

# PROJECTOR 6

## ENTRY IN PYTHON

Basic commands and functions in  
Python

**PRESENTED BY**  
Oleksandr Ukrainets

# Agenda

1

I/O - input/output

2

Data type

3

Bool / Comparision  
operator

4

Conditional construct

5

Logic operator/in  
operator/ identity  
operators

6

Loop - statement

7

Function

8

List and Tuple

# I/O – function

```
input('Write some text for user')
```

Get info from  
user



Work with  
data



Output

```
print('Display information',  
      sep = '\n', end = ';')
```

# Data Type

```
graph TD; DT[Data Type] --> S[Simple]; DT --> C[Complex]; S --> S1[1 Number(Int/Binary/Octal/Hex)]; S --> S2[2 Bool]; S --> S3[3 Number(Float/Decimal/Complex/Fraction)]; C --> C1[1 Ordered collection]; C --> C2[2 Unordered collection]; C --> C3[3 Library collection];
```

## Simple

## Complex

1

Number(Int/Binary/Octal/Hex)

2

Bool

3

Number(Float/Decimal/Complex/Fraction)

1

Ordered collection

2

Unordered collection

3

Library collection

# Number data types

Название (EN)	Название (RU)	Пример
int	целое число	42
float	дробное число	3.14
complex	комплексное число	1 + 2j
bool	логическое	True/False
str	строка	"Hello, World!"
bytes	байты	b"hello"

# Number data types

Название (EN)	Название (UK)	В системе	Пример
float	дробне число	float	3.14
complex	комплексне число	complex	1 + 2j
fractions.Fraction	дріб	Fraction	Fraction(1, 3)
decimal.Decimal	десятькове число	Decimal	Decimal('0.1')

# Bool and comparison operators

Boolean — це тип даних, який може приймати тільки два значення: True (Істина) і False (Неправда).

Він часто використовується для перевірки умов та керування потоком виконання програми.

```
Bool system <class 'type'>  
Value bool  
0: False  
and 0.0:False  
and False is False \m
```

\*Another data is True

# Comparison operators

Оператор	Приклад	Опис
==	a == b	a дорівнює b
!=	a != b	a не дорівнює b
<	a < b	a менше за b
>	a > b	a більше за b
<=	a <= b	a менше або дорівнює b
>=	a >= b	a більше або дорівнює b



# Logical operators

Оператор	Як працює	Приклад
and	Повертає <code>True</code> тільки тоді, коли обидва операнди є істинними	<code>x &gt; 5 and x &lt; 10</code>
or	Повертає <code>True</code> тоді, коли хоча б один з операндів є істинним	<code>x &lt; 5 or x &gt; 10</code>
not	Повертає зворотне значення операнду	<code>not(x &gt; 5 and x &lt; 10)</code>

# Example of code

```
age = 18
```

```
print(age >= 18 and age <= 100)
```



0.0s

True

# In / Not in – operators

Оператор	Як працює	Приклад
in	Повертає <code>True</code> , якщо певний об'єкт міститься в іншому об'єкті	<code>3 in [1, 2, 3, 4, 5]</code>
not in	Повертає <code>True</code> , якщо певний об'єкт не міститься в іншому об'єкті	<code>6 not in [1, 2, 3, 4, 5]</code>

# Example of code

```
country = ( 'USA', 'UK', 'JAP' )  
  
new_country = 'CAD'  
  
✓ print(new_country in country, \  
      |      | new_country not in country )  
✓ 0.0s
```

# Identity operators

Оператор	Як працює	Приклад
is	Повертає <code>True</code> , якщо два операнди є одним і тим же об'єктом	<code>x is y</code>
is not	Повертає <code>True</code> , якщо два операнди не є одним і тим же об'єктом	<code>x is not y</code>

# Example of code

```
a = 5
print(f'A = {a}, id_a = {id(a)}')
b = a
print(f'b is a? {b is a}')
```

✓ 0.0s

A = 5, id\_a = 4518113656

b is a? True

# String

Назва(EN)	Назва(UK)	В системі	Приклад
String	Рядок	str	"Hello, world!"
Concatenation	Конкатенація	+	"Hello" + " " + "world!"
Indexing	Індексація	[]	"Hello"[0] = "H"
Slicing	Зріз	[:]	"Hello"[1:4] = "ell"
Length	Довжина	len()	len("Hello") = 5
Formatting	Форматування	%, format()	"Hello %s" % ("world") = "Hello world", "Hello {}".format("world") = "Hello world"

