

Заняття 1. Налаштування Python Windows

Для роботи крім самого інтерпретатора будуть використовуватися:

- бібліотеки Python: *Numpy*, *Matplotlib*, *Pillow*, *SciPy*.
- оболонка IDE - *Jupyter Notebook*,
(*IDLE* , *PyCharm* - <https://www.jetbrains.com/ru-ru/pycharm/download/>).
- бібліотека *Open CV* в версії для *Python*.

Anaconda

1. Встановити *Anaconda* з інстальатора з офіційного сайта

<https://www.anaconda.com/download>



Бібліотеки *Numpy*, *Matplotlib*, *Pillow*, *SciPy* та IDE середовище *Jupyter Notebook* інсталиються автоматично.

Pure Python

1. Встановити *Python* інстальатором з офіційного сайта

<https://python.org/downloads/windows/>

2. Оновити *pip* - систему керування пакетами Python в командному рядку:

```
C:\WINDOWS\system32>pip3 install --upgrade pip
```

Бібліотеки

1. Подивитись всі встановлені бібліотеки та їхні версії можна через *pip* (*pip3* для *Python 3*) через командний рядок:

```
C:\WINDOWS\system32>pip list
Package                               Version
-----
aiohttp                              3.8.1
aiosignal                            1.2.0
alabaster                             0.7.12
anaconda-client                       1.9.0
anaconda-navigator                    2.4.2
anaconda-project                      0.10.2
```

або подивитись версію обраної бібліотеки можна через *pip* в командному рядку:

```
C:\WINDOWS\system32>pip show numpy
Name: numpy
Version: 1.23.4
```

```
C:\WINDOWS\system32>pip show matplotlib
Name: matplotlib
Version: 3.6.2
Summary: Python plotting package
```

```
C:\WINDOWS\system32>pip show jupyter
Name: jupyter
Version: 1.0.0
```

```
C:\WINDOWS\system32>pip show scipy
Name: scipy
Version: 1.7.3
```

```
C:\WINDOWS\system32>pip show opencv-python
Name: opencv-python
Version: 4.6.0.66
```

2. Встановити відсутні бібліотеки *NymPy*, *Matplotlib*, *Pillow*, *SciPy*, *OpenCV* та IDE *Jupyter Notebook* через *pip* в командному рядку:

```
C:\WINDOWS\system32>pip install numpy
```

```
C:\WINDOWS\system32>pip install matplotlib
```

```
C:\WINDOWS\system32>pip install Scipy
```

```
C:\WINDOWS\system32>pip install pillow
```

```
C:\WINDOWS\system32>pip install jupyter
```

```
C:\WINDOWS\system32>pip install opencv-python
```

Примітка.

- Бібліотеки можна встановити в ноутбучі такою ж командою з додаванням символу знака оклику поперед `!pip install ...`
- Встановити бібліотеку OpenCV можна також з офіційного сайта

<http://opencv.org/downloads.html>

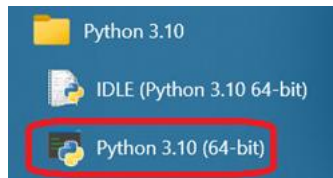
3. В разі потреби оновіть версію до поточної актуальної через *pip*:

```
C:\WINDOWS\system32>pip install --upgrade opencv-python==4.9.0.80
```

