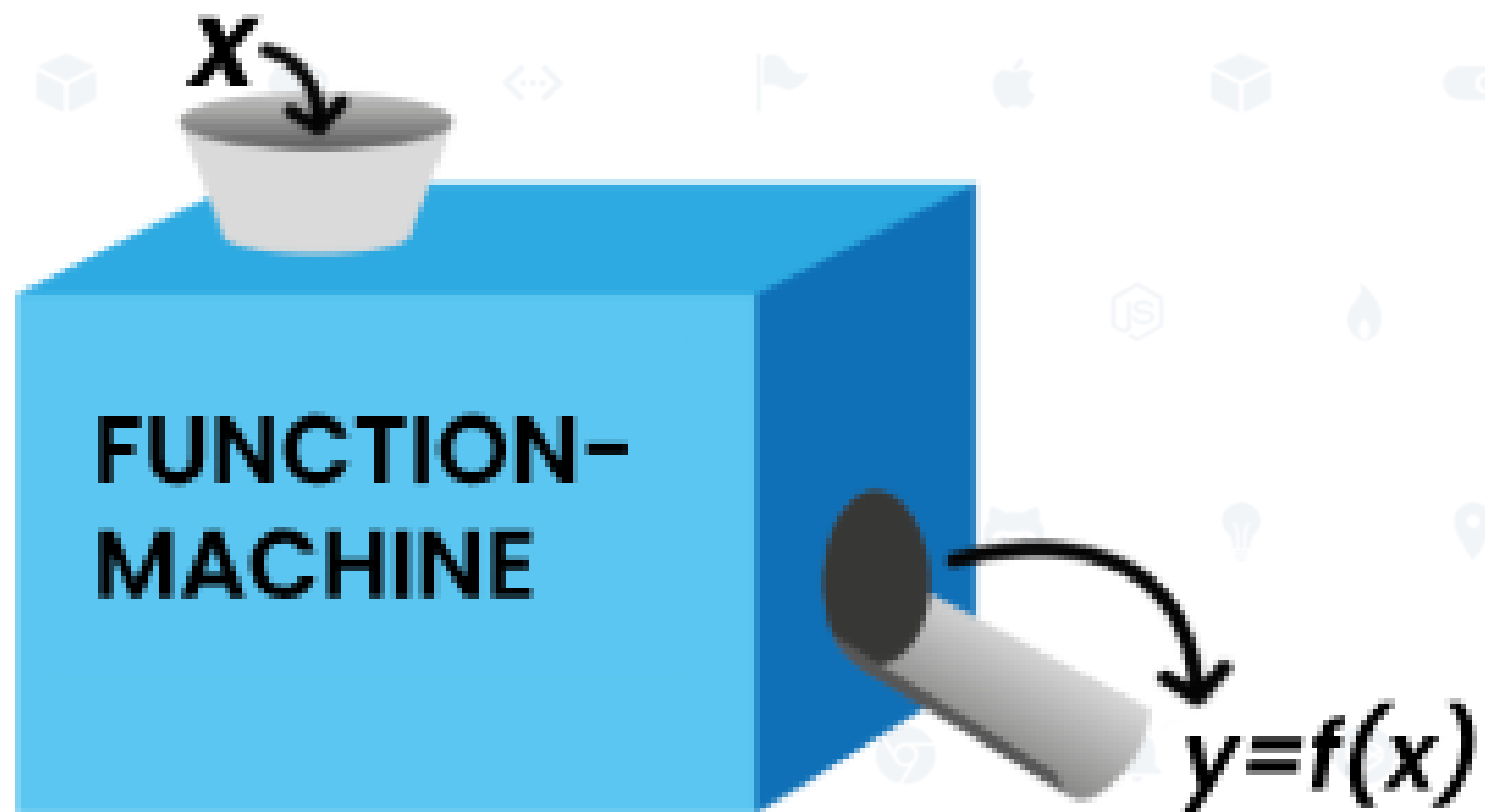


Функції



Що має мати функція обов'язково:

Ім'я функції

Ключове слово def

Круглі дужки ()

Тіло функції (pass)

Що може бути в функції опційно:

Аргументи (прописуються при створенні, позиційні й іменовані)

Значення за замовченням

Параметри (передаються при виклику)

Повернення результату (return). Може бути декілька чи жодного. Але при його виконанні, функція припиняє працювати!!!

Docstring

Декоратори

Анотації типів

Лямбда-функції

Функція в Python, яка не має імені та створюється за допомогою ключового слова `lambda`. Вона використовується для коротких обчислень або завдань і часто передається як аргумент до інших функцій.

`lambda` аргументи: вираз

`lambda` — ключове слово, що визначає лямбда-функцію. Замість **`def func()`**

Аргументи — список вхідних значень, розділених комами (може бути порожнім). Замість **`*args`**

Вираз — один рядок, що обчислюється і повертається як результат. Замість **тіла** та **`return`**

Найчастіше використовується у **`map()`**, **`filter()`**, **`reduce()`** (не є вбудованою, імпортується з `functools`).

