

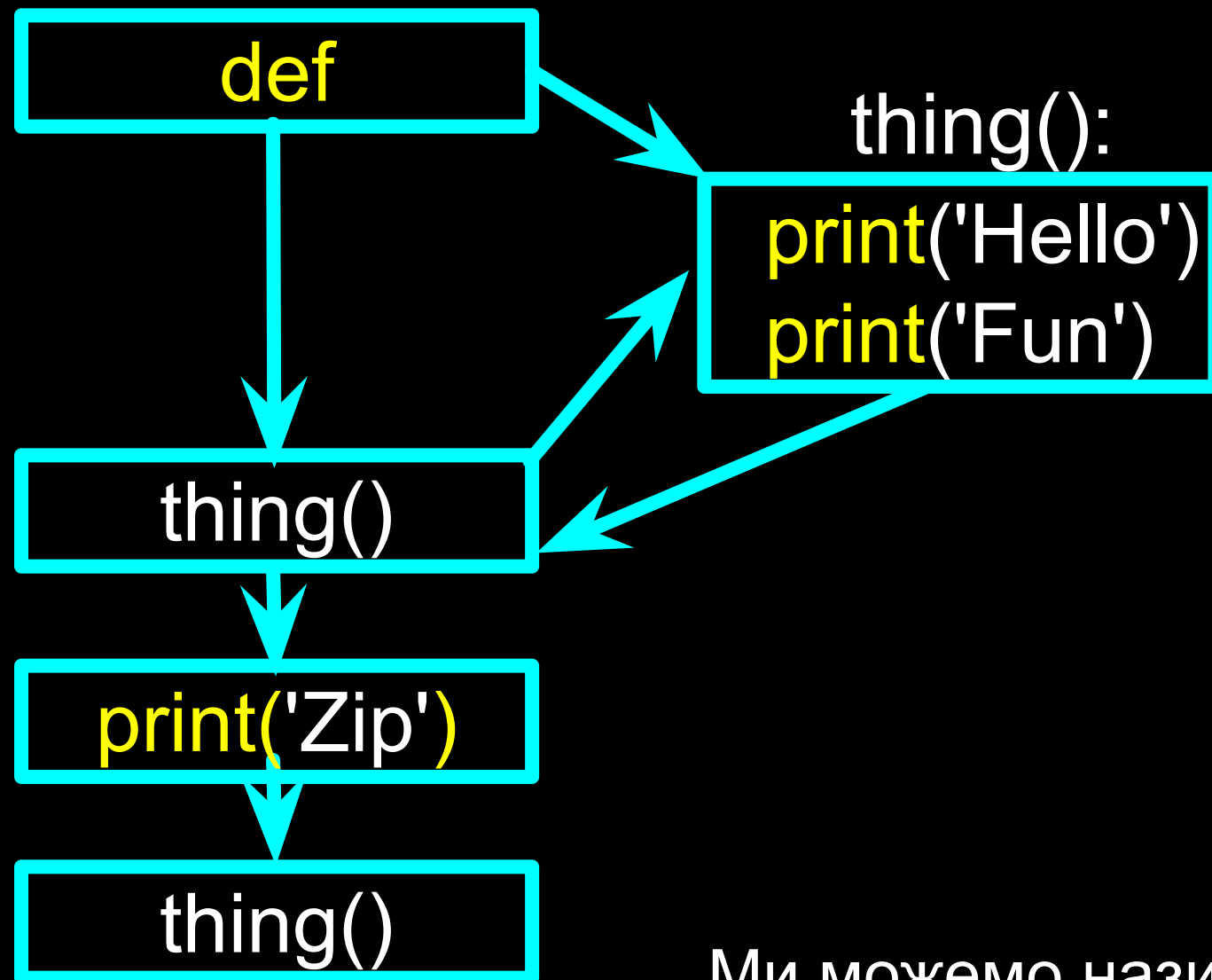
# Функції

## Розділ 4

Python для всіх  
[www.py4e.com](http://www.py4e.com)



# Кроки для зберігання та повторного використання



Програма:

```
def thing():  
    print('Hello')  
    print('Fun')
```

```
thing()  
print('Zip')  
thing()
```

Вивід:

```
Hello  
Fun  
Zip  
Hello  
Fun
```

Ми можемо називати «функціями» рядки коду багаторазового використання

# Функції Python

- Існує два типи функцій в Python.
  - Вбудовані функції (надаються як частина Python): `print()`, `input()`, `type()`, `float ()`, `int ()` ...
  - Функції, які ми самі створюємо та використовуємо
    - Ми сприймаємо імена функції як «нові» зарезервовані слова (тобто уникаємо їх при іменуванні змінних)

# Визначення функції

- У Python **функція** – це код багаторазового використання, який отримує на вхід **аргумент(и)**, виконує певні обчислення, а потім повертає результат чи результати
- Для визначення **функції** застосовуємо ключове слово **def**.
- Викликаємо / звертаємось до **функції** за допомогою її імені, дужок та **аргументів** у цих дужках

big = max('Hello world')

