Laboratorium **Programowanie w języku Python 2** Wydział Elektrotechniki Automatyki I Informatyki Politechnika Świętokrzyska	
Studia: **Stacionarne I stopnia**	Kierunek: **Informatyka**
Data wykonania: **04.03.2021**	Grupa: **3ID16B**
Imię I nazwisko: **Arkadiusz Więcław**	Temat ćwiczenia: **Programowanie funkcyjne** **List comprehension**

Zad 1:

Przykład 1.1

```
# wyswietla  w postaci listy  wartosc podniesione do kwadratu listy o
nazwie x uzywa lambda
square = lambda x: x * x
x = [1, 2, 3, 4, 5]
print("\nPrzykład 1.1 = " ,list(map(lambda num: num * num, x)))
```

Wynik:

```
Przykład 1.1 =  [1, 4, 9, 16, 25]
```

Przykład 1.2

```
# wyswietla  w postaci listy  wartosc podniesione do kwadratu listy o
nazwie x uzywa funkcji
x = [1, 2, 3, 4, 5]
def square(num):
    return num * num
print("\nPrzykład 1.2 = " ,list(map(square, x)))
```

Wynik:

```
Przykład 1.2 =  [1, 4, 9, 16, 25]
```

Przykład 1.3

```
# wyswietla  w postaci listy  wartosc podniesione do kwadratu listy o
nazwie x tworzy zmienna square  ktore przechowuje  wyrazanie lambda
x = [1, 2, 3, 4, 5]
print("\nPrzykład 1.3 = " ,list(map(lambda num: num * num, x)))
```

Wynik:

```
Przykład 1.3 =  [1, 4, 9, 16, 25]
```

Przykład 1.4

```
#do zmiennej sa przypiswane  wyniki  mnozanie obecnej wartosc
zmiennej product z zmienna num
product = 1
x = [1, 2, 3, 4]
for num in x:
    product = product * num
print("\nPrzykład 1.4 = " ,product)
```

Wynik:

```
Przykład 1.4 =  24
```

Przykład 1.5

```
# licz to samo co 4 przykladzie zamiast zwyklej petli wykorzystuje
funckje reduce wraz
# z wyrazeniem lambda na poczatku 1 * 2 wynik tego jest nastepnie jest
mnozony przez koljene liczbe na liscie i tak dalej az lista sie
skonczy
from functools import reduce
product = reduce((lambda x, y: x * y), [1, 2, 3, 4])
print("\nPrzykład 1.5 = " ,product)
```

Wynik:

```
Przykład 1.5 =  24
```

Przykład 1.6

```
from functools import reduce
lis = [ 1 , 3, 5, 6, 2 ]
output = reduce(lambda a,b : a if a > b else b, lis)
print("\nPrzykład 1.6 = " ,output)
```

Wynik:

```
Przykład 1.6 =  6
```

Przykład 1.7

```
#do zmiennej  x jest przypisywana  wartosc od -5 do 5
#nastepnie jest tworzona nowa pusta lista
#nastepnie w petli sa sprawdzane czy wartosc przypisane do x sa
mniejsze od 0 jesli tak sa
#przypisywane do wczesniej utworzonej listy
x = range(-5, 5)
new_list = []
for num in x:
    if num < 0:
        new_list.append(num)
print("\nPrzykład 1.7 = ",new_list)
```

Wynik:

```
Przykład 1.7 =  [-5, -4, -3, -2, -1]
```

Przykład 1.8

```
#do zmienej x sa przyspisywane wartosc od - 5 do 5
#nastepnie w zmiennej  all_less_than_zero jest zapisywane w postaci
list  wyniki metody
#filter w której znajduje sie  wyrazenie ktora sprawdza czy wartosc sa
mniejsze od 0 jesli tak to przypisuje je do listy
x = range(-5, 5)
all_less_than_zero = list(filter(lambda num: num < 0, x))
print("\nPrzykład 1.8 = ",all_less_than_zero)
```

Wynik:

```
Przykład 1.8 =  [-5, -4, -3, -2, -1]
```

Przykład 1.9

```
# tworzy 2 tuple i zwraca obiekt zip z nich
a = ("John", "Charles", "Mike")
b = ("Jenny", "Christy", "Monica", "Vicky")
x = zip(a, b)
print("\nPrzykład 1.9 = ",*x)
```

Wynik:

```
Przykład 1.9 =  ('John', 'Jenny') ('Charles', 'Christy') ('Mike',
'Monica')
```

Przykład 1.10

```
#wykorzystuja reduce wyswietlane jest wynik  calkowite sumy elementow
zawarty w przekazanej liscie
# funckja sum robi to samo co linika wyzej
import functools, operator
w = functools.reduce(operator.add, [1, 2, 3, 4], 0)
sum([1, 2, 3, 4])
print("\nPrzykład 1.10 = ", w)
```

Wynik:

```
Przykład 1.10 =  10
```

Przykład 1.11

```
#petla iteruje po liscie w czasie dzialanie petli kolejnych iteracjach
petli przypisywane
#sa kolejne wynik mnozenia wartosc zmniennej product z i
import functools
product = 1
for i in [1, 2, 3]:
    product *= i
print("\nPrzykład 1.11 = ", product)
```

