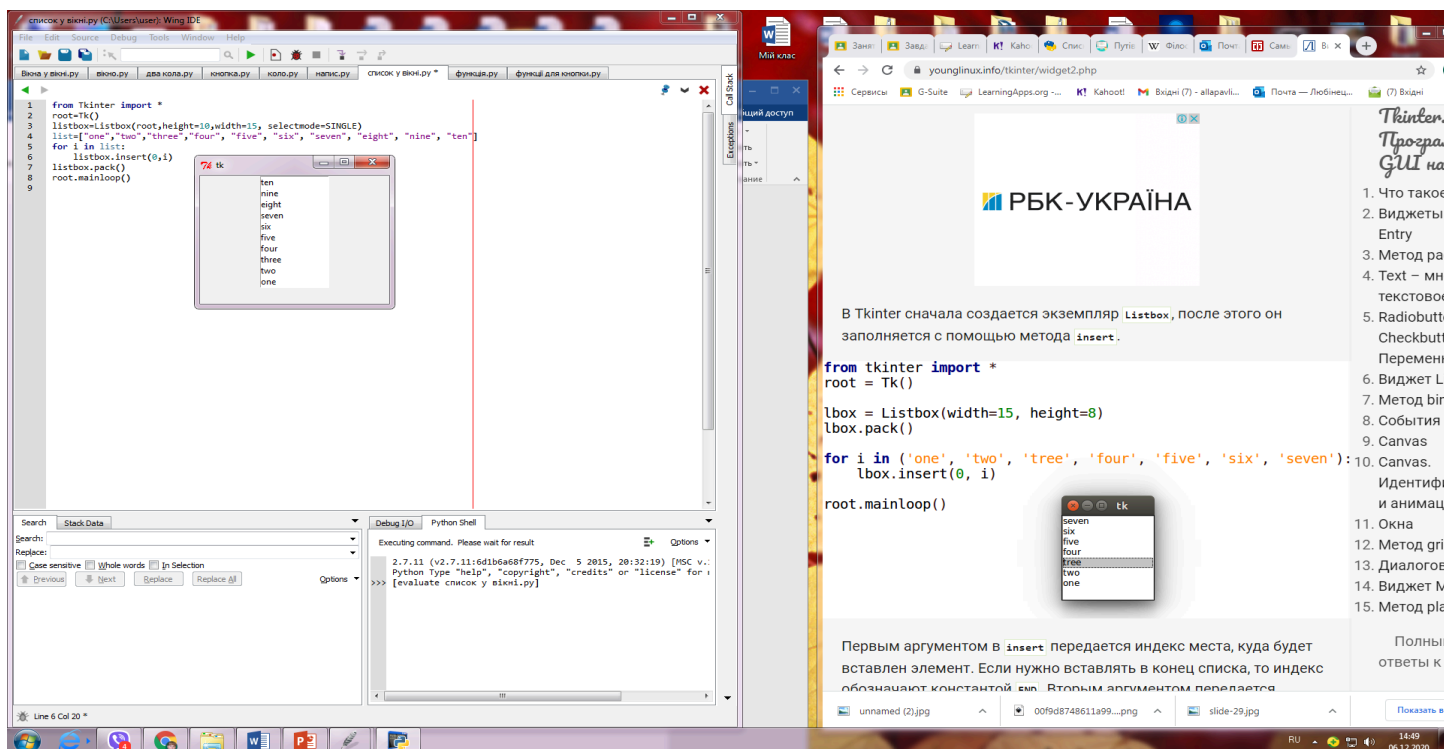
В цьому режимі можна вибрати скільки завгодно елементів, затиснувши клавіші **Ctrl** або **Shift**.

В Python 2.x для того, щоб відображалися українські букви, слід перед стрічкою поставити букву **u**.



Наприклад:





Завдання для самостійного виконання

1. Описати значення параметрів у вказаних рядках програми:

listbox1=Listbox(root,height=5,width=15,selectmode= SINGLE)

root	
height=5	
width=15	
selectmode= SINGLE	

2. Написати програму для створення списку свого класу, де можна обрати старосту (Староста в класі один, тобто вибрати можна одне значення зі списку).
3. Написати програму для вибору із переліку навчальних предметів своїх улюблених (передбачити множинний вибір).