,

Функції з параметрами, з поверненням значення

```
def square(number):  
    return number ** 2  
print(square(4))
```

Функції без параметрів, без повернення значень

```
def make_call():  
    make_call_to_db
```

За відсутності return функція поверне None

The background of the slide is filled with a repeating pattern of small, light blue icons. These icons include various symbols such as gears, people, speech bubbles, briefcases, LinkedIn logos, flags, apples, cubes, eyes, code symbols, games, hexagons, flames, shields, information icons, folders, lightbulbs, location pins, cat faces, clouds, circles with numbers, triangles, hands, and computer monitors.

Необов'язкові параметри (значення за замовчуванням)

```
def greet(name="Guest"):  
    return f"Hello, {name}"
```

```
print(greet())  
print(greet("Kyrylo"))
```



Позиційні аргументи

```
def greet(first_name, last_name):  
    return(f"Hello, {first_name} {last_name}!")
```

```
print(greet("John", "Doe"))
```

Іменовані аргументи

```
def greet(first_name, last_name):  
    return(f"Hello, {first_name} {last_name}!")
```

```
greet(first_name="Jane", last_name="Smith")
```



```
def devide(a,b):  
    return a/b
```

```
print(devide(10, 5)) #2.0
```

```
print(devide(5, 10)) #0.5
```

```
print(devide(a = 10, b = 5)) #2.0
```

```
print(devide(b = 10, a = 5)) #2.0
```




```
def devide(a,b):  
    return a/b
```

```
print(devide(a=10, 5))
```

SyntaxError: positional argument follows keyword argument

```
print(devide(10, a=5))
```

TypeError: devide() got multiple values for argument 'a'

Змішування позиційних і іменованих аргументів

```
def greet(first_name, last_name, age):  
    return(f"{first_name} {last_name} is {age} years old.")
```

```
print(greet("John", "Doe", age=30))
```

***args, **kwargs**

```
def sum_numbers(*args):  
    return sum(args)
```

```
def print_info(**kwargs):  
    for key, value in kwargs.items():  
        print(f"{key}: {value}")
```

```
def full_info(*args, **kwargs):  
    print("Positional arguments:", args)  
    print("Keyword arguments:", kwargs)
```

*args - кортеж, **kwargs - словник.

*args, **kwargs - поширені слова, але не обов'язкові

Примусово позиційні аргументи (/)

```
def greet(first_name, last_name, /, greeting="Hello"):
    print(f"{greeting}, {first_name} {last_name}!")
```

Аргументи до "/" можуть бути лише позиційними!

```
greet("John", "Doe", greeting="Hi")
```

Примусово іменовані аргументи (*)

```
def greet(*, first_name, last_name):  
    return(f"Hello, {first_name} {last_name}!")
```

Усі параметри після * нього повинні передаватися лише іменовано.

```
greet(first_name="John", last_name="Doe")
```

Комбінування / і *

```
def greet(first_name, /, last_name, *, greeting="Hello"):
    return(f"{greeting}, {first_name} {last_name}!")
```

```
print(greet("Kyrylo", "Polinchuk", greeting = "Hi"))
```

Анотації типів

```
def add_numbers(a: int, b: int) -> int:  
    return a + b
```

```
def greet(name: str) -> None:  
    print(f"Hello, {name}!")
```



```
from typing import List, Tuple, Dict, Union
```

```
def process_items(items: List[int]) -> Tuple[int, int]:  
    return min(items), max(items)
```

```
def parse_value(value: Union[int, str]) -> str:  
    return str(value)
```

```
def find_user(user_id: int) -> Optional[str]:  
    return "User Name" if user_id == 1 else None
```




Python 3.9+

```
def get_user_info(users: list[str]) -> dict[str, int]:  
    return {user: len(user) for user in users}
```

Docstring

```
def add(a: int, b: int) -> int:
```

```
    """ Add two integers and return the result.
```

```
    Args:
```

```
    a (int): The first integer.
```

```
    b (int): The second integer.
```

```
    Returns: int: The sum of the two integers. """
```

```
    return a + b
```

```
help(add)
```

```
print(add.__doc__)
```


Як працює замикання (closure)

Замикання виникає, коли функція визначається всередині іншої функції.

Внутрішня функція має доступ до змінних із оточення (scope) зовнішньої функції.

Ці змінні "запам'ятовуються" навіть після завершення роботи зовнішньої функції.

```
def outer_function(x):  
    def inner_function(y):  
        return x + y  
    return inner_function
```



```
def make_multiplier(factor):  
    def multiplier(x):  
        return x * factor  
    return multiplier
```

```
double = make_multiplier(2)  
triple = make_multiplier(3)
```

```
double(5) #10
```

```
triple(10) # 30
```

