1. Narzędzie umożliwiające utworzenie listy na podstawie innej
2. Zmniejszają czytelność kodu ale równocześnie umożliwiają zmniejszenie jego ilości
3. b = [i\*2 for i in a]
4. c = [i for i in a if i not in b]
5. c = [x for row in matrix for x in row ]
6. Specjalne funkcje, która zwraca kolejny rezultat I jednocześnie ponownie jest wyonywana od momentu,gdzie została zatrzymana.
7. Generatory mają specjalne słowo yield zamiast return. Yield w przeciwieństwie do returna nie powoduje całkowitego wyjścia z generatora. Przy kolejnych powtórzeniach program wraca do miejsca skąd wcześniej zwrócilismy wartość.

def generator():

num = 0

while True:

num +=2

yield num

generator\_instance = generator()

print(next(generator\_instance))

1. def generator(n):

x, y = 0, 1

for \_ in range(n):

x, y = y, x+y

yield(y)

gen\_instance = generator(5)

print(next(gen\_instance))

1. Funkcje anonimowe
2. Nie mają nazw i zajmują mniej kodu, lambda nie ma słowa return. Lambdy nie modyfikują przesyłanych wartości tylko zwracają ich zmodyfikowane kopie.
3. zmienna = lambda x: x%2

print(zmienna(5))

1. Kiedy daną funkcję potrzebujemy na krótki okres czasu
2. x = lambda a,b: a+b

print(x(4,6))

1. list =[2, 4, 6, 8]

lam = filter(lambda x: x>4, list)

print(list(lam))

1. list =[2, 4, 6, 8]

lam = map(lambda x: x+5, list)

print(list(lam))

1. from functools import reduce

list =[2, 4, 6, 8]

lam = reduce(lambda x,y: x\*y, list)

print(lam)