Wynik:

```
Przykład 1.11 =  6
```

Przykład 1.12

```
#jak w przykladzie ex_11  tylko ze wszystko jest wykonywane w jednej
#linijce  wykorzystuja funkcje reduce i operator mnozenie ktory bedzie
mnozyl kolejne elementy  listy
product = functools.reduce(operator.mul, [1, 2, 3], 1)
print("\nPrzykład 1.12 = ", product)
```

Wynik:

```
Przykład 1.12 =  6
```

Przykład 2.1

```
#program wykorzystuje list comprehension  zwraca potegi liczb
zawartych w liscie
print("\nPrzykład 2.1 = " ,[x * x for x in [1, 2, 3, 4]])
```

Wynik:

```
Przykład 2.1 =  [1, 4, 9, 16]
```

Przykład 2.2

```
#program  wyswietla liczby mniejsze od 0   najpierw  wykorzystujac
list comprehension i programowanie funkcyjne
x = range(-5, 5)
all_less_than_zero = list(filter(lambda num: num < 0, x))
print("\nPrzykład 2.2 = " ,all_less_than_zero)
x = range(-5, 5)
all_less_than_zero = [num for num in x if num < 0]
```

Wynik:

```
Przykład 2.2 =  [-5, -4, -3, -2, -1]
```

Przykład 2.3

```
#wypisuje kolejne indeksy macierzy, gdzie 1-wszy indeks ma byd
parzysty z zakresu 0-9, a 2-gi indeks ma byd z zakresu 0-7.
from matplotlib import pylab
from pylab import *
a=zeros((10,8))
a[::2,:]=1
b=zeros(a.shape)
print("\nPrzykład 2.3 = " ,)
for (y,x) in [(y,x) for y in range(a.shape[0]) for x in
    range(a.shape[1]) if a[y,x]>0.5]:
    print(y,x)
b[y,x]=a[y,x]
imshow(b,interpolation='none',cmap=cm.gray)
```

Wynik:

```
Przykład 2.3 =

0 0,0 1,0 2,0 3,0 4,0 5,0 6,0 7,2 0,2 1,2 2,2 3,2 4,2 5,

2 6,2 7,4 0,4 1,4 2,4 3,4 4,4 5,4 6,4 7,6 0,6 1,6 2,6 3,6 4,6 5,6 6,6
7,8 0,8 1,8 2,8 3,8 4,8 5,8 6,8 7.
```

Przykład 2.4

```
#program dzieli zagniezdzona liste  na mniejsze czesci
transposed = []
matrix = [[1, 2, 3, 4], [4, 5, 6, 8]]
for i in range(len(matrix[0])):
    transposed_row = []
for row in matrix:
    transposed_row.append(row[i])
transposed.append(transposed_row)
print("\nPrzykład 2.4 = " ,transposed)
```

Wynik:

```
Przykład 2.4 =  [[4, 8]]
```

Przykład 2.5

```
#dzieli macierz na elementy nieparzyste oraz parzyste, modyfikując
macierz.
matrix = [[1, 2], [3,4], [5,6], [7,8]]
transpose = [[row[i] for row in matrix] for i in range(2)]
print ("\nPrzykład 2.5 = ", transpose)
```

Wynik:

```
Przykład 2.5 =  [[1, 3, 5, 7], [2, 4, 6, 8]]
```

Przykład 2.6

```
#Wypisuje pierwiastki liczb mniejszych od 0 z podanego zakresu.
x = range(-5, 5)
all_less_than_zero = list(map(lambda num: num * num,
list(filter(lambda num: num < 0, x))))
print ("\nPrzykład 2.6 = ", all_less_than_zero)
```

Wynik:

```
Przykład 2.6 =  [25, 16, 9, 4, 1]
```

Przykład 2.7

```
#wypisuje pierwiastki liczb mniejszych od 0 z podanego zakresu za
pomocą funkcji anonimowej.
x = range(-5, 5)
all_less_than_zero = [num * num for num in x if num < 0]
print ("\nPrzykład 2.7 = ", all_less_than_zero)
```

Wynik:

```
Przykład 2.7 =  [25, 16, 9, 4, 1]
```

Przykład 2.8

```
#wypisuje “orange”, jeśli napotka “apple”, a następnie szuka słów z
literami a i e lub k.
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
print ("\nPrzykład 2.8 = ", [(x if x != "apple" else "orange") for x
in fruits if 'a' in x and 'e' in x or 'k' in x])
```

Wynik:

```
Przykład 2.8 =  ['orange', 'kiwi']
```

Przykład 2.9

```
#przyklad wyswietla na dwa sposoby elementy listy zaczynac kazdy z
nich z  duzej litery
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
print ("\nPrzykład 2.9 = ")
print ([f[0].upper()+f[1:] for f in fruits])
print ([f.capitalize() for f in fruits])
```

Wynik:

```
Przykład 2.9 =
```

```
['Apple', 'Banana', 'Cherry', 'Kiwi', 'Mango']
['Apple', 'Banana', 'Cherry', 'Kiwi', 'Mango']
```

Przykład 2.10

```
#wypisuje nazwę kraju, a obok jego numer.
DