**Лабораторна робота №1**

**Тема. Робота з Python. Завантаження програми. Робота в інтерактивному режимі. Цикли.**

**Мета роботи**. Отримати навички роботи в інтерактивному режимі.

**Зміст**.

1. Вивчення відомостей про роботу в інтерактивному режимі.

2. Виконання роботи.

3. Отримання результату.

**Ключові положення.**

**Що вміє робити python:**

• Робота с xml/html файлами  
• Робота с http запитами  
• GUI (графічний інтерфейс)  
• Створення веб-сценарієв  
• Робота с FTP  
• Робота з зображеннями, аудіо та відео файлами  
• Робототехніка  
• Програмування математичних та наукових обчислень  
Та багато іншого...

**Python можно загрузити з веб-сайту https://python.org/downloads/windows/, обираємо python 3.**

Після завантаження та установки **python 3** відкриваємо IDLE Рис.1

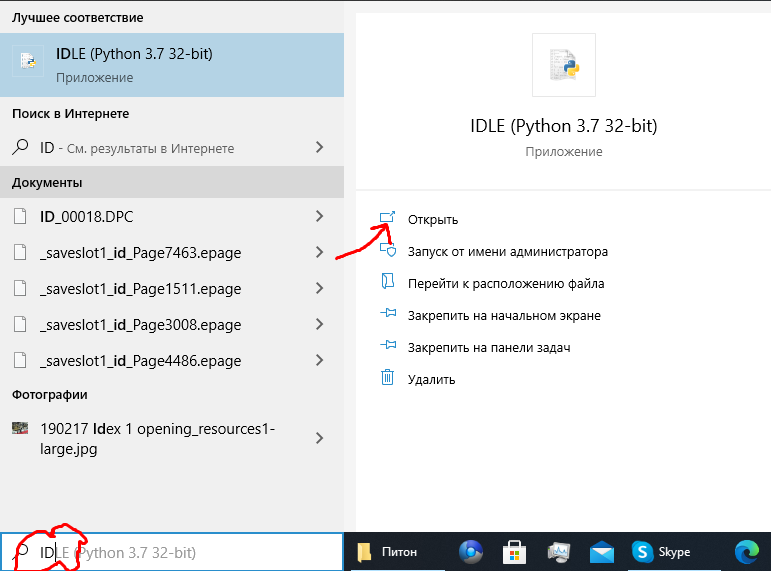


Рис.1

Щоб написати **“hello world”** на python достатньо усього однієї строки print("Hello world") та натиснемо **Enter.** Рис.2

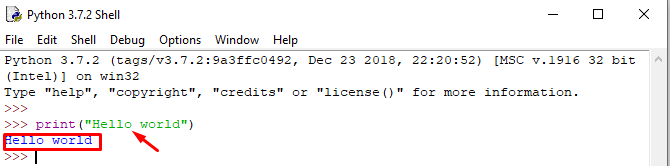


Рис.2

Інтерактивний режим не буде основним. В основному ви будете зберігати програмний код в файл і запускати вже файл. Для того, щоб створити нове вікно в інтерактивному режимі IDLE, виберіть File → New File (або натисніть Ctrl + N). Рис.3

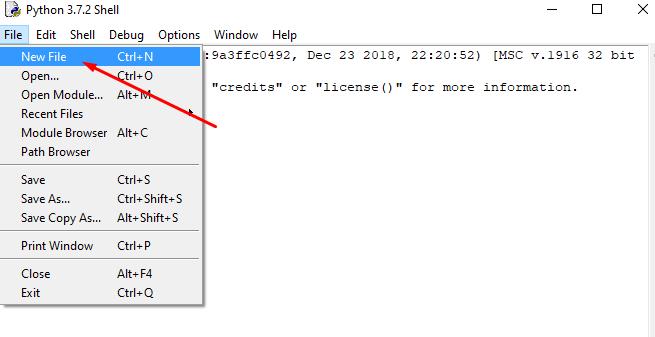


Рис.3

У вікні введіть наступний код: name = input ("Як Вас звати?") Print ("Привіт,", name) Рис.5

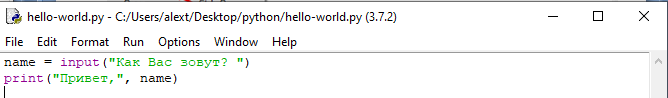


Рис.5

Тепер натиснемо F5 (або виберемо в меню IDLE Run → Run Module) і переконаємося, що программа, що ми написали, працює. Перед запуском IDLE запропонує нам зберегти файл. Збережемо туди, куди вам буде зручно, після чого програма запуститься.Рис.6,7

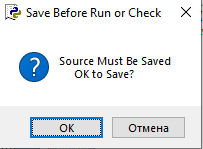


Рис.6

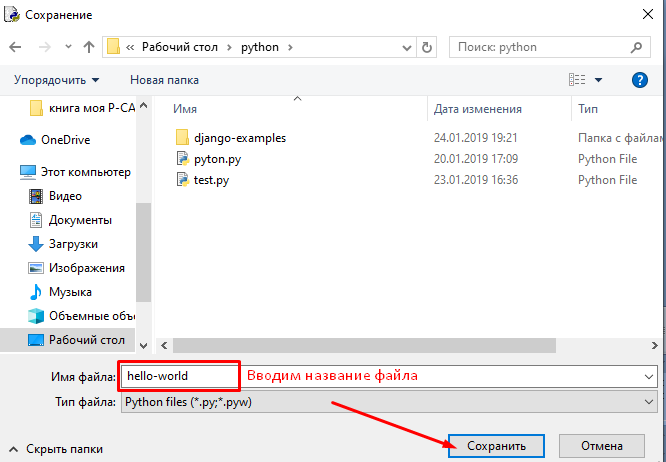


Рис.7

Перший рядок друкує питання ("Як Вас звати?") i очікує, поки ви не надрукуєте що-небудь і не натиснете Enter, і зберігає введене значення в змінній name. У другому рядку ми використовуємо функцію print для виведення тексту на екран, в даному випадку для виведення "Привіт," і того, що зберігається в змінної "name". Рис.8

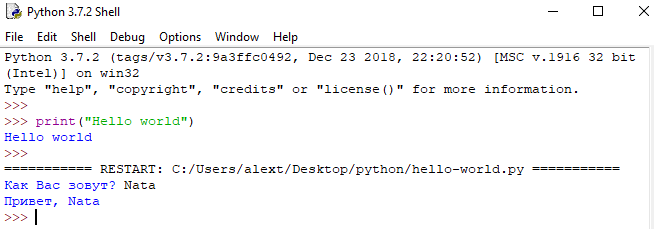


Рис.8

**Синтаксис**

1. Кінець рядка є кінцем інструкції (крапка з комою не потрібна).
2. Вкладені інструкції об'єднуються в блоки по величині відступів. Відступ може бути будь-яким, головне, щоб в межах одного вкладеного блоку відступ був однаковий. І про читабельність коду не забувайте. Відступ в 1 пробіл, наприклад, не найкраще рішення. Використовуйте 4 пробілу (або знак табуляції, на худий кінець).
3. Вкладені інструкції в Python записуються відповідно до одним і тим же шаблоном, коли основна інструкція завершується двокрапкою, слідом за яким розташовується вкладений блок коду, зазвичай з відступом під рядком основний інструкції.

**Введення і виведення даних**

Введення даних здійснюється за допомогою команди input (список введення):

a = input ()

print (a)

У дужках функції можна вказати повідомлення - коментар до даних які вводяться:

a = input ( "Введіть кількість:")

Команда input () за замовчуванням сприймає вхідні дані як рядок символів. Тому, щоб ввести цілочисельне значення, слід вказати тип даних int ():

a = int (input ())

Для введення дійсних чисел застосовується команда

a = float (input ())

Вивід даних здійснюється за допомогою команди print (список виведення):

a = 1

b = 2

print (a)

print (a + b)

print ( 'сума =', a + b)

Існує можливість запису команд в один рядок, розділяючи їх через **;** . Однак не слід часто використовувати такий спосіб, це знижує читабельність:

a = 1; b = 2; print (a)

print (a + b)

print ( 'сума =', a + b)

Для команди print може додаватися так званий сепаратор - роздільник між елементами виведення:

x = 2

y = 5

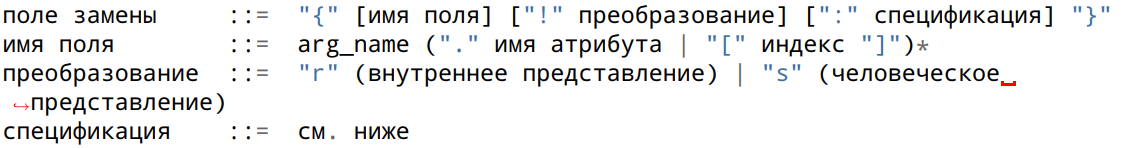
print (x, "+", y, "=", x + y, sep = "")

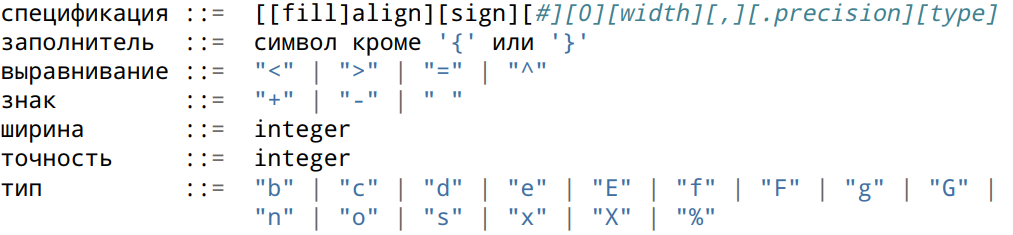
Результат відобразиться з пробілами між елементами: 2 + 5 = 7

Для форматованого виведення використовується **format**:

Строковий метод format () повертає відформатовану версію рядка, замінюючи ідентифікатори в фігурних дужках {}. Ідентифікатори можуть бути позиційними, числовими індексами, ключами словників, іменами змінних.

**Синтаксис команди format:**

****



Аргументів на format () може бути більше, ніж ідентифікаторів в рядку. В такому випадку ті які залишилися - ігноруються.

Ідентифікатори можуть бути або індексами аргументів, або ключами:

Наприклад: вивід числа 11 без форматування. Рис.9

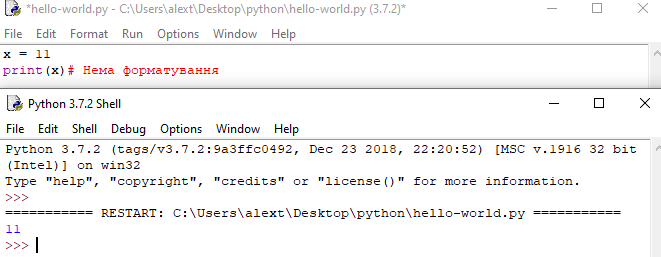


Рис.9

Наприклад: вивід числа 11 з форматуванням. Рис.10

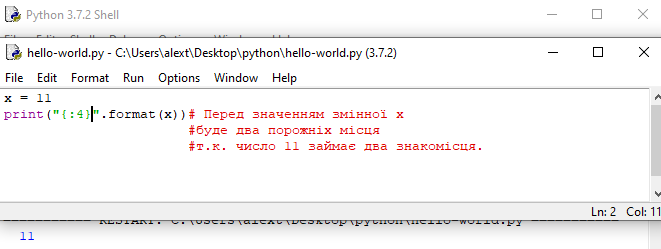


Рис.10

В результаті виведеться число 11, а перед ним два пробілу, так як було вказано використовувати для виведення чотири знакомісця.

Доступні наступні варіанті вирівнювання

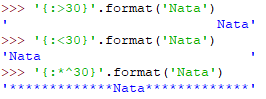
**'<'** Символи-наповнювачі будуть праворуч (вирівнювання об'єкта по лівому краю) (за замовчуванням).

**'>'** Вирівнювання об'єкта по правому краю.

**'='** Наповнювач буде після знака, але перед цифрами. Працює тільки з числовими типами.

**'^'** Вирівнювання по центру.

Наприклад:



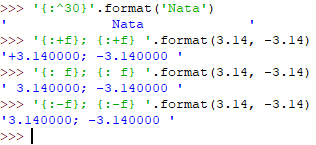
**Опція «Знак»**

**'+'** -Знак має бути надрукований для всіх чисел.

**'-' -**Минус для негативних, нічого для позитивних.

**'Пропуск'** - Минус для негативних, пробіл для позитивних.

Наприклад:



**Поле «Тип»** може приймати наступні значення:

'd','i', 'u' -Десяткове число.

'o' -Число в вісімковій системі числення.