Аргумент

Присвоивания

'w'

Результат

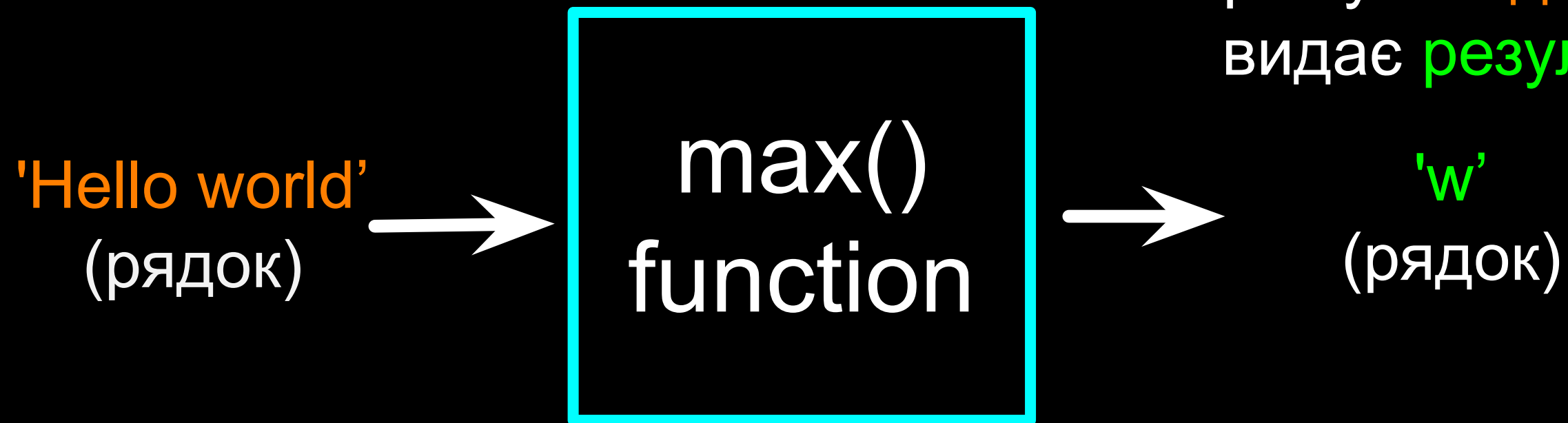
```
>>> big = max('Hello world')
>>> print(big)
w
>>> tiny = min('Hello world')
>>> print(tiny)

>>>
```

# Функція max

```
>>> big = max('Hello world')  
>>> print(big)  
w
```

Функція - це деякий збережений код, який ми використовуємо. Вона отримує **вхідні дані** й видає **результат**.



Цей код написав Гвідо

# Функція max

```
>>> big = max('Hello world')
>>> print(big)
w
```

Функція - це деякий збережений код, який ми використовуємо. Вона отримує **вхідні дані** й видає **результат**.

'Hello world'  
(рядок)



```
def max(inp):
    blah
    blah
    for x in inp:
        blah
        blah
```



'w'  
(рядок)

Цей код написав Гвідо

# Конвертування типів

- Коли вираз містить ціле число (integer) і число з рухомою крапкою (float), ціле число **неявно** конвертується у число з рухомою крапкою
- Ви можете контролювати це за допомогою вбудованих функцій `int()` і `float()`

```
>>> print(float(99) / 100)
0.99
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>> print(1 + 2 * float(3) / 4 - 5)
-2.5
>>>
```



# Конвертація рядків

- Ви також можете використовувати `int()` і `float()` для конвертування рядка в число
- Ви отримаєте **помилку**, якщо рядок не містить цифрових символів

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str
(not "int") to str
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

Власні функції...

# Створення власних функцій

- Ми створюємо нову функцію за допомогою ключового слова **def**, після якого йде ім'я функції з дужками. У дужках, за необхідності, додаються параметри
- Тіло функції пишеться із відступом
- Так **def** визначає функцію, **запам'ятовує** її, але не запускає

```
def print_lyrics():  
    print("I'm a lumberjack, and I'm okay.")  
    print('I sleep all night and I work all day.')
```

print\_lyrics():

```
print("I'm a lumberjack, and I'm okay.")  
print('I sleep all night and I work all day.')
```

```
x = 5
```

```
print('Hello')
```

```
def print_lyrics():
```

```
    print("I'm a lumberjack, and I'm okay.")
```

```
    print('I sleep all night and I work all day.')
```

```
print('Yo')
```

```
x = x + 2
```

```
print(x)
```

Hello!