# String formatting

Назва(EN)	Як працює	Приклад
Concatenation	Рядкова конкатенація	"Hello" + " " + "world!" = "Hello world!"
% Operator	Оператор %	"Hello %s" % ("world") = "Hello world"
str.format() method	Метод str.format()	"Hello {}".format("world") = "Hello world"
f-Strings	f-Рядки	f"Hello {'world'}" = "Hello world"



# String methods

Method	Description
<u>capitalize()</u>	Converts the first character to upper case
<u>casefold()</u>	Converts string into lower case
<u>center()</u>	Returns a centered string
<u>count()</u>	Returns the number of times a specified value occurs in a string
<u>encode()</u>	Returns an encoded version of the string
<u>endswith()</u>	Returns true if the string ends with the specified value
<u>expandtabs()</u>	Sets the tab size of the string
<u>find()</u>	Searches the string for a specified value and returns the position of where it was found
<u>format()</u>	Formats specified values in a string
format_map()	Formats specified values in a string
<u>index()</u>	Searches the string for a specified value and returns the position of where it was found
<u>isalnum()</u>	Returns True if all characters in the string are alphanumeric
<u>isalpha()</u>	Returns True if all characters in the string are in the alphabet
<u>isascii()</u>	Returns True if all characters in the string are ascii characters
<u>isdecimal()</u>	Returns True if all characters in the string are decimals
<u>isdigit()</u>	Returns True if all characters in the string are digits
<u>isidentifier()</u>	Returns True if the string is an identifier
<u>islower()</u>	Returns True if all characters in the string are lower case
<u>isnumeric()</u>	Returns True if all characters in the string are numeric

\*To check all methods `dir('str')`

# If/ Elif / Else – condition

Назва(EN)	Як працює	Приклад
if statement	УМОВНИЙ оператор if	<code>if x &gt; y: print("x is greater than y")</code>
else statement	Оператор else	<code>if x &gt; y: print("x is greater than y") else: print("y is greater than x")</code>
elif statement	Оператор elif	<code>if x &gt; y: print("x is greater than y") elif x &lt; y: print("y is greater than x") else: print("x and y are equal")</code>

# If - else statement

```
age = 18
```

```
if age >= 18:
```

```
    # If conditional is true, than block of code
```

```
    print("You are an adult.")
```

```
else:
```

```
    # In any another input
```

```
    print("You are not yet an adult.")
```

✓ 0.0s

You are an adult.

# If – elif – else

```
x = 10

if x > 10:
    print("x більше 10")
elif x == 10:
    print("x дорівнює 10")
else:
    print("x менше 10")
```

✓ 0.0s


x дорівнює 10

# Match / Case (*Python 3.9+*)

```
def math_operation(a, b, operator):  
    match operator:  
        case '+':  
            return a + b  
        case '-':  
            return a - b  
        case '*':  
            return a * b  
        case '/':  
            if b == 0:  
                return "Ділення на нуль неможливе"  
            else:  
                return a / b  
        case _:  
            return "Невідомий оператор"
```

# Test tasks

1. Напишіть програму, яка зчитує вік користувача та виводить повідомлення "Ви повнолітній", якщо користувач має 18 або більше років, та "Ви неповнолітній", якщо менше 18.
2. Створіть програму, яка визначає, чи є рік, введений користувачем, високосним. Якщо рік високосний, програма має вивести "Це високосний рік", інакше - "Це не високосний рік".
3. Напишіть програму, яка зчитує три числа від користувача та виводить найбільше з них.
4. Створіть програму, яка запитує у користувача кольори світлофора, який знаходиться на перехресті, та виводить повідомлення "Рух заборонено", "Рух дозволено" або "Рух дозволено з обережністю", в залежності від того, який колір світлофора вказує на поточний момент.
5. Напишіть програму, яка перевіряє, чи належить введене користувачем число діапазону від 1 до 100 включно, та виводить повідомлення "Число належить діапазону", якщо так, і "Число не належить діапазону", якщо ні.