'x' -Число в шістнадцятковій системі числення (букви в нижньому регістрі).

'X' -Число в шістнадцятковій системі числення (літери у верхньому регістрі).

'e' -Число з плаваючою точкою з експонентою (експонента в нижньому регістрі).

'E' -Число з плаваючою точкою з експонентою (експонента в верхньому регістрі).

'f', 'F' -Число з плаваючою крапкою (звичайний формат).

'g' -Число з плаваючою крапкою з експонентою (експонента в нижньому регістрі), якщо вона менше, ніж -4 або точності, інакше звичайний формат.

'G' -Число з плаваючою крапкою. з експонентою (експонента в верхньому регістрі), якщо вона менше, ніж -4 або точності, інакше звичайний формат.

'c' -Символ (рядок з одного символу або число - код символу).

's' -Строка.

'%' -Число множиться на 100, відображається число з плаваючою точкою, а за ним знак%.

Для форматування дійсних чисел з плаваючою точкою використовується наступна команда:

print ('{0: .2f}'. format (дійсне число))

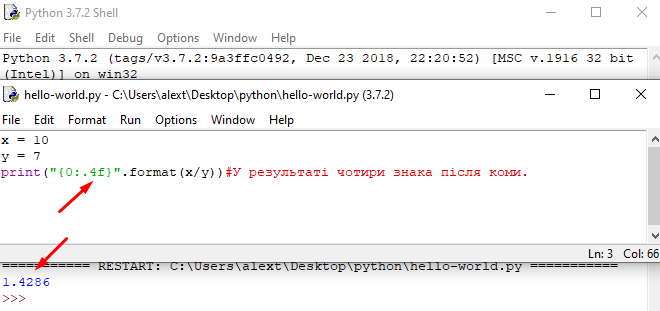


Рис.11

**Типи даних**

1. Цілі числа (тип int) - позитивні і негативні цілі числа, а також 0 (наприклад, 4, 687, -45, 0).
2. Числа з плаваючою точкою (тип float) - дробові, вони ж речові, числа (наприклад, 1.45, -3.789654, 0.00453). Примітка: для поділу цілої та дробової частин тут використовується точка, а не кома.
3. Строки (тип str) - набір символів, укладених в лапки (наприклад, "ball", "What is your name?", 'dkfjUUv', '6589'). Примітка: лапки в Python можуть бути одинарними або подвійними; одиночний символ в лапках також є рядком, окремого символьного типу в Пітоні немає.

**Дії над цілими числами**

**Табл.1**

|  |  |
| --- | --- |
| ^ | Зведення у ступінь |
| \* | Множення |
| / | Ділення |
| + | Додавання |
| - | Віднімання |
| // | Отримання цілої частини від ділення |
| % | Остача від ділення |
| -х | Зміна знаку числа |
| abs | Модуль числа |
| Divmod(x,y) | Пара(х//у,х%у) |
| Pow(x,y[,z]) | Ху по модулю ( якщо модуль надано) |

**Операції в програмуванні**

**Операція** - це виконання будь-яких дій над даними, які в даному випадку називають операндами. Саму дію виконує оператор - спеціальний інструмент.

Так в математиці і програмуванні символ плюса є оператором операції додавання по відношенню до чисел. У разі рядків цей же оператор виконує операцію конкатенації - з'єднання.

Наприклад:

* >>> 10.25 + 98.36
* 108.61
* >>> 'Hello' + 'World'
* 'HelloWorld'

**Бітові операції**

Над цілими числами також можна робити бітові операції табл.2.

**Табл.2**

|  |  |
| --- | --- |
| | | Побітове але |
| ^ | Побітове виключаючи але |
| & | Побітове і |
| << | Побітовий зсув вліво |
| >> | Побітовий зсув вправо |
| ~ | Інверсія бітів |

**Системи числення**

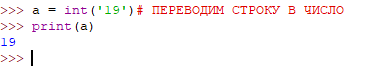
• int ([object], [підставу системи числення]) - перетворення до цілого числа в десятковій системі числення. За замовчуванням система числення десяткова, але можна задати будь-яку підставу від 2 до 36 включно.

• bin (x) - перетворення цілого числа в двійкову систему числення.

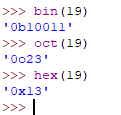
• hex (х) - перетворення цілого числа в шістнадцятирічну систему числення.