Творче завдання

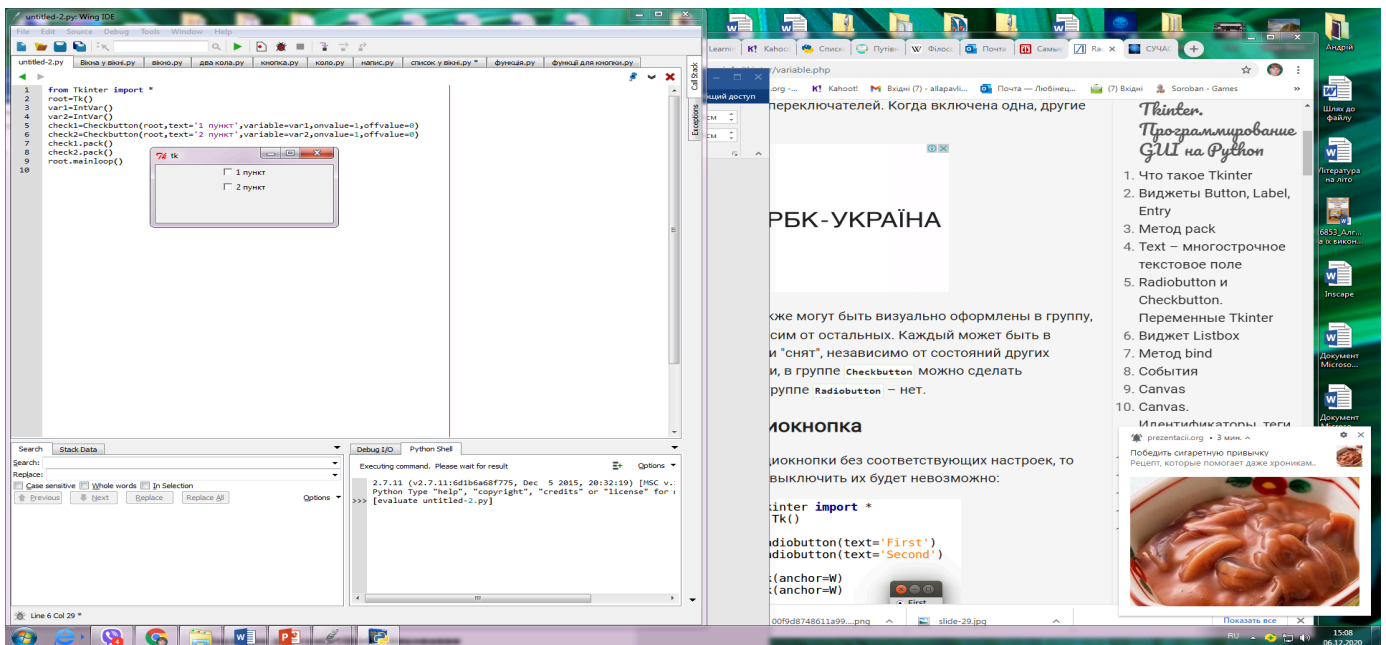
1. **Розклад уроків.** Розробіть проєкт, який містить кнопки з днями тижня. При натисненні на кнопку з'являється вікно зі списком уроків.
2. **Меню.** Розробіть проєкт, який передбачає вибір меню для різних категорій споживачів (наприклад, вегетаріанці, діабетики, різні види дієт тощо).

§ 21. Властивості та методи елементів керування.

Прапорець

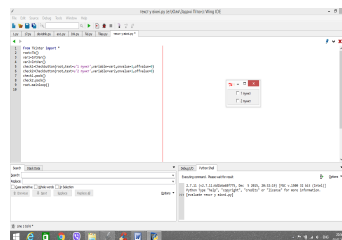
Для створення прапорців використовується клас **Checkbutton**. Це віджет, який дозволяє відмітити „галочкою“ певні пункти у вікні. Прапорці не зв'язані між собою, але змінні-константи тут потрібні, щоб знімати дані про стан прапорця. За значенням змінної, що зв'язана з прапорцем (**Checkbutton**), можна визначити: встановлено прапорець чи ні. Кожен прапорець повинен мати свою прикріплену змінну Tkinter.

Наприклад:



З допомогою опції *onvalue* встановлюється значення, якого набуває зв'язана з прапорцем змінна, коли прапорець включений (обраний).

З допомогою опції *offvalue* встановлюється значення, якого набуває зв'язана з прапорцем змінна, коли прапорець виключений (необраний).



Завдання для самостійного виконання

1. Описати значення параметрів у вказаних рядку програми:

check1=Checkbutton(root, text='1 пункт', variable=var1, onvalue=1, offvalue=0)

root

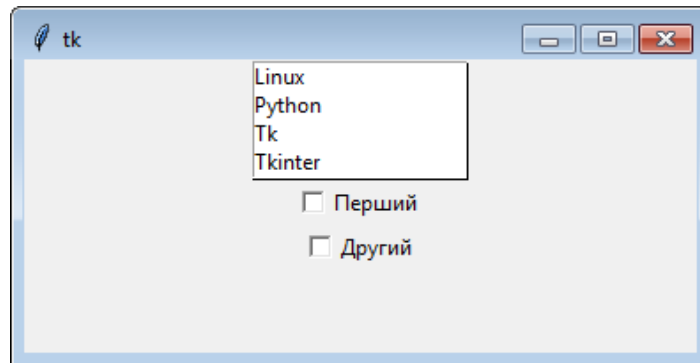
text='1 пункт'

variable=var1

onvalue=1

offvalue=0

2. Створити програму з елементами керування, щоб отримати таке вікно.



Творче завдання

Ключові компетентності. Створити вікно з текстом «Ключові компетентності для навчання впродовж життя». Використовуючи прапорці, вкажіть:

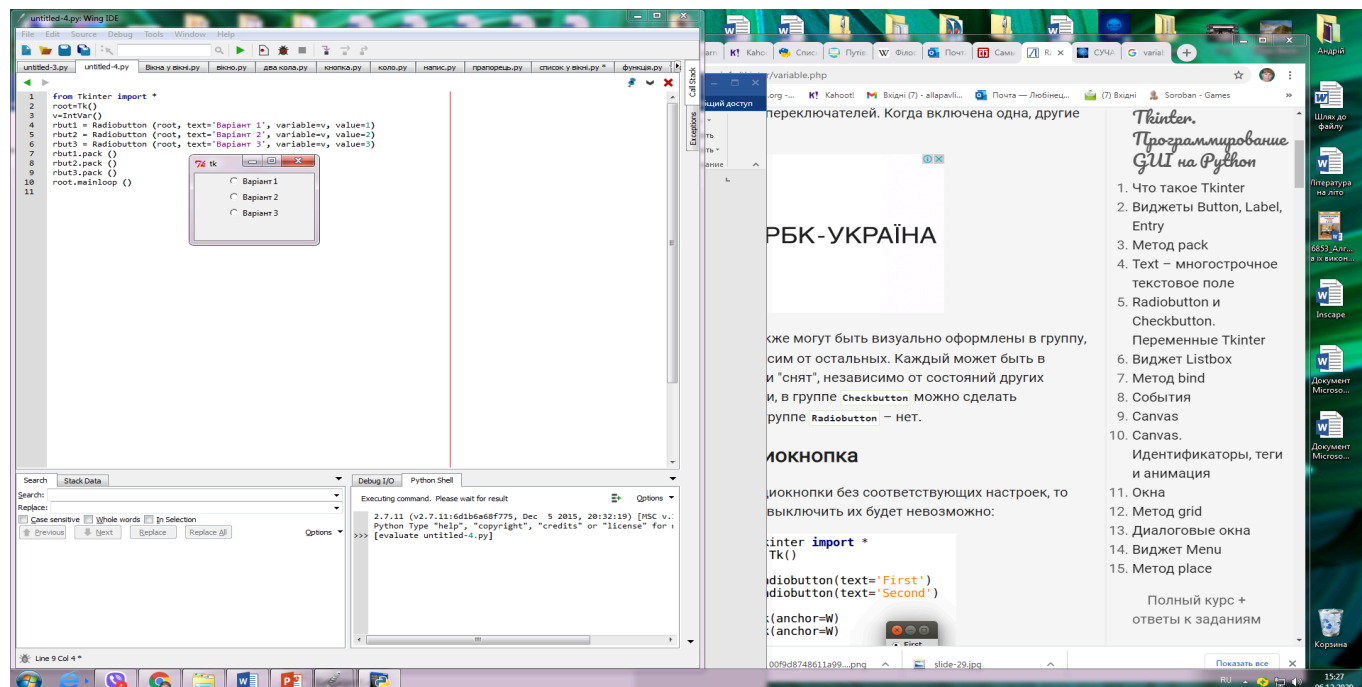
- ☐ Мовна грамотність
- ☐ Культурна обізнаність та самовираження
- ☐ Підприємницька
- ☐ Суспільна
- ☐ Особиста
- ☐ Соціальна
- ☐ Навчальна
- ☐ Цифрова

§ 22. Властивості та методи елементів керування. Перемикачі

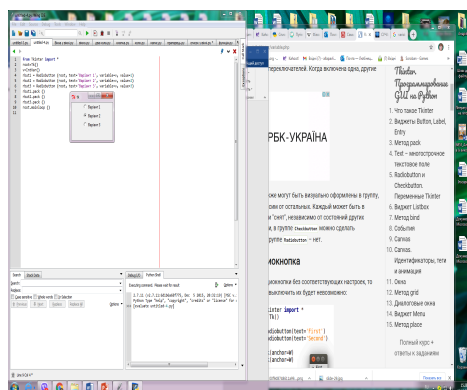
Для створення перемикачів використовується клас **Radiobutton**.

Перемикач — елемент списку з місцем для мітки за умови можливості виставити лише одну мітку. Перемикач завжди використовують у групі, причому увімкненим може бути лише один перемикач.

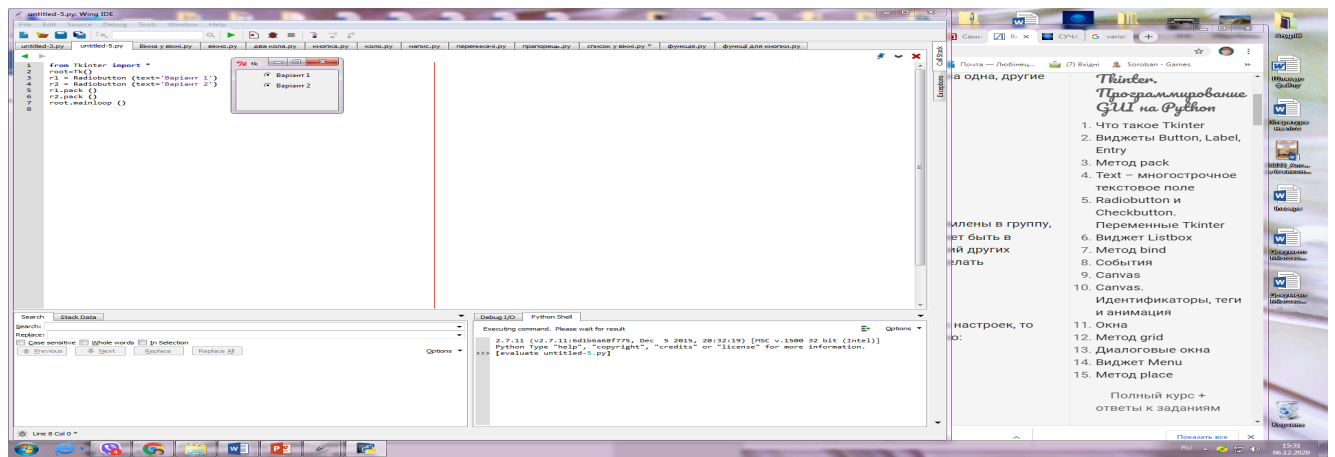
Реалізація цього віджета відрізняється від реалізації для Checkbutton:



У цьому прикладі використано лише одну змінну. Залежно від того, який пункт обрано, вона змінює своє значення.



