Politechnika Wrocławska

# Programowanie obiektowe

## W11OPA-SI0054L (INP001045L)

## rok akademicki 2023/24

## semestr zimowy

# Laboratorium 3


Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220


Politechnika Wrocławska

# Plan

- Wyrażenia listowe
- Wyrażenia generujące
- Funkcje anonimowe
- Domknięcie

Politechnika Wrocławska

# Wyrażenie listowe

- Wyrażenia listowe (comprehension list) pozwalają w łatwy sposób generować listy
- W wielu przypadkach wyrażenie listowe może zastąpić pętlę for

Politechnika Wrocławska

# Wyrażenie listowe

- Przykłady

comprehension_list.py

```
1   # -*- coding: utf-8 -*-
2   """
3   Przykład użycia wyrażenia listowego
4   (comprehension list, listy składanej)
5   do utworzenia listy.
6   """
7
8   square = []
9   for x in range(5):
10      square += [x*x]
11
12  print(square)
13
14  square2 = [x*x for x in range(5)]
15  print(square2)
16
```

```
[0, 1, 4, 9, 16]
[0, 1, 4, 9, 16]
```

Politechnika Wrocławska

# Wyrażenie listowe

- Przykłady

comprehension_list_02.py

```
# -*- coding: utf-8 -*-
"""
Przykład użycia wyrażenia listowego
(comprehension list, listy składanej)
do utworzenia listy, z wykorzystaniem instrukcji warunkowej.
"""

square = []
for x in range(5):
    if x % 2 == 0:
        square += [x*x]

print(square)

square2 = [x*x for x in range(5) if x % 2 == 0]
print(square2)

```

```
[0, 4, 16]
[0, 4, 16]
```

Politechnika Wrocławska

# Wyrażenie listowe

- Przykłady

comprehension_list_03.py

```python
# -*- coding: utf-8 -*-
"""
Przykłady użycia wyrażenia listowego
(comprehension list, listy składanej).

Przykład pochodzi ze strony internetowej:
    https://www.w3schools.com/python/python_lists_comprehension.asp
"""

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]
print(newlist)
input()

newlist = [x for x in fruits if x != "apple"]
print(newlist)
input()
```

Politechnika Wrocławska

# Wyrażenie listowe

- Przykłady

comprehension_list_03.py

```
16  newlist = [x for x in fruits if x != "apple"]
17  print(newlist)
18  input()
19
20  newlist = [x for x in fruits]
21  print(newlist)
22  input()
23
24  newlist = [x.upper() for x in fruits]
25  print(newlist)
26  input()
27
28  newlist = ['hello' for x in fruits]
29  print(newlist)
30  input()
31
32  newlist = [x if x != "banana" else "orange" for x in fruits]
33  print(newlist)
34  input()
```

Politechnika Wrocławska

# Wyrażenie listowe

- Wyrażenie listowe można wykorzystać do generowania listy list

comprehension_list_04.py

```
1   # -*- coding: utf-8 -*-
2   """
3   Przykłady użycia wyrażenia listowego
4   do utworzenia listy list.
5   """
6
7   matrix = [[i + j for j in range(5) ] for i in range(7)]
8   print(matrix)
9
10  matrix2 = [["*" for j in range(i)] for i in range(5)]
11  print(matrix2)
```

Politechnika Wrocławska

# Wyrażenie listowe

- Wyrażenie listowe można wykorzystać do generowania listy list

comprehension_list_04.py

```
1   # -*- coding: utf-8 -*-
2   """
3   Przykłady użycia wyrażenia listowego
4   do utworzenia listy list.
5   """
6
7   matrix = [[i + j for j in range(5) ] for i in range(7)]
8   print(matrix)
9
10  matrix2 = [["*" for j in range(i)] for i in range(5)]
11  print(matrix2)
```

```
[[0, 1, 2, 3, 4], [1, 2, 3, 4, 5], [2, 3, 4, 5, 6], [3, 4, 5, 6, 7],
[4, 5, 6, 7, 8], [5, 6, 7, 8, 9], [6, 7, 8, 9, 10]]
[[], ['*'], ['*', '*'], ['*', '*', '*'], ['*', '*', '*', '*']]
```

Politechnika Wrocławska

# Wyrażenie generujące

- Wyrażenie generujące może być zastosowane, gdy wszystkie elementy listy nie muszą być znane jednocześnie
- Wyrażenie generujące objęte jest nawiasami zwykłymi ()

Politechnika Wrocławska

# Wyrażenie generujące

- Przykład

generator_expressions_01.py

```
# -*- coding: utf-8 -*-
"""
Przykładowy skrypt pozwalający porównać
wyrażenia listowe oraz wyrażenia generujące
"""

#lista kwadratów liczb
list1 = [x*x for x in range(10)]
print(list1)
#suma elementów tej listy
print(sum(list1))

#suma elementów listy utworzonej przez wyrażenie listowe
print(sum([x*x for x in range(10)]))

#suma obliczona z wykorzystaniem wyrażenia generującego
print(sum((x*x for x in range(10))))

#wyrażenie generujące
g = (x*x for x in range(10))

```

Politechnika Wrocławska

# Wyrażenie generujące

- Przykład

generator_expressions_02.py

```
# -*- coding: utf-8 -*-
"""
Przykład pokazujący działanie generatora
utworzonego przez wyrażenie generujące.
"""

#utworzenie generatora z wykorzystaniem wyrażenia generującego
g = (x*x for x in range(3))

#odczytanie wartości generatora
print(next(g))
input()

#kolejne odczytanie wartości generatora
print(next(g))
input()

#kolejne odczytanie wartości generatora
print(next(g))
input()

#to odczytanie wartości generatora skutkuje błędem
print(next(g))
```

Politechnika Wrocławska

# Wyrażenie generujące

- Przykład

generator_expressions_02.py

```
1   # -*- coding: utf-8 -*-
2   """
3   Przykład pokazujący działanie generatora
4   utworzonego przez wyrażenie generujące.
5   """
6
7   #utworzenie generatora z wykorzystaniem wyrażeni
8   g = (x*x for x in range(3))
9
10  #odczytanie wartości generatora
11  print(next(g))
12  input()
13
14  #kolejne odczytanie wartości generatora
15  print(next(g))
16  input()
17
18  #kolejne odczytanie wartości generatora
19  print(next(g))
20  input()
21
22  #to odczytanie wartości generatora skutkuje błędem
23  print(next(g))
```

```
0


1


4


Traceback (most recent call last):

  File "D:\repozytoria\python-examples\generatory
\generator_expressions_02.py", line 23, in <module>
    print(next(g))

StopIteration
```

Politechnika Wrocławska

# Funkcje anonimowe

- Mechanizm funkcji anonimowych pozwala zwięźle zapisać kod małych funkcji
- Definicja funkcji może zawierać pojedyncze wyrażenie
- Definicja funkcji anonimowej wykorzystuje słowo kluczowe **`lambda`**

Politechnika Wrocławska

# Funkcje anonimowe

- Przykład

lambda_01.py

```
# -*- coding: utf-8 -*-
"""
Przykład pokazujący użycie funkcji anonimowej.
"""

# definicja funkcji anonimowej (funkcja przypisana do zmiennej f)
f = lambda x : x*x

# wywołanie funkcji anonimowej (przechowywanej w zmiennej f)
for i in range(10):
    print(f(i))

# funkcja anonimowa zdefiniowana w miejscu wywołania
for i in range(10):
    print((lambda x : x*x)(i))
```

Politechnika Wrocławska

# Funkcje anonimowe

- Przykład

lambda_02.py

```
1   # -*- coding: utf-8 -*-
2   """
3   Przykład pokazujący wykorzystanie funkcji anonimowej
4   przyjmującej dwa argumenty.
5   """
6
7   # definicja funkcji anonimowej (funkcja przypisana do zmiennej f)
8   multipler = lambda x,y : x*y
9
10  # wywołanie funkcji anonimowej (przechowywanej w zmiennej f)
11  for i in range(10):
12      print(multipler(i,2))
13
```

Politechnika Wrocławska

# Funkcje anonimowe

- Przykład

lambda_03.py

```
# -*- coding: utf-8 -*-
"""
Przykład pokazujący funkcję anonimową
wykorzystujący instrukcję warunkową.
"""

# definicja funkcji anonimowej
# (funkcja przypisana do zmiennej smaller)
smaller = lambda x,y : x if x < y else y

# wywołanie funkcji anonimowej
for i in [1, 3, 5, 7, 4, 8, 2, 6]:
    print(smaller(i,5))
```

Politechnika Wrocławska

# Domknięcie

- Obiekt funkcji może być powiązany ze stanem otaczających go zasięgów
- Funkcję z pamięcią stanu nazywamy domknięciem

Politechnika Wrocławska

# Domknięcie

- Przykład

closure_01.py

```
# -*- coding: utf-8 -*-
"""
Przykład pokazujący wykorzystanie domknięcia.
"""

# funkcja multiplication przyjmuje mnożnik
# zwraca funkcję, która przyjmuje mnożną
# i oblicza wynik mnożenia
def multiplication(multiplier):
    def multipy(multiplicant):
        return multiplicant*multiplier

    return multipy

# funkcja multiplication dla mnożnika 2 zwraca
# funkcję podwającą
doubler = multiplication(2)
print(doubler(4))
```

Politechnika Wrocławska

# Domknięcie

- Przykład

```
15  # funkcja multiplication dla mnożnika 2 zwraca
16  # funkcję podwajającą
17  doubler = multiplication(2)
18  print(doubler(4))
19
20  # funkcja multiplication dla mnożnika 2 zwraca
21  # funkcję potrajającą
22  tripler = multiplication(3)
23  print(tripler(7))
```

Politechnika Wrocławska

# Domknięcie

- Przykład z wykorzystaniem funkcji anonimowej

closure_02.py

```
1   # -*- coding: utf-8 -*-
2   """
3   Przykład pokazujący wykorzystanie domknięcia.
4   """
5
6   # funkcja multiplication przyjmuje mnożnik
7   # zwraca funkcję, która przyjmuje mnożną
8   # i oblicza wynik mnożenia
9   def multiplication(multiplier):
10      return lambda multiplicant: multiplicant*multiplier
11
12  # funkcja multiplication dla mnożnika 2 zwraca
13  # funkcję podwającą
14  doubler = multiplication(2)
15  print(doubler(4))
```

Politechnika Wrocławska

# Podsumowanie

- Wyrażenia listowe
- Wyrażenia generujące
- Funkcje anonimowe
- Domknięcie