• oct (х) - перетворення цілого числа в восьмирічну систему числення.

Наприклад:







**Числа (float) – з плаваючою комою.**

Речові числа підтримують ті ж операції, що і цілі. Однак (через представлення чисел в комп'ютері) речові числа неточні, і це може привести до помилок:

>>> 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1

Виведе 0.9999999999999999

Для високої точності використовують інші об'єкти (наприклад Decimal і Fraction). Також речові числа не підтримують довгу арифметику.

Наприклад:

>>> a = 3 \*\* 1000  
>>> a + 0.1  
Traceback (most recent call last):  
File "", line 1, in  
OverflowError: int too large to convert to float

Наприклад:

>>> c = 150  
>>> d = 12.9  
>>> c + d  
162.9  
>>> p = abs(d - c) *# Модуль числа*>>> print(p)  
137.1  
>>> round(p) *# Округлення*137

Додаткові методи роботи з цілими числами:

**float.as\_integer\_ratio ()** - пара цілих чисел, чиє ставлення одно цього числа.

**float.is\_integer ()** - чи є значення цілим числом.

**float.hex ()** - переводить float в hex (шістнадцятирічну систему числення).

classmethod **float.fromhex (s)** - float з шістнадцяткового рядка.

Наприклад:





Модуль **math** надає більш складні математичні функції

Наприклад:

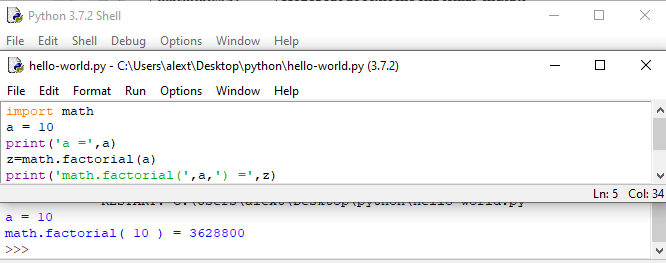
>>> import math  
>>> math.pi  
3.141592653589793  
>>> math.sqrt(85)  
9.219544457292887

**Бібліотека модуля** **math (деякі функції)**

Табл.3

|  |  |
| --- | --- |
| **math.ceil(x)** | Повертає найближче ціле число більше, ніж x |
| **math.fabs(x)** | Повертає абсолютне значення числа x |
| **math.factorial(x)** | Обчислює факторіал x |
| **math.floor(x)** | Повертає найближче ціле число менше, ніж x |
| **math.exp(x)** | Обчислює e\*\*x |
| **math.log2(x)** | Логарифм по основі 2 |
| **math.log10(x)** | Логарифм по основі 10 |
| **math.log(x[, base])** | За замовчуванням обчислює логарифм за основою e, додатково можна вказати основу логарифма |
| **math.pow(x, y)** | Обчислює значення x в ступені y |
| **math.sqrt(x)** | Корень квадратний від x |

Наприклад:



**Тригометричні функції модуля math**

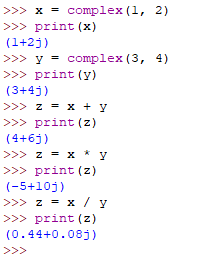
|  |  |
| --- | --- |
| **math.cos(x)** | Повертає cos числа х |
| **math.sin(x)** | Повертає sin числа х |
| **math.tan(x)** | Повертає tan числа х |
| **math.acos(x)** | Повертає acos числа х |
| **math.asin(x)** | Повертає asin числа х |
| **math.atan(x)** | Повертає atan числа х |

Модуль **random** реалізує генератор випадкових чисел і функції випадкового вибору.

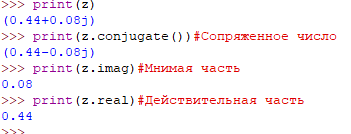
Наприклад:

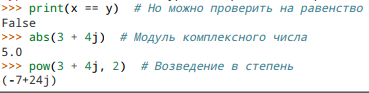
>>> import random  
>>> random.random()  
0.15651968855132303

Робота з комплексними числами у прикладах:



Комплексні числа не можна порівнювати!!!



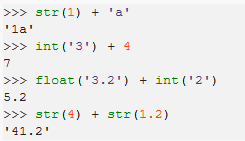


**Зміна типів даних**

Для зміни одних типів даних в інші в мові Python передбачений ряд вбудованих в нього функцій.

