

# **СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ**

**122 «Комп'ютерні науки»  
КН-18**

**2020 / 2021 навчальний рік**

# PYTHON #5

1. Функції
2. Поліморфізм
3. Область видимості
4. Зіставлення аргументів

[https://github.com/eabshkvprof/2021\\_Mod\\_Prog\\_Tech](https://github.com/eabshkvprof/2021_Mod_Prog_Tech)

# ФУНКЦІЇ

*Функція* → базова програмна структура мови , що забезпечує багаторазове використання програмного коду і зменшує його надмірність.

*Функція* → засіб, що дозволяє групувати набори інструкцій так, що в програмі вони можуть запускатися неодноразово.

# ФУНКЦІЇ

Інструкція	Приклад
<b>def , return</b>	<b>Def</b> myfunc(a,b): return a+b
<b>Виклик</b>	myfunc(1,2)
<b>yield</b>	<b>Def</b> sqr(x): for l in range (x): <b>yield</b> i**2
<b>global</b>	<b>Def</b> spb( ): <b>global</b> x; x = 'new'
<b>nonlocal</b>	<b>Def</b> spb( ): <b>nonlocal</b> x; x = 'new'
<b>lambda</b>	Func = [ <b>lambda</b> x: x*2, <b>lambda</b> x: x**2]

# СТВОРЕННЯ ФУНКЦІЇ

**def <name>(arg1, arg2,...,argN) :**

Заголовок

*<statements>*

Відступ

**def <name>(arg1, arg2,...,argN) :**

*<statements>*

*return <value>*

Вертає результат  
(необов'язково)

*<statements>*

**def** - ІНСТРУКЦІЯ, яка створює новий об'єкт типу **функція** і присвоює ім'я цьому об'єкту.

# ПОЛІМОРІФІЗМ

**!!! ПОЛІМОРФІЗМ** – сенс операції залежить від типів оброблюваних об'єктів. Python - це мова з динамічною типізацією, поліморфізм в ньому проявляється всюди.

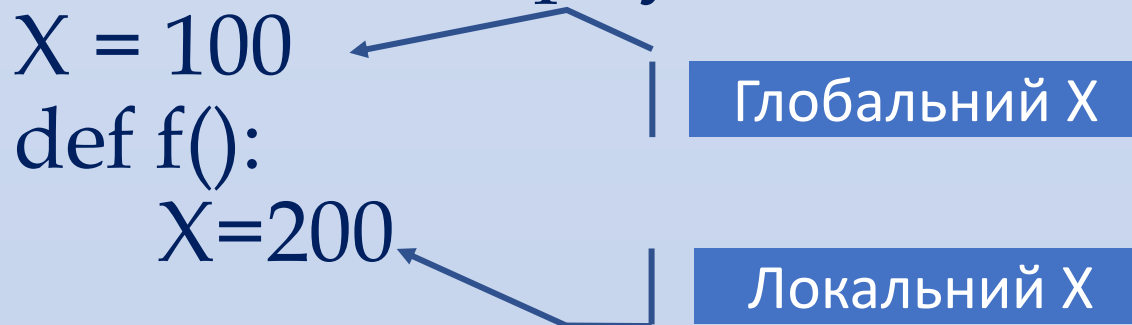
Всі операції в мові Python є поліморфічні, поки об'єкти підтримують очікуваний інтерфейс (протокол), функція зможе обробляти їх.

**Поліморфізм** - це можливість обробки різних типів даних за допомогою "одне і тієї ж" функції, або методу.

# ОБЛАСТЬ ВИДИМОСТІ

За замовчуванням всі імена, значення яких присвоюються всередині функції, асоціюються з простором імен цієї функції і ніяк інакше. Це **ЛОКАЛЬНІ** змінні.

- Імена, які визначаються всередині інструкції **def**, видно тільки програмного коду всередині інструкції **def**. До цих імен **неможна** звернутися за межами функції.



# ОБЛАСТЬ ВИДИМОСТІ

- **LOCAL** область - присвоювання змінної виконується всередині інструкції *def*, змінна є локальною для цієї функції.
  - **NONLOCAL** область - присвоювання проводиться в межах охоплюючий інструкції *def*, змінна є нелокальною для цієї функції.
  - **GLOBAL** область присвоювання проводиться за межами всіх інструкцій *def*, вона є глобальною для всього файлу.
- Python - **лексична область видимості** - видимість змінної визначаються місцем розташування цієї змінної у вихідних текстах програми, а не місцем, звідки викликаються функції.



# ІНСТРУКЦІЇ **global**, **nonlocal**

!!! Інструкції **global** / **nonlocal** не оголошують тип або розмір змінної - вони оголошують простір імен.

Інструкція **global** дозволяє змінювати змінні, що знаходяться на верхньому рівні модуля, за межами інструкції **def**.

- Глобальні імена - це імена, які визначені на верхньому рівні вміщаючого модуля.
- Глобальні імена повинні оголошуватися, тільки якщо їм будуть присвоюватися значення всередині функцій.
- Звертатися до глобальних імен всередині функцій можна і без об'явлення їх глобальними.