Якщо буде створено два перемикача без відповідних атрибутів, то вони обидва будуть увімкнені і змінити щось буде неможливо:



Ці перемикачі ніяк не зв'язані між собою та за замовчуванням – обидва увімкнені. Зв'язок встановлюється через змінну, різні значення якої відповідають увімкненню різних перемикачів групи.

У всіх кнопок однієї групи атрибут *variable* описує прикріплення до віджета змінної, тобто набуває значення спільної для групи змінної. Атрибуту *value* присвоюються різні значення цієї змінної.

В Tkinter використовують лише спеціальні класи-змінні:

- ❖ BooleanVar – для булевих значень 0 або 1 та True або False,
- ❖ IntVar – для цілих значень,
- ❖ DoubleVar – для дійсних (дробових) значень,
- ❖ StringVar – для рядкових (стрічкових) значень.

Завдання для самостійного виконання

1. Встановити відповідність між класом та його назвою.

- | | |
|----------------|--------------|
| A. Radiobutton | 1. Рамка |
| Б. Checkbutton | 2. Перемикач |
| В. Frame | 3. Прапорець |
| Г. Listbox | 4. Список |

2. Описати значення параметрів у вказаному рядку програми:

Rbut3 = Radiobutton (root, text='Варіант 3', variable=v, value=3)

root	_____
text='Варіант 3'	—
variable=v	_____
value=3	—

	—

3. Створи вікно для вибору одного міста для подорожі.

Творче завдання

Тест про вчителя. Створи вікно «Тест про вчителя» та запитання з відповідями у ньому за зразком, використовуючи перемикачі.

- В учителя все повинно бути прекрасним: душа, тіло і.....
 - о Почерк
 - о Журнал
- Коли вчитель каже "одна година", він має на увазі....
 - о Урок
 - о Годину
- Справжній керівник - це ... керівник
 - о Чуйний
 - о Класний
- Раніше їх було всього 5, а тепер – 12.
 - о Бали
 - о Пальці

Додай кнопку, яка буде видавати сповіщення про закінчення тестування.

§ 23. Властивості та методи елементів керування.

Полотно для малювання

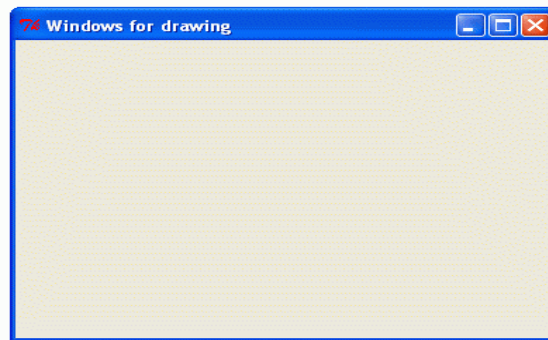
Полотно (Canvas) - це прямокутна область, призначена для малювання картинок або інших складних макетів. На полотні можна розміщувати графіку, текст, віджети або кадри.

Для створення полотна використовується клас **Canvas**. Цей клас застосовується для відображення діаграм, багатокутників та інших типів графічних елементів.

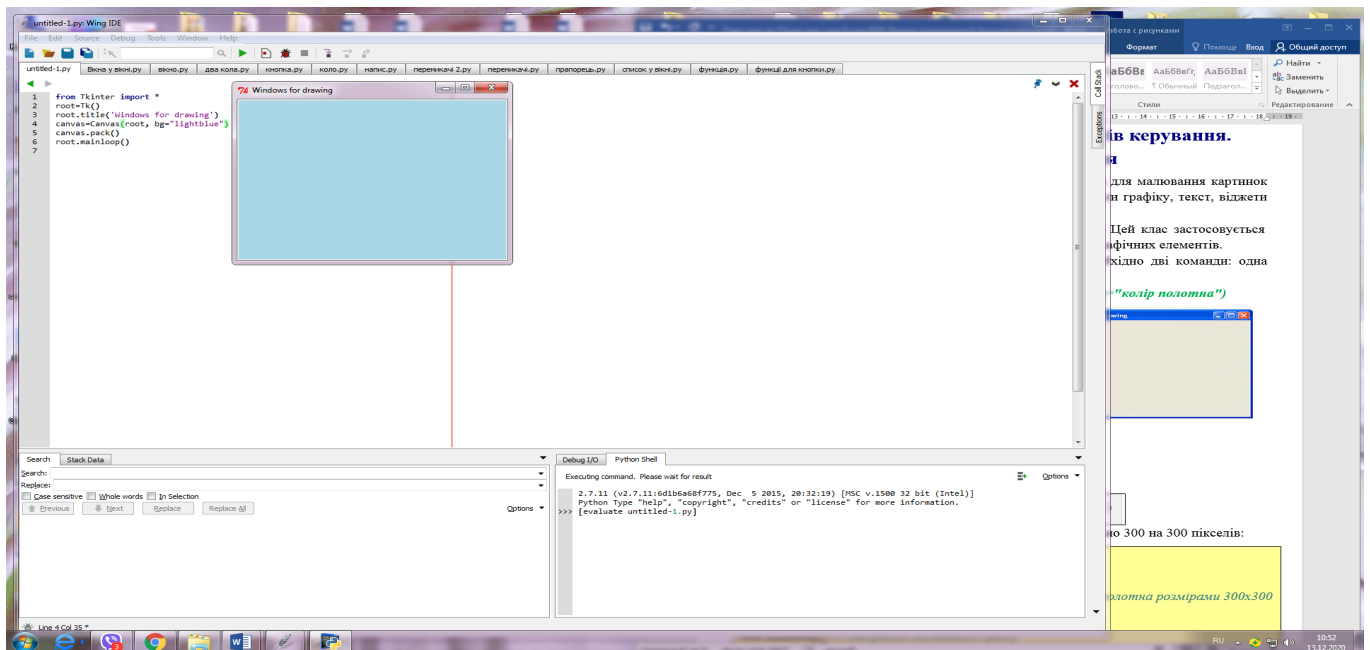
Для того, щоб зробити полотно для малювання, необхідно дві команди: одна створює полотно, друга його розміщує:

назва_полотна=Canvas (до якого вікна відноситься, bg="колір полотна")

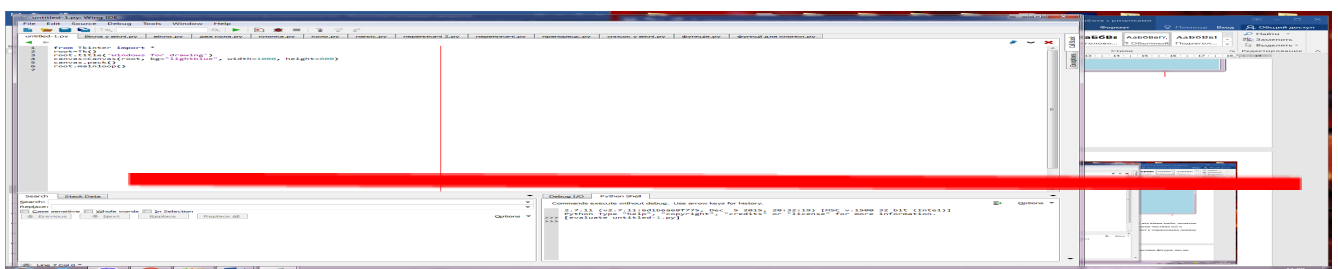
назва_полотна.pack()



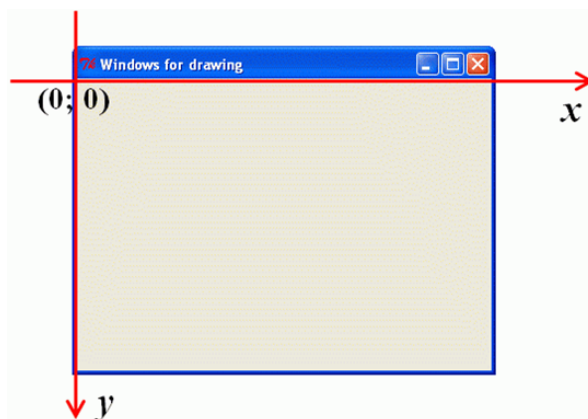
Програма для створення полотна має вигляд:



Для полотна можна задавати розміри (ширину та висоту) і колір.



На цьому полотні є система координат, але на відміну від вікна turtle, початок координат знаходиться завжди у лівому верхньому куті, додатна частина осі x спрямована звичайним чином вправо, а от додатна частина вісі y у спрямована донизу (!). Тобто у вікні видно лише першу координатну чверть - частина фігури, що не потрапляє у цю чверть просто не відображається.

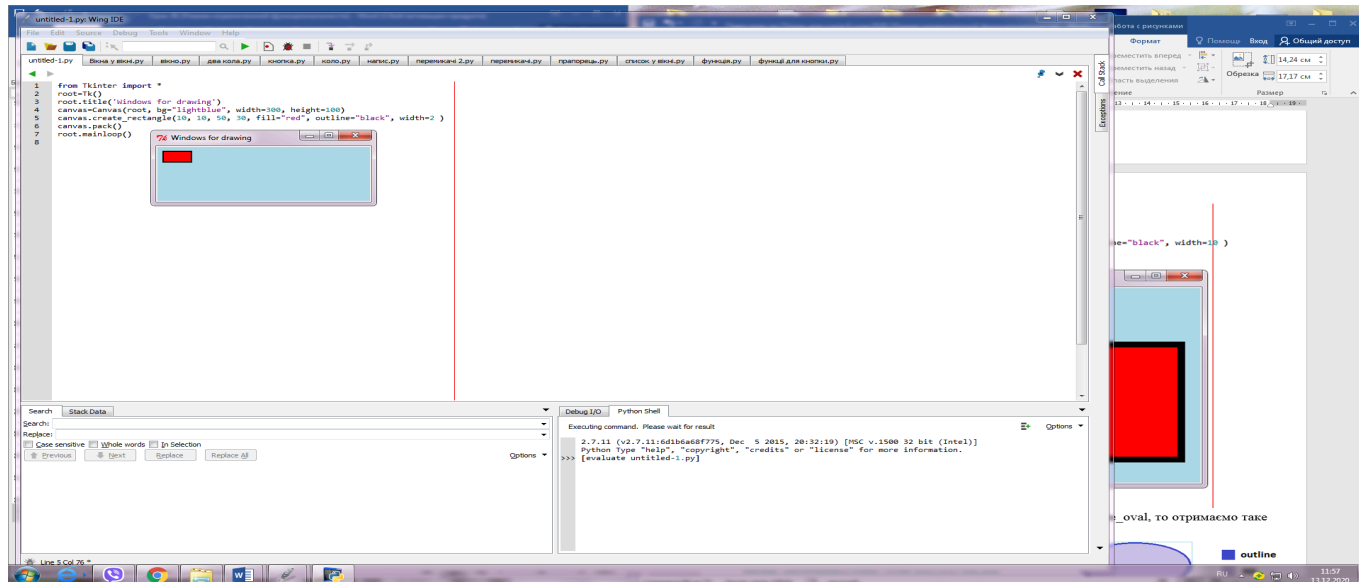
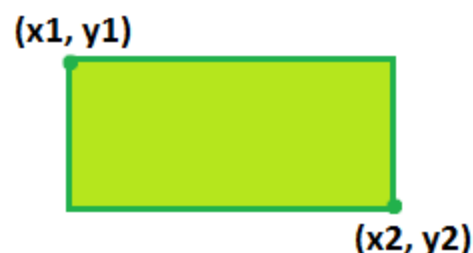