Yo

7

# Визначення та використання

- Якщо ви **визначили** функцію, потім **звертатиметеся** (або **викликатимете**) її скільки заманеться
- Так ви **зберігаєте** та **повторно використовуєте** шаблон

```
x = 5
print('Hello')

def print_lyrics():
    print("I'm a lumberjack, and I'm okay.")
    print('I sleep all night and I work all day.')

print('Yo')
print_lyrics()
x = x + 2
print(x)
```

Hello

Yo

I'm a lumberjack, and I'm okay.

I sleep all night and I work all day.

7

# Аргументи

- **Аргумент** – це передане **функції** значення, яке використовується як **вхідні дані** при її виклику
- Ми використовуємо **аргументи**, щоб «наказати» **функції** виконувати різні **завдання**, коли ми час від часу викликатимемо її
- **Аргументи** беремо у дужки після імені **функції**


```
big = max('Hello world')
```



Аргумент

# Параметри

Параметр – це змінна, яку ми використовуємо у визначенні функції. Наче «тримач» або «комірка», яка дозволяє коду функції отримати доступ до аргументів, для конкретного виклику функції.



```
>>> def greet(lang):  
...     if lang == 'es':  
...         print('Hola')  
...     elif lang == 'fr':  
...         print('Bonjour')  
...     else:  
...         print('Hello')  
...
```

```
>>> greet('en')
```

```
Hello
```

```
>>> greet('es')
```

```
Hola
```

```
>>> greet('fr')
```

```
Bonjour
```

```
>>>
```



Аргумент



# Повернені значення

Часто функція отримує **аргументи**, виконує певні обчислення і **повертає значення**, яке використовується як значення виклику функції у виразі. Аби зробити це, використайте ключове слово **return**.

```
def greet():  
    return "Hello"
```

```
print(greet(), "Glenn")  
print(greet(), "Sally")
```

```
Hello Glenn  
Hello Sally
```

# Повернене значення

- Продуктивна функція дає результат (або повертає значення)
- Інструкція `return` завершує виконання функції та повертає її результат

```
>>> def greet(lang):  
...     if lang == 'es':  
...         return 'Hola'  
...     elif lang == 'fr':  
...         return 'Bonjour'  
...     else:  
...         return 'Hello'  
...  
>>> print(greet('en'), 'Glenn')  
Hello Glenn  
>>> print(greet('es'), 'Sally')  
Hola Sally  
>>> print(greet('fr'), 'Michael')  
Bonjour Michael  
>>>
```

# Аргументи, параметри та результати

```
>>> big = max('Hello world')  
>>> print(big)  
w
```

Аргумент → 'Hello world'

Параметр

```
def max(inp):  
    blah  
    blah  
    for x in inp:  
        blah  
        blah  
    return 'w'
```

→ 'w'  
Результат

# Кілька параметрів / аргументів

- Ми можемо **визначити** більше одного **параметра** у **функції**
- Ми просто додаємо кілька **аргументів** під час виклику **функції**
- Кількість і порядок аргументів й параметрів мають збігатися

```
def addtwo(a, b):  
    added = a + b  
    return added
```

```
x = addtwo(3, 5)  
print(x)
```

8

# Порожні (непродуктивні) функції

- Якщо функція не повертає значення, ми називаємо її «порожньою» («void»)
- Функції, що повертають значення – продуктивні
- Порожні функції – непродуктивні

# Створювати функції чи ні...

- Організуйте код за допомогою «абзаців» – зафіксуйте завершену думку та «назвіть її»
- Не повторюйтеся – змусьте код працювати, та повторно використовуйте цей код
- Якщо щось занадто складне або довге, поділіть це на логічні частини та створіть відповідні функції
- Зберіть бібліотеку типових речей (що ви робите знову й знову) – можливо, поділіться нею з друзями...

# Підсумки

- Функції
- Вбудовані функції
- Конвертування типів (int, float)
- Конвертування рядків
- Параметри
- Аргументи
- Результати (продуктивні функції)
- Порожні (непродуктивні) функції
- Навіщо використовувати функції?

## Вправа

Перепишіть ваш розрахунок оплати праці, щоб робітник отримував у 1,5 рази більше за понаднормову роботу, і створіть функцію **computePay**, яка отримуватиме два параметри (години й ставку)

Введіть години: 45

Введіть ставку: 10

Оплата: 475.0

Enter Hours: 45

Enter Rate: 10

Pay: 475.0

$$475 = 40 * 10 + 5 * 15$$





# Права власності / Застереження

Авторські права на ці слайди з 2010 року належать Чарльзу Северенсу ([www.dr-chuck.com](http://www.dr-chuck.com)) зі Школи інформації Мічиганського університету та застережені ліцензією Creative Commons Attribution 4.0. Будь ласка, збережіть цей фінальний слайд у всіх копіях документа, щоб відповідати вимогам ліцензії щодо посилань на джерела. При повторній публікації матеріалів, якщо щось зміните, додайте ім'я та організацію до переліку співавторів нижче.

Першоджерело: Чарльз Северенс, Школа інформації  
Мічиганського університету

Переклад: Платформа Prometheus