Ці функції перетворять те, що поміщається в їх дужки відповідно в ціле число, дійсне число або рядок.

Наприклад:



**Правила написання імен змінних**

1.Бажано давати змінним осмислені імена, що говорять про призначення даних, на які вони посилаються.

2.Ім’я змінної не повинно збігатися з командами мови (зарезервованими ключовими словами).

3.Ім’я змінної має починатися з букви або символу підкреслення (\_), але не з цифри.

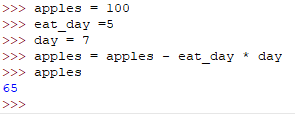
4.Ім’я змінної не повинно містити пробіли.

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

Наприклад:



Приклад роботи зі змінними в інтерактивному режимі:



**Синтаксис інструкції if**

Спочатку записується частина if з умовним виразом, далі можуть слідувати одна або більше необов'язкових частин elif, і, нарешті, необов'язкова частина else. Загальна форма записи умовної інструкції if виглядає наступним чином:

if test1:  
 state1  
elif test2:  
 state2  
else:  
 state3

Наприклад:

a = int(input())  
if a < -5:  
 print('Low')  
elif -5 <= a <= 5:  
 print('Mid')  
else:  
 print('High')

**Перевірка істинності в Python**

• Будь-яке число, не рівне 0, або непорожній об'єкт - істина.

• Числа, рівні 0, порожні об'єкти і значення None - брехня

• Операції порівняння застосовуються до структур даних рекурсивно

• Операції порівняння повертають True або False

• Логічні оператори and і or повертають істинний або помилковий об'єкт-операнд

**Логічні оператори:**

X and Y

Істина, якщо обидва значення X і Y істинні.

X or Y

Істина, якщо хоча б одне зі значень X або Y істинно.

not X

Істина, якщо X ложно.

**Тримісний вираз if / else**

Наступна інструкція:

if X:

A = Y

else:

A = Z

Досить коротка, але, тим не менше, займає цілих 4 рядки. Спеціально для таких випадків і було придумано вираз if / else:

A = Y if X else Z

У даній інструкції інтерпретатор виконає вираз Y, якщо X істинно, в іншому випадку виконається вираз Z.

>>> A = 't' if 'spam' else 'f'

>>> A

't'

**Цикл while**

Виконує тіло циклу до тих пір, поки умова циклу істинно.

Наприклад:

>>> i = 5  
>>> while i < 15:  
... print(i)  
... i = i + 2  
...  
5

7

9  
11  
13

**Цикл for**

Цикл for вже трішки складніший, трохи менш універсальний, але виконується набагато швидше циклу while. Цей цикл проходиться по будь-якому ітеріруемому об'єкту (наприклад рядку або списку), і під час кожного проходу виконує тіло циклу.Рис.1