Інструкція **nonlocal** застосовується до імен, які перебувають в локальних областях видимості, що охоплюються інструкцій **def**.<sup>9</sup>

# ПРАВИЛО LEGB

## PYTHON Вбудована область видимості

Зумевлені імена (open, range ....)

**B**

## МОДУЛЬ Глобальна область видимості

Імена, визначені на верхнім рівні модуля або оголошені в інструкції **def** як глобальні

**G**

## ФУНКЦІЯ non локальна область видимості

Імена, визначені в локальній області видимості та в функції, яка охоплює

**E**

## ФУНКЦІЯ локальна області видимості

Імена, визначені в тілі функції

**L**

# ОБЛАСТЬ ВИДИМОСТИ

```
X = 100
def func(Y):
    Z=X+Y
    return Z
print (func(1))
print (X)
```

---

101

100

```
X = 100
def func(Y):
    X=200
    Z=X+Y
    return Z
print (func(1))
print (X)
```

---

201

100

```
X = 100
def func(Y):
    global X
    X=200
    Z=X+Y
    return Z
print (func(1))
print (X)
```

---

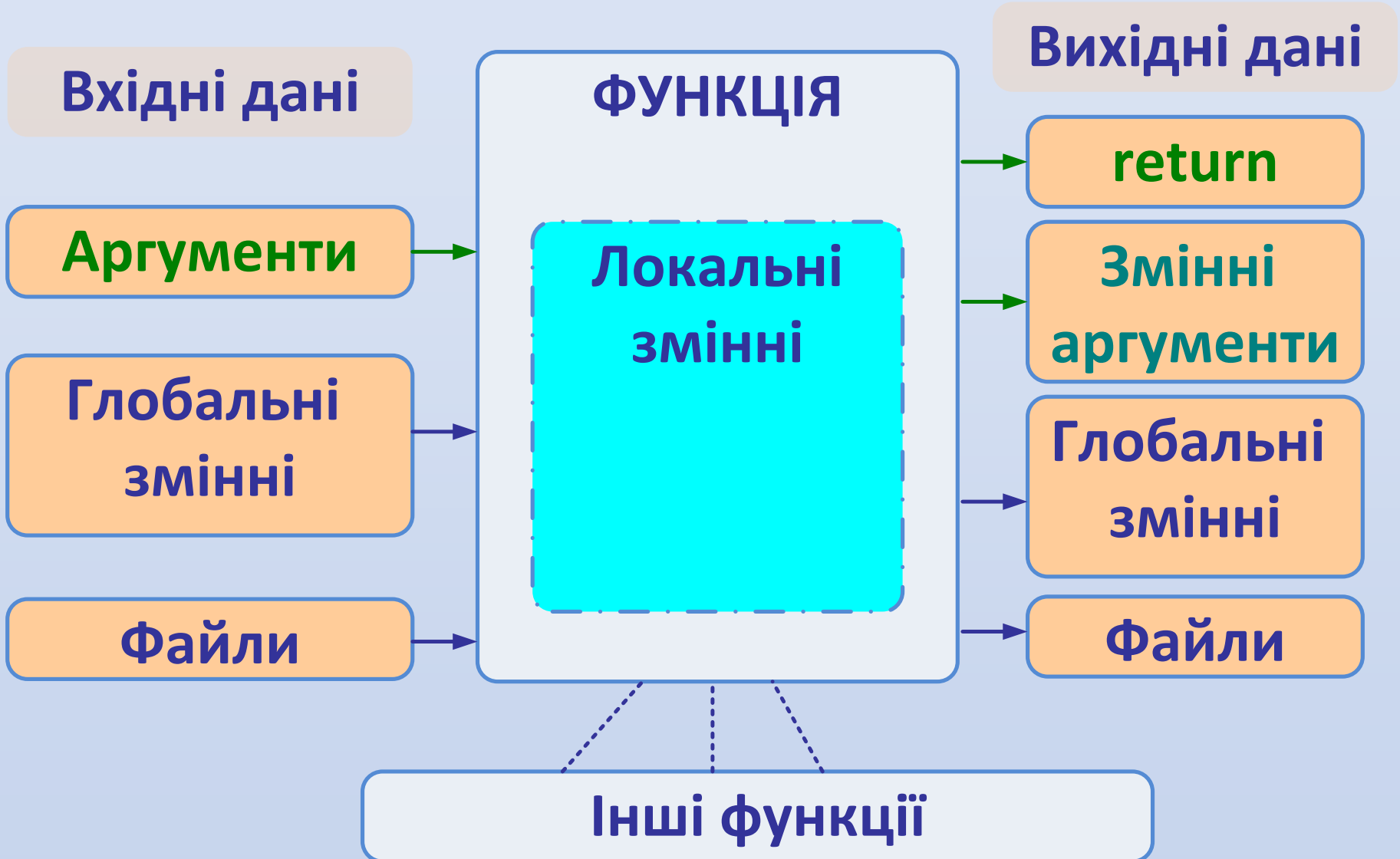
201

200

# АРГУМЕНТИ

- **НЕЗМІННІ** АРГУМЕНТИ - передаються «за значенням» – передаються в вигляді **посилань** на об'єкти – (не в вигляді копій!).  
Безпосередня зміни аргументу всередині функції НЕМОЖЛИВА.
- **ЗМІННІ** АРГУМЕНТИ - передаються «за вказівником» – передаються в вигляді вказівника . Допускається можливість безпосередньої зміни аргументу всередині функції.

