Programowanie funkcyjne w Pythonie.

https://pl.wikipedia.org/wiki/Programowanie_funkcyjne

Podejście do programowanie (styl) który kładzie nacisk na tworzenie funkcji zamiast na zmianę stanu zmiennych. Koncetruje się bardziej na tym CO ma się wydarzyć a nie jak.

Python zazwyczaj nie jest pierwszym wyborem jeśli chodzi o ten styl pisania kodu - chociażby przez to, ze właściwie wszystko możemy w nim modyfikować, podczas gdy w programowaniu funkcyjnym raczej stawia się na niemodyfikowalność stanu (immutability).

Python jednak daje trochę możliwości, które umożlwiają pozanie tego podejścia np. to, że funkcje w nim możemy traktować jako wartości (first class functions - o czym często nieświadomie można przekonać się zaczynając przygodę z Pythonem i zapminając o dodaniu "()" po nazwie funkcji), to, że możemy zwracać funkcje jako wartości (higher order functions).

Lambda

- to (zazwyczaj) krótkie funkcje, którym nie przydzielamy nazwy. Najczęściej używane gdy w kodzie potrzebujemy na "juz i teraz" jakąś krótką funkcję (mieszczącą się np. w jednej linijce).

```
x = lambda a : a + 10
print(x(5))
```

```
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))

mytripler = myfunc(3)
```

```
print(mytripler(11))
```

Map

- umożliwia zastosowanie jakiejś funkcji na elementach kolekcji (czyli możemy zastąpić pętle). Zwraca iterator - więc jeśli chcemy finalnie mieć inny typ (np. listę) to musimy dokonać konwersji.

```
def kwadrat(x):
    return x * x


nums = [1, 2, 3, 4, 5]
do_kwadratu = map(kwadrat, nums)


print(list(do_kwadratu))
```

Filter

- umożliwia zastosowanie jakiejś funkcji na kolekcji celem odfiltrowania wartości. Czyli zwraca nam jedynie te wartości na których wykonana funkcja zwróciła `True`.

Reduce

- akumuluje wartości zebrane po wykonaniu funkcji na kolekcji

```
import functools


def silnia(n):
    return functools.reduce(lambda x, y: x * y, range(1, n + 1))
```

Odnośniki

https://www.w3schools.com/python/python_lambda.asp

Wersja #3
Utworzono 2025-08-01 17:23:08 UTC przez Przemek Jeske
Zaktualizowano 2025-08-01 17:50:28 UTC przez Przemek Jeske