Запуск інтерпретатора проводиться

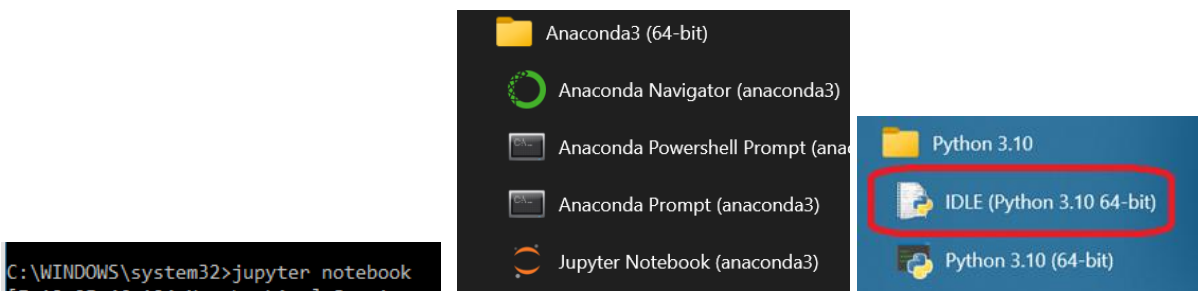
a. з командного рядка або кнопкою панелі «Пуск» (pure Python):

```
C:\WINDOWS\system32>python
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022,
AMD64) on win32
Type "help", "copyright", "credits" or "license()"
>>> import cv2
>>> cv2.__version__
'4.6.0'
```



b. через IDE:

- Python IDLE для pure Python (IDLE запускається кнопкою панелі «Пуск»);
- Jupyter Notebook (запускається з командного рядка для pure Python, кнопкою панелі «Пуск» для Anaconda)



```
IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
Python 3.10.6 (tags/v3.10.6:9c7b4bd,
AMD64) on win32
Type "help", "copyright", "credits"
>>> import cv2
>>> cv2.__version__
'4.6.0'
>>>
```

Bat for desktop icon

```
C:\Windows\System32\cmd.exe /k "jupyter notebook"
```

Jupyter Notebook

Інтернет браузер з вікном середовища Jupyter Notebook відкривається автоматично

В разі неспрацювання автоматичного запуску блокнота в адресному рядку браузера слід ввести одну з адрес з командного рядка Windows

```
To access the notebook, open this file in a browser:
    file:///C:/Users/Igor/AppData/Roaming/jupyter/runtime/nbserver-11384-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=717a2e4c73b01a477bbad0b423b5c328db401f6279db35d5
or http://127.0.0.1:8888/?token=717a2e4c73b01a477bbad0b423b5c328db401f6279db35d5
```

Встановлення робочого каталога для Jupyter Notebook

1. Знайти файл конфігурації **jupyter_notebook_config.py** (зазвичай **C:\Users\your_user_name\.jupyter**) .
2. В разі відсутності файлу створіть його введенням в командному рядку:
jupyter notebook --generate-config
3. Знайти в файлі конфігурації **jupyter_notebook_config.py** рядок:
c.NotebookApp.notebook_dir
4. Задайте для параметра шлях до потрібного каталогу:
c.NotebookApp.notebook_dir = 'C:/your/new/path'

 jupyter

QuitLogout

FilesRunningClusters

Select items to perform actions on them.

UploadNew↺

☐	0		Name	Last Modified	File size
☐		📁	/		
☐		📁	3D Objects	4 года назад	
☐		📁	Contacts	4 года назад	
☐		📁	Desktop	2 місяця назад	
☐		📁	DIP	7 місяців назад	
☐		📁	Documents	4 місяця назад	
☐		📁	Downloads	9 часов назад	

Jupyter Z1Beginningshort Last Checkpoint: 34 minutes ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

Example

[1]: `!pip list`

Package	Version
anyio	4.4.0
argon2-cffi	23.1.0
argon2-cffi-bindings	21.2.0
arrow	1.3.0
asttokens	2.4.1
async-lru	2.0.4
attrs	23.2.0

[2]: `!pip freeze`

```
anyio==4.4.0
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
asttokens==2.4.1
async-lru==2.0.4
attrs==23.2.0
Babel==2.15.0
beautifulsoup4==4.12.3
black==24.4.0
```

Python basics

В залежності від введених даних типи змінних та результатів (**int**, **float**, **complex**) визначаються автоматично. Для виведення типу використовується функція **type()**.