- 
1. Створіть програму, яка зчитує вік користувача та використовуючи `match/case` виводить повідомлення залежно від діапазону віку: "Ви підліток" для віку від 13 до 17, "Ви доросла людина" для віку від 18 до 59, "Ви сеніор" для віку від 60 та старше, або "Введіть правильний вік", якщо вік не потрапляє в жоден з цих діапазонів.
  2. Напишіть програму, яка зчитує номер місяця від користувача та використовуючи `match/case`

# Loop control statement

**Блок не може бути порожнім — це помилка для системи!**

## **Pass:**

На випадок, якщо ми не хочемо поки що нічого писати,  
або ще не знаємо що

```
def add_value():  
    |  
    pass
```

✓ 0.0s



# Loops for/while

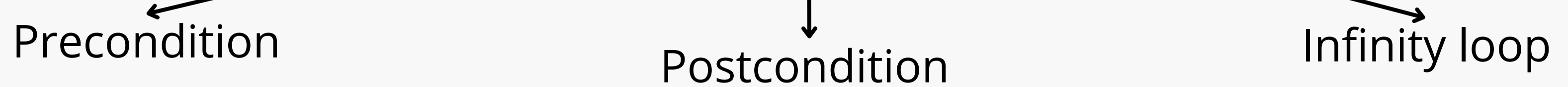
Цикли використовуються в тих випадках, коли нам потрібно зробити що-небудь багато разів. Нерідко вам доведеться виконати якусь операцію даних знову і знову. При цьому ми можемо змінювати умову в процесі роботи

Ітерація в програмуванні - організація опрацювання даних, за якої дії повторюються багаторазово

# Example of code

Назва(EN)	Назва(UK)	Як працює	Приклад
for loop	цикл for	Виконується певну кількість разів заздалегідь відомий	<code>`for i in range(5): print(i)`</code> - виведе числа від 0 до 4
while loop	цикл while	Виконується доти, поки умова істинна	<code>`i = 0 while i &lt; 5: print(i) i += 1`</code> - виведе числа від 0 до 4
do-while loop	цикл do-while	Виконується хоча б один раз, потім перевіряється умова і, якщо вона істинна, цикл виконується знову	<code>`i = 0 do: print(i) i += 1 while i &lt; 5`</code> - виведе числа від 0 до 4

# Types of loops



```
x = 0
if x < 10:
    while x < 10:
        print(x)
        x += 1
```

```
x = 5
while True:
    # complete loop
    x -= 1
    if x < 0:
        break
```

```
while True:
    print('Inf true')
```

# Example of for loop

```
country = ('USA', 'UK', 'JAP')
```

```
for i in country: # where i it is iterator,  
    print(i)
```



0.0s

USA

UK

JAP

# Example of code

```
counter = 0
while True:
    if counter != len(country):
        print(country[counter])
        counter += 1
    elif counter == len(country):
        break
```

# Example of code

```
# or  
counter = 0  
while counter != len(country):  
    print(country[counter])  
    counter += 1
```

# Control flow

Назва (EN)	Назва (ЕК)	Як працює
continue	продовжити	Припиняє поточну ітерацію циклу і переходить до наступної ітерації без виконання коду, який знаходиться після команди continue.
break	перервати	Завершує цикл і виходить з нього, не дочекується його природної завершеності.
pass	нічого не робити	Ця команда нічого не робить. Використовується як заповнювач для майбутнього коду.
try/except	спробувати/ виняток	Використовується для обробки виключень. Код, який може викликати виключення, поміщається в блок try, а код обробки виключень - в блок except. Якщо виключення не виникає, блок except не виконується. Якщо виключення виникає, виконується відповідний блок except.

# Break iteration

```
# Example of break
for i in range(0, 15):
    if i == 11:
        print(f'End point {i}, break loop')
        break
```



# Example of continue

```
# Example of continue
for i in range(0, 15):
    if i == 11:
        print(f'Pass through point {i}, continue loop')
        continue
```

# Tasks

1. Знайти суму перших  $N$  натуральних чисел.
2. Вивести всі парні числа від 1 до  $N$ .
3. Перевірити, чи є дане число простим.
4. Знайти факторіал числа  $N$ .
5. Обчислити середнє арифметичне заданого списку чисел.
6. Зчитати рядок з клавіатури і вивести його в зворотньому порядку.
7. Знайти найбільший спільний дільник двох чисел.
8. Підрахувати кількість входжень заданого символу у рядок.
9. Знайти суму цифр числа  $N$ .
10. Перевірити, чи є рядок паліндромом.

## RLE encoder

Given a string of letters (without numbers), create a string encoding it by the rules where the first character is char itself, followed by a number indicating the number of letter repeats.

Examples:

ABBA => A1B2A1 ATTTGC => A1T3G1C1

*Thank  
you!*