# АРГУМЕНТИ



# АРГУМЕНТ за ЗАМОВЧУВАННЯМ

Дозволяють зробити окремі аргументи функції необов'язковими.

Якщо значення не передається, аргумент отримує значення за замовчуванням, яке визначено у заголовку функції

```
def <name>(arg1, arg2=<def_value>,...) :  
    <statements>
```

# АРГУМЕНТИ. РЕКОМЕНДАЦІЇ

- Для передачі значень до функції використовуйте аргументи, для повернення результатів - інструкцію **return**.
- **Не використовуйте глобальні змінні** (! тільки якщо це дійсно необхідно).
- **Не впливайте на змінювані аргументи**, якщо модуль, що викликає, не передбачає цього.

# ЗІСТАВЛЕННЯ АРГУМЕНТІВ

#Визначення функції

```
def foo(arg1, arg2,...,argN) :  
    <statements>
```

#Виклик функції

```
foo(par_x, par_y,..., par_z)
```



Який параметр виклику відповідає аргументу визначення ?



# ЗІСТАВЛЕННЯ АРГУМЕНТІВ

- **ПОЗИЦІЙНИ** аргументи - відповідність визначається за позицією аргументу у визначенні функції та її виклику.
- **ІМЕНОВАНІ** аргументи – дозволяють визначити відповідність за іменами, а не за позиціями аргументів.

# ЗІСТАВЛЕННЯ АРГУМЕНТІВ

Опис	Виклик	
<code>def func(name)</code>	<code>func(value)</code>	За позицією або за ім'ям
<code>def func (name=value)</code>	<code>func (name=value)</code>	Значення аргументу за замовчуванням, якщо аргумент не передається функції
<code>def func(*name)</code>	<code>Func (*sequence)</code>	Довільне число аргументів за позицією (кортеж)
<code>def func(**name)</code>	<code>func(** dict)</code>	Довільне число аргументів за іменами (словник)
<code>def func(*args, name)</code>		≥3.0 Аргументи, що передаються тільки за іменами
<code>def func(*, name=value)</code>		

# ЗІСТАВЛЕННЯ АРГУМЕНТІВ

У заголовку функції аргументи повинні вказуватися в наступному порядку:  
будь-які звичайні аргументи (**name**),  
- за якими можуть слідувати аргументи зі значеннями за замовчуванням (**name = value**),  
-- за якими слідують аргументи в формі **\*args** (або **\*** в 3.0), якщо є ,  
--- за якими можуть слідувати будь-які імена або пари **name = value** аргументів, які передаються тільки по імені (в 3.0),  
---- за якими можуть слідувати аргументи в формі **\*\*kwargs** (останні!).

# ЗІСТАВЛЕННЯ АРГУМЕНТІВ

У виклику функції аргументи повинні вказуватися в наступному порядку: будь-які позиційні аргументи (значення), за якими можуть слідувати будь-які іменовані аргументи (**name = value**) і аргументи у формі **\*sequence**, за якими можуть слідувати аргументи в формі **\*\*dict** (остання, відповідає **\*\*kwargs** заголовку).

# ЗІСТАВЛЕННЯ АРГУМЕНТІВ

Дії інтерпретатору:

1. Сопоставлення неіменованих аргументів за позиціями.
2. Сопоставлення іменованих аргументів за іменами
3. Сопоставлення додаткових неіменованих аргументів з кортежем *\*args*.
4. Сопоставлення додаткових іменованих аргументів з словником *\*\*kwargs*.
5. Сопоставлення значень за замовчуванням з відсутніми іменованими

## Рекомендована ЛІТЕРАТУРА

- **Програмування числових методів мовою Python:** підруч. / А. В. Анісімов, А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2014. – 640 с.
- **Програмування числових методів мовою Python:** навч. посіб. / А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий, Є. В. Глушко ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2013. – 463 с.
- **Основи програмування Python:** Підручник для студ. спеціальності 122 «Компютерні науки» / А.В.Яковенко; КПІ.- Київ: КПІ, 2018 . – 195 с.
- **Лутц М.** Изучаем Python, 4-е издание. - СПб.: Символ-Плюс. 2011.- 1280 с.: ил.

# Контрольні запитання

- Наведіть визначення функції в мові Python. Надайте опис інструкції створення функції. вкажіть призначення її компонентів.
- Наведіть перелік областей видимості для змінних функції. Надайте інструкції керування областями видимості для змінних функції. Наведіть приклади використання.
- Наведіть визначення аргументів за замовчуванням, наведіть приклади використання.
- Наведіть правила зіставлення аргументів функції у її описі та її виклику.

**The END**  
**Mod 1. Lec 5.**