За потреби тип можна явно вказувати після знаку прирівнювання

```
**=exact_type(variable or expression).
```

```
# exact_type - any build in type: int, uint8, float...
```

ПРИКЛАД

Expression 1:

```
Expression1.1
```

```
Expression1.2
```

Expression 2

...

Expression N:

```
Expression N.1:
```

```
Expression N.1.1
```

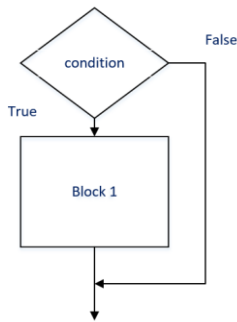
```
Expression N.2
```

...

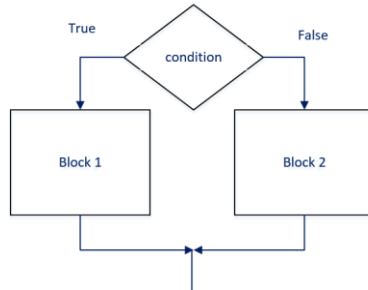
```
Expression N.N
```

Expression N+1

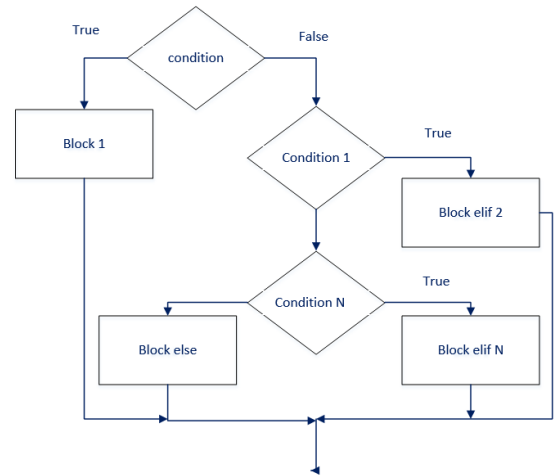
Розгалуження



```
if condition:
    block1
next expression
```



```
if condition:
    block 1
else:
    block 2
next expression
```



```
if condition:
    block1
elif condition 1:
    block elif 2
elif condition N:
    block elif N
else:
    block else
next expression
```

Математичні функції

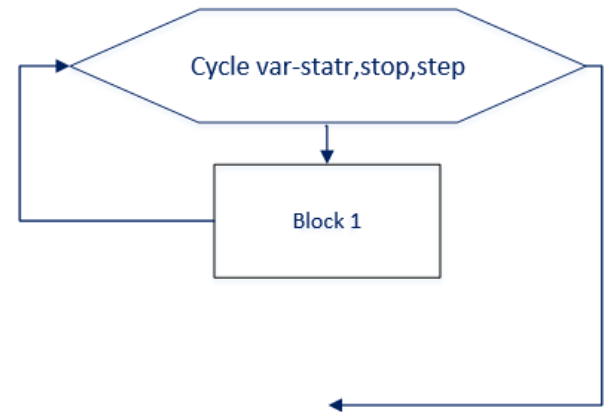
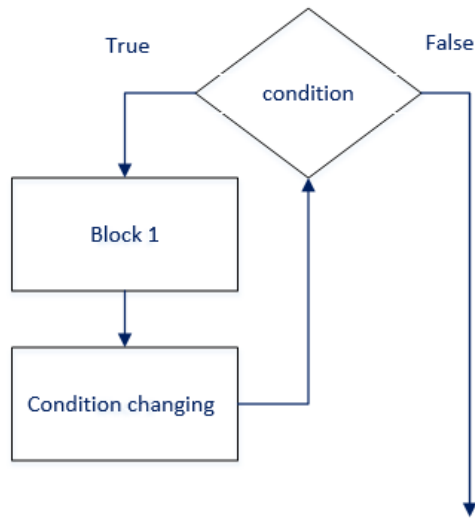
Щоб використовувати математичні функції необхідно імпортувати стандартний модуль **math** (**cmath** для роботи з комплексними числами) за допомогою оператора `import math`.