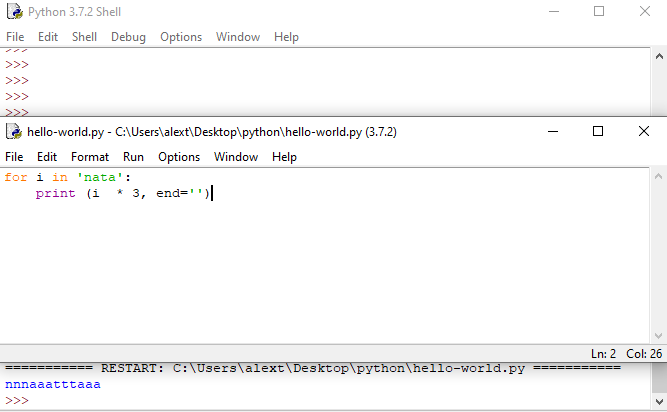


Рис.1

Без end = '' – розташовується у стовпчик. Рис.2

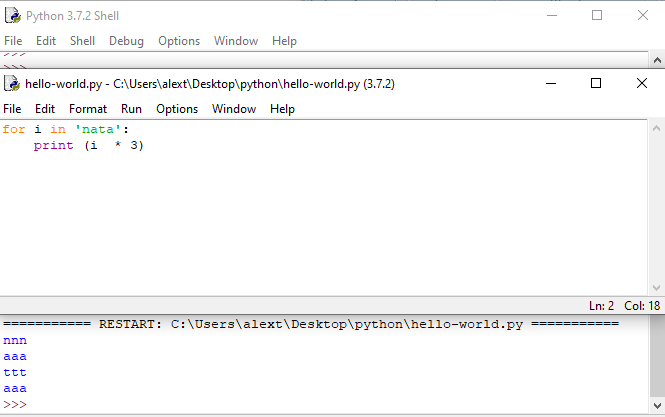


Рис.2

**Оператор break**

Достроково перериває цикл. Рис.3

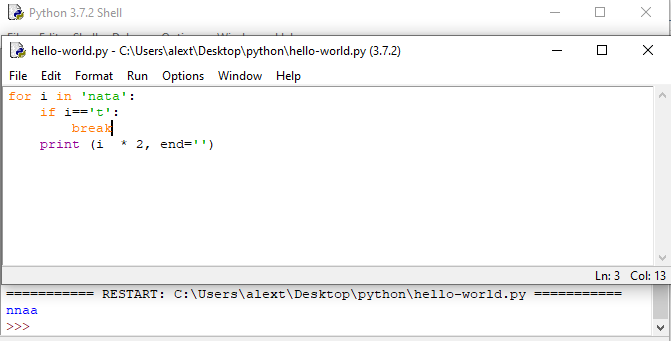


Рис.3

**Оператор continue**

Починає наступний прохід циклу, минаючи решту тіла циклу (for або while). Рис.4

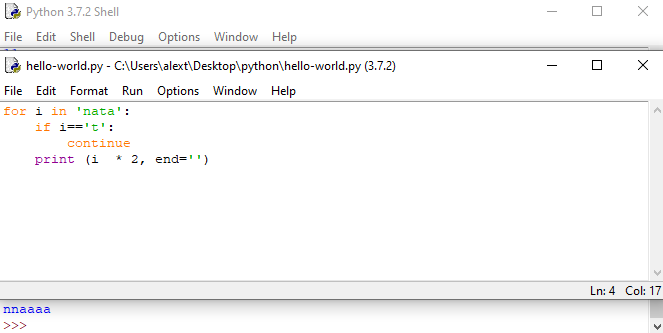


Рис.4

Слово **else**, застосоване в циклі for або while, перевіряє, чи був проведений вихід з циклу інструкцією break, або ж "природним" чином. Рис.5

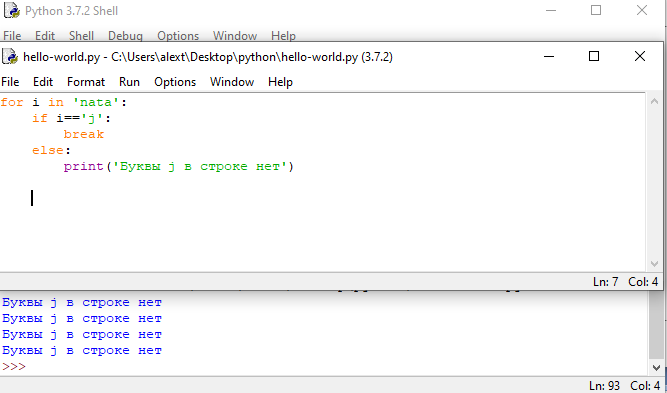


Рис.5

**Лабораторне завдання**

1. Напишіть програму, яка запитувала б у користувача:

- ПІБ ("Ваші прізвище, ім'я, по батькові?")

- вік ("Скільки Вам років?")

- місце проживання ("Де Ви живете?")

-де Ви навчаєтесь (“Де Ви навчаєтесь?”)

-номер Вашої групи( “Номер Вашої групи?”)

-порядковий номер по списку у группі(“Який Ваш порядковий номер у списку групи?”)

-***питання відповідно до варіанту***

Після цього виводила б рядки:

"Ваше ім'я"

"Ваш вік"

"Ви живете в"

“Ви навчаєтесь в”.

“Номер моєї групи -”

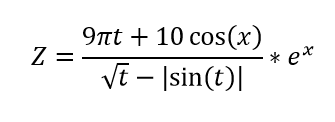
“Мій порядковий номер у списку групи-“

***«Ваш варіант відповіді»***

**№ Варіанту - остання цифра у списку групи.**

|  |  |
| --- | --- |
| № Варіанту | Питання |
| 0 | Як справи? |
| 1 | Як Ви себе почуваєте? |
| 2 | Коли будете вдома? |
| 3 | Яку оцінку отримав на ЗНО по українській мові? |
| 4 | Сьогодні сонячно? |
| 5 | Коли нарешті карантин? |
| 6 | Як звати Вашого друга? |
| 7 | Ви думаєте вступати у магістратуру? |
| 8 | Якого кольору Ваш зошит? |
| 9 | Який Ваш настрій сьогодні? |

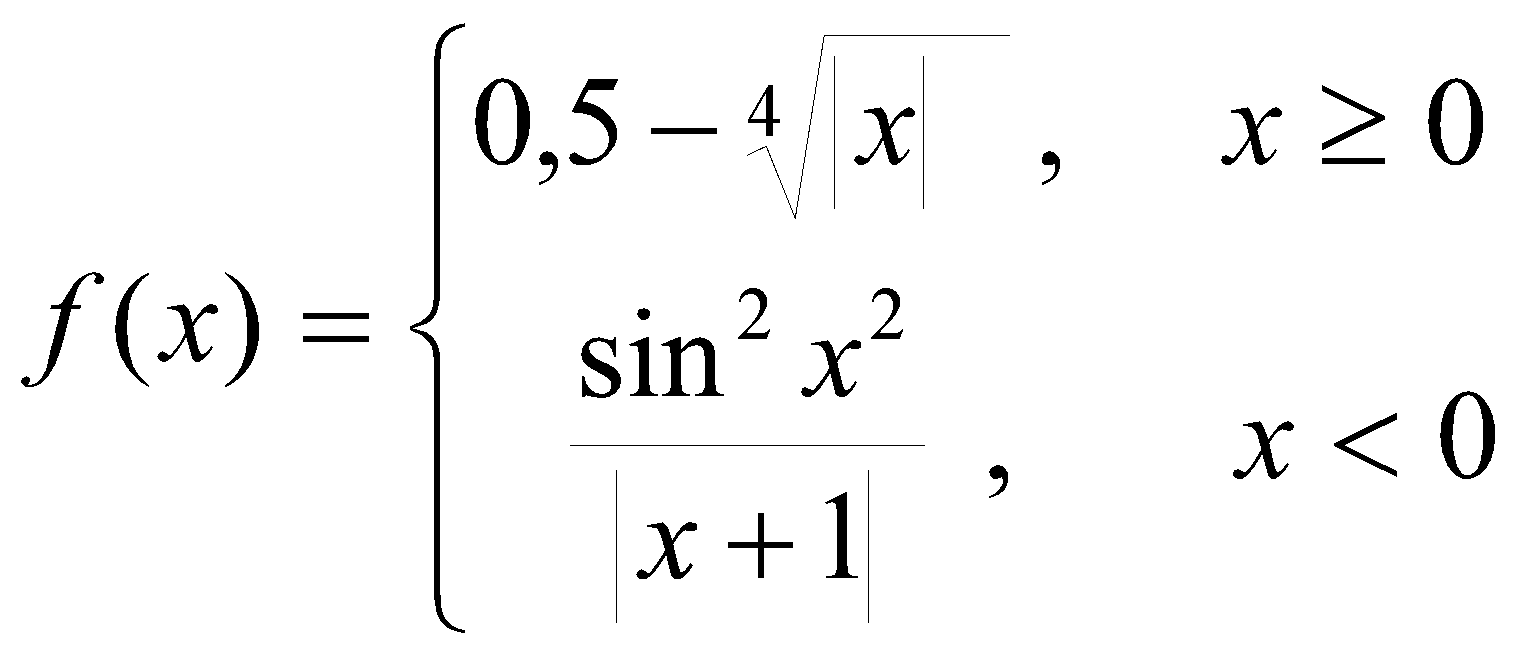
2. Написати програму для розрахунку Z

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/lab2_example.png?attredirects=0)

Для введення даних використовуйте команду input, визначивши тип змінних.

Результат вивести з двома знаками після коми. х - остання цифра у списку групи, t=1.

3.Вичисліть значення функції:



1. Дано три цілих числа. Вибрати з них ті, які належать інтервалу [1,№]. Де №- остання цифра порядкового номеру у списку групи.
2. Дано 3 числа. Знайти мінімальне серед них і вивести на екран. (Для №- парного). Знайти максимальне серед них і вивести на екран. (Для №- непарного).