Python for Data Science

Лекція 4. Функції, генератори

Функції

- Функція — це блок організації коду, який може бути використана повторно
- Функції бувають іменовані та анонімні.
- First Class Functions. Функції у Python можуть бути використані як будь-який інший об’єкт. Їх можна передавати в інші функції в якості аргументів, зберігати у змінних та контейнерах тощо.

Іменовані функції

- Синтаксис оголошення іменованої функції:

```
def my_function(arg):
    # code here
```

- У Python прийнято іменувати функції у snake_case

Анонімні функції

- Синтаксис оголошення анонімної функції:

```
lambda args: expression
```

- Приклад:

```
x = lambda a: a * 3
print(x(10))  # 30
```

Виклик функції

- Для виклику функції використовуються круглі дужки після назви функції, у дужках вказуються аргументи:

```
print("String is an argument for print function")
```

- Значення, яке повертає функція, можна присвоїти:

```
int_variable = int("1")  # Returns int value
```

Повернення значень

- Повернення значень функцією відбувається за допомогою ключового слова `return`:

```
def my_function(args):
    ...
    return value
```

Повернення значень

- Функції, які не повертають значення, повертають `None`:

```
def my_function(a):
    a += 1

print(my_function(1))   # None
```

Функції без тіла

- Ключове слово `pass` використовується для пустих функцій, які залишають просто для збереження структури коду:

```
def my_function():
    pass
```

Аргументи функцій

- Функція без аргументів:

```
def my_function():
    ...
```

- Позиційні аргументи:

```
def my_function(a, b):
    ...
```

Аргументи функцій

- Значення аргументів за замовчуванням:

```
def my_function(a=1, b=2, c=3):
    ...
```

Аргументи функцій

- Список аргументів:

```
def my_function(*args):
    for i in args:
        print(i)
```

Аргументи функцій

- Аргументи типу ключ-значення (словарь аргументів):

```
def my_function(**kwargs):
    for k, v in kwargs.items():
        print(k, v)
```

Keyword-only аргументи

- За допомогою символу asterisk (*) після позиційних аргументів, можна зробити наступні аргументи keyword-only:

```
def my_function(a, b, *, flag=True):
    ...
```

Області видимості

У Python є три типи областей видимості для змінних:

- **Локальні** — ті змінні, що використовуються на одному рівні вкладеності, наприклад у поточній функції

- **Глобальні** — ключове слово `global` дозволяє звернутися до змінної, що була оголошена поза поточної функції

- **Нелокальні** — ключове слово `nonlocal` дозволяє звернутися до змінної, яка знаходиться поза поточного контексту, але не є глобальною. Наприклад, коли одна функція вкладена в іншу і батьківська містить необхідно змінну

Генератори

- Генератор – це абстракція послідовності елементів

- Ключовою відмінністю від списку є те, що генератор не зберігає всі елементи, а генерує їх за запитом

Генератори

- Для створення ітераторів в Python існує окремий синтаксис
- Для реалізації генераторів використовується ключове слово `yield` (замість return)

```
def createGenerator(gen) :
    mylist = range(gen)
    for i in mylist :
        yield i*i
mygenerator = createGenerator(3) # створюємо генератор
print(mygenerator) # mygenerator є об’єктом!
for i in mygenerator:
    print(i)
```

Генератори

- Найпростіший спосіб перебрати елементи генератора — це цикл `for`
- За допомогою функції `next()` можна брати наступний елемент генератора
- Генератор можна легко перетворити у список:

```
list(range(10))
```

Рекурсія

- Рекурсивна функція — це функція, яка викликає саму себе
- Рекурсія — це альтернатива циклам
- Приклад рекурсії:

```
def recursion():
    password = input("Type pass: ")
    if password != "12345":
        recursion()
```

Задачі

- Напишіть функцію, яка приймає довільну кількість аргументів, перевіряє кожен з них та повертає адфп`False`, якщо якийсь з аргументів не є рядком. Якщо всі аргументи є рядками, то функція повертає `True`.

- Напишіть функцію, яка приймає один аргумент — сторону квадрата. Функція має повертати його площу, периметр та діагональ.

Задачі

- Напишіть функцію, яка приймає число менше 100 та за допомогою рекурсії збільшує його на 1, поки число не дорівнює 100. Не використовуйте цикли.

- Напишіть функцію, яка приймає два списки, об’єднує їх в один та повертає його.