Список функцій модуля `math` в Python

Функція	Опис
<code>ceil(x)</code>	Найменше ціле число, більше чи рівне x .
<code>fabs(x)</code>	Абсолютне значення x .
<code>factorial(x)</code>	Факторіал x .
<code>floor(x)</code>	Найбільше ціле число, менше чи рівне x .
<code>fmod(x, y)</code>	Залишок при діленні x на y .
<code>frexp(x)</code>	Мантиса та експонента x у вигляді пари (m, e) .
<code>fsum(список)</code>	Сума значень.
<code>modf(x)</code>	Дробова та ціла частини x .
<code>trunc(x)</code>	Відкидає дробову частину числа.
<code>exp(x)</code>	$e^{**}x$.

<code>expm1(x)</code>	$e^x - 1$.
<code>log(x[, b])</code>	Логарифм x за основою b (за замовчуванням e).
<code>log2(x)</code>	Логарифм x за основою 2.
<code>log10(x)</code>	Логарифм x за основою 10.
<code>pow(x, y)</code>	x , піднесений до степеня y .
<code>sqrt(x)</code>	Квадратний корінь з x .
<code>acos(x)</code>	Арккосинус x .
<code>asin(x)</code>	Арксинус x .
<code>atan(x)</code>	Арктангенс x .
<code>atan2(y, x)</code>	$\text{atan}(y/x)$.
<code>cos(x)</code>	Косинус x .
<code>hypot(x, y)</code>	Гіпотенуза трикутника з катетами x та y .
<code>sin(x)</code>	Синус x .
<code>tan(x)</code>	Тангенс x .
<code>degrees(x)</code>	Перетворення кута x з радіанів у градуси.
<code>radians(x)</code>	Перетворення кута x з градусів у радіани.
<code>acosh(x)</code>	Зворотний гіперболічний косинус x .
<code>asinh(x)</code>	Зворотний гіперболічний синус x .
<code>atanh(x)</code>	Зворотний гіперболічний тангенс x .
<code>cosh(x)</code>	Гіперболічний косинус x .
<code>sinh(x)</code>	Гіперболічний синус x .
<code>tanh(x)</code>	Гіперболічний тангенс x .
<code>erf(x)</code>	Функція помилки у точці x .
<code>gamma(x)</code>	Гамма-функція у точці x .
<code>pi</code>	3,14159
<code>e</code>	2,71828

Цикли



```
for cycle_variable in cycle_range:
    expression 1
    expression N
```

```
range(<start=0>,stop,<step=1>) # stop NOT INCLUDE
```

```
while condition:
    block1
    changing condition
next expression
```

ПРИКЛАД

Функції користувача

```
def function_name(<args>):
    function_body expressions
    < return res1 <,res2...>>
```

ПРИКЛАД

Модуль роботи з часом time

Інформація про вказаний годинник.

```
time.get_clock_info(name)
```

Годинники: ім'я-виклик

```
'process_time':    time.process_time(),    'monotonic':      time.monotonic(),  
'thread_time':    time.thread_time(),
```

```
'time':           time.time()           'perf_counter':  time.perf_counter(),
```

Результат:

Adjustable (регульований): True, якщо годинник можна змінити автоматично (наприклад, демоном NTP) або вручну системним адміністратором, False інакше

Implementation (реалізація): назва основної функції C, яка використовується для отримання значення тактової частоти.

Monotonic (монотонний): True, якщо годинник не може повернутися назад, False інакше

resolution: роздільна здатність годинника в секундах (float)

Результатом виклику годинника є дійсне число секунд. (**_ns() – наносекунд як ціле) . Важливість РОЗДІЛЬНОЇ ЗДАТНОСТІ - ТОЧНОСТІ

ПРИКЛАД