На полотно можна додавати малюнки: точки, лінії, кола, овали, прямокутники, багатокутники.

Для розміщення на полотні прямокутника застосуємо команду `create_rectangle`.

Прямокутник задається координатами двох несусідніх вершин, наприклад, лівого верхнього та правого нижнього кутів.

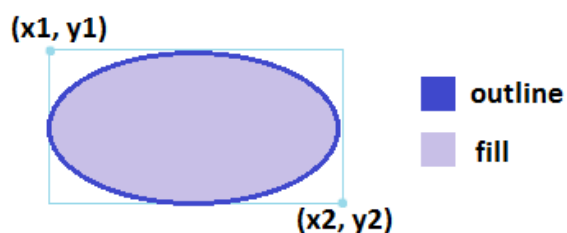
назва_полотна.create_rectangle(x1, y1, x2, y2, fill="колір прямок.", outline="колір лінії", width=товщина лінії),
де x1, y1, x2, y2 - координати двох крайніх точок прямокутника.

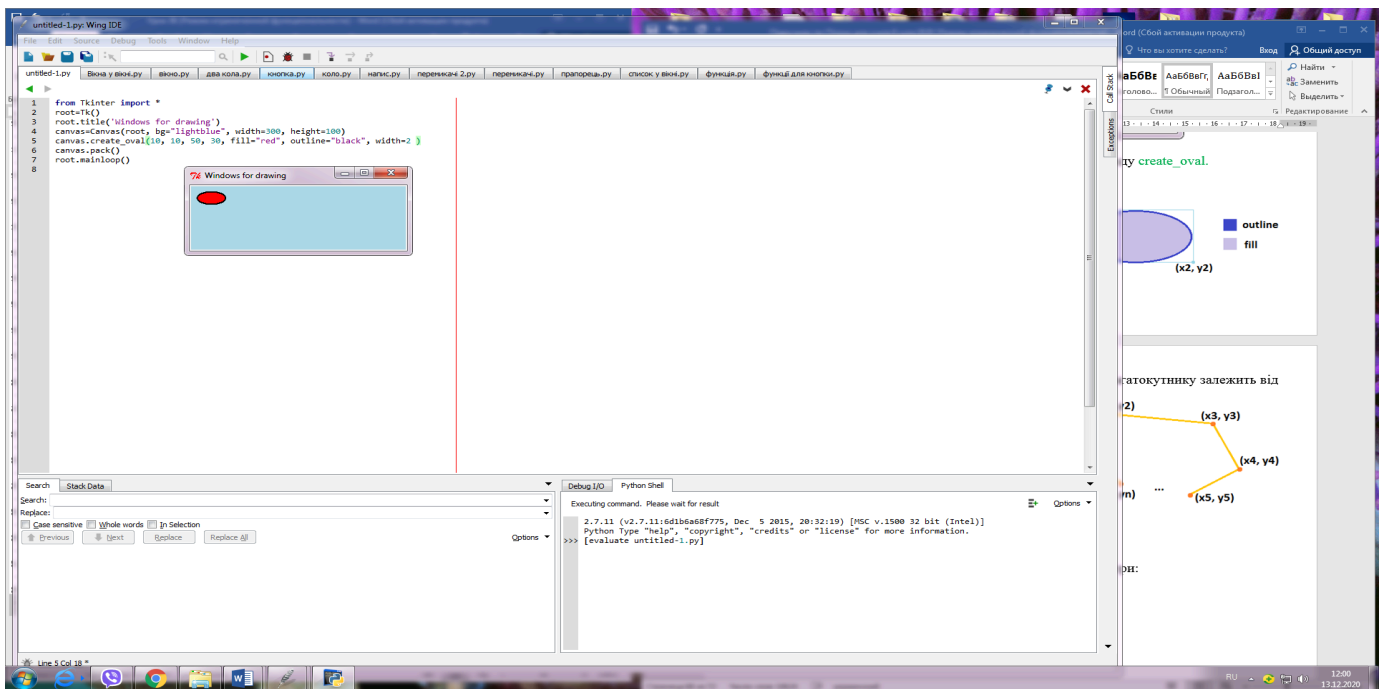


Для розміщення на полотні прямокутника застосуємо команду `create_oval`.

назва_полотна.create_oval(x1, y1, x2, y2, fill="колір овалу", outline="колір лінії", width="товщина лінії")

де x1, y1, x2, y2 - крайні точки уявного прямокутника.



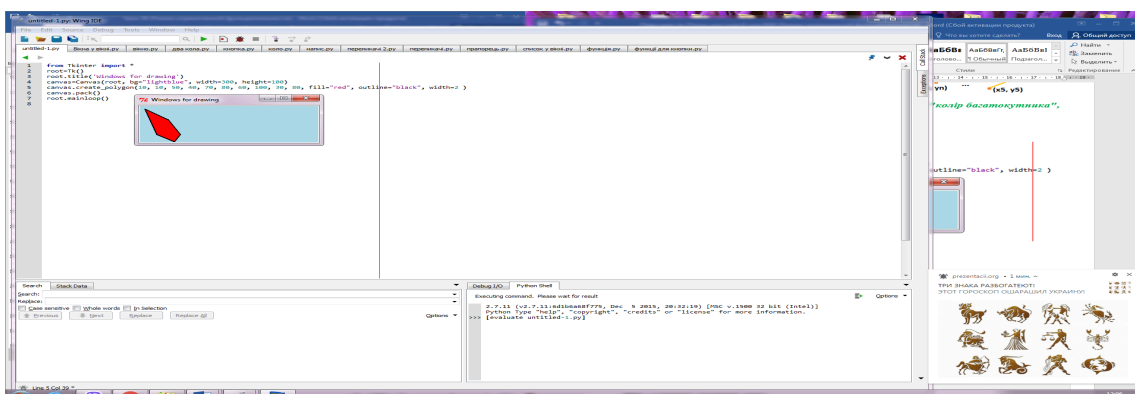
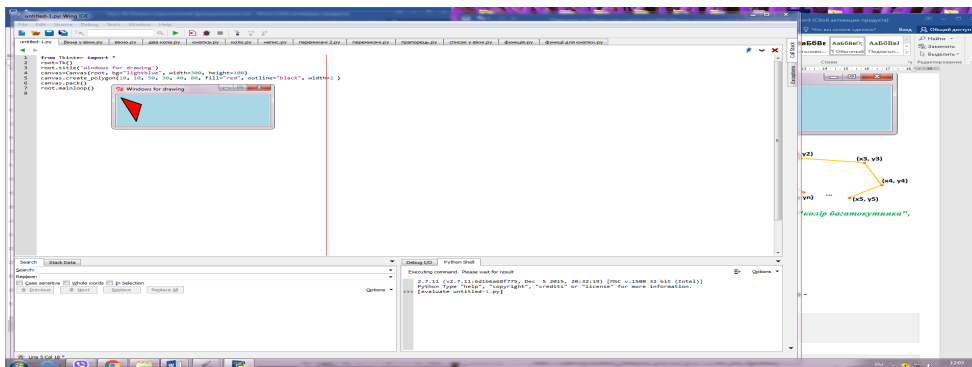
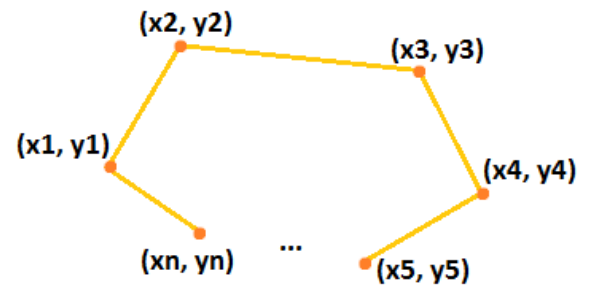


Також можна побудувати багатокутник. Кількість кутів у багатокутнику залежить від того, скільки координат точок кутів ми введемо.

Для побудови багатокутника використовуються команда:

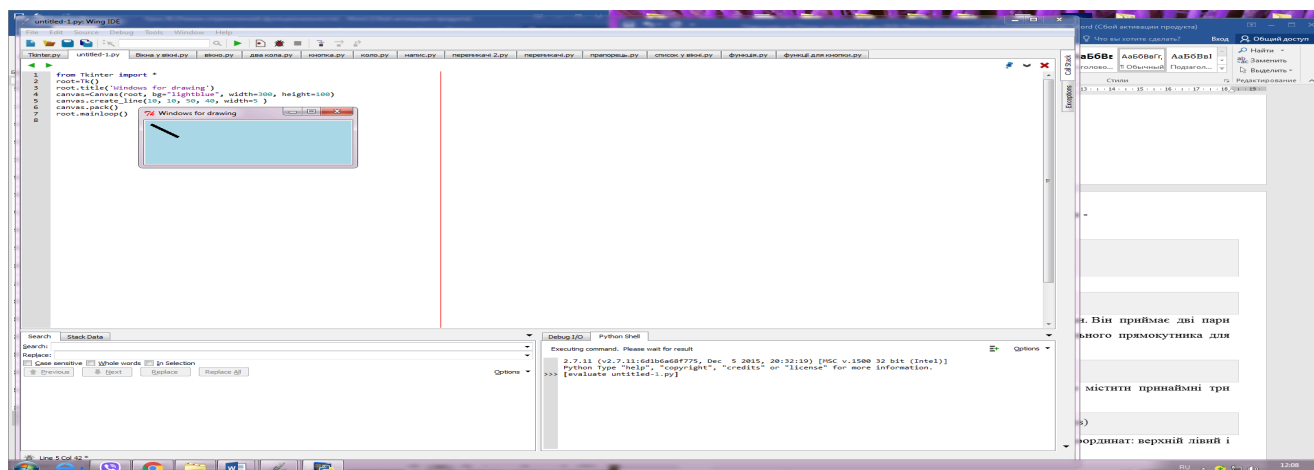
назва_полотна.create_polygon(x1, y1, x2, y2, ..., xn, yn, fill="колір багатокутника", outline="колір лінії", width="товщина лінії"),

де $x1, y1, x2, y2, \dots, xn, yn$ - координат точок кутів



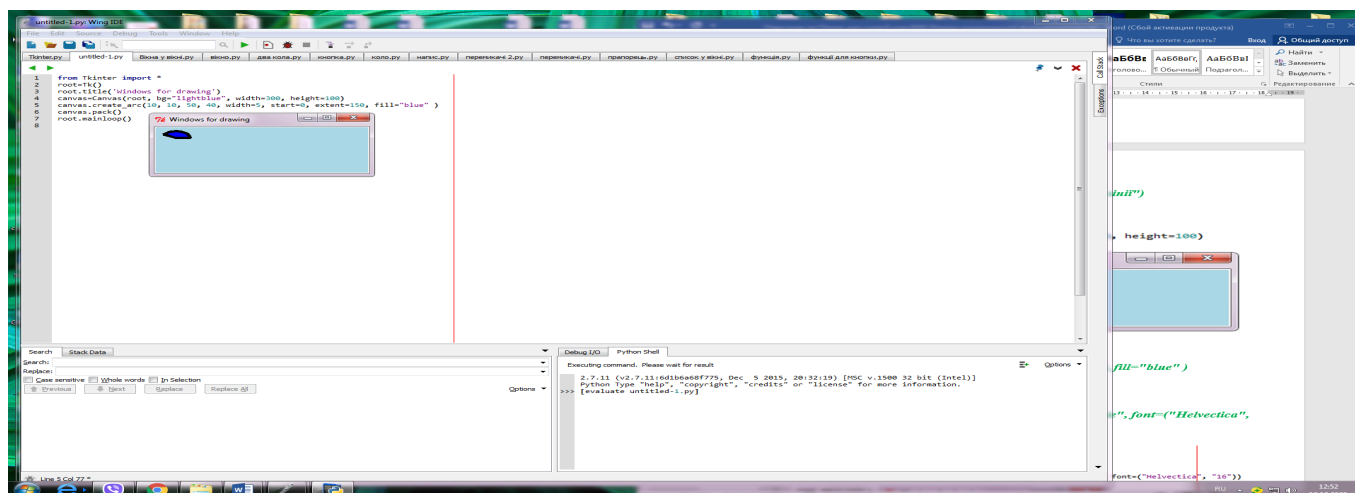
Для побудови лінії використовуються команда:

назва_полотна.create_line(x1, y1, x2, y2, width="товщина лінії")



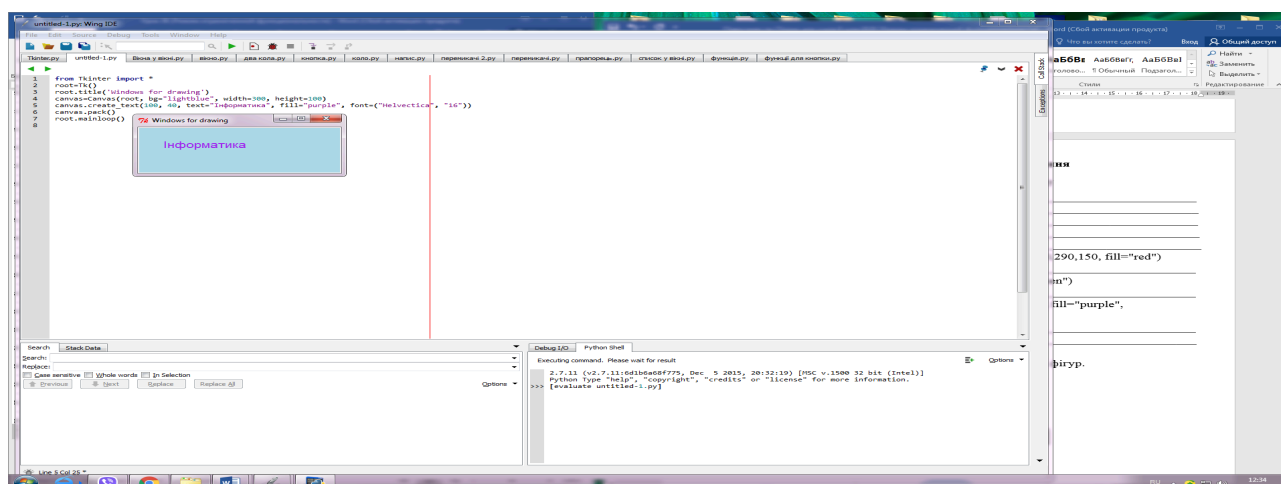
Для створення дуги використовується команда:

canvas.create_arc(10, 10, 50, 40, width=5, start=0, extent=150, fill="blue")



Для створення дуги використовується команда:

canvas.create_text(100, 40, text="Інформатика", fill="purple", font=("Helvetica", "16"))



Завдання для самостійного виконання

1. Запишіть, що означають стрічки вказаного нижче коду

```
from Tkinter import * _____  
root = Tk() _____  
canvas = Canvas(root, width=300, height=300) _____  
canvas.pack() _____  
canvas = canvas.create_oval(10,10,290,290, fill="#00f") _____  
diamond = canvas.create_polygon(150,10,10,150,150,290,290,150, fill="red")  
_____  
square = canvas.create_rectangle(80,80,220,220, fill="green")  
_____  
text = canvas.create_text(150,150, text="Tkinter canvas", fill="purple",  
font=("Helvetica", "16"))  
_____  
root.mainloop() _____
```

2. Створи програми для відображення усіх розглянутих фігур.

Творче завдання

3. Створи на полотні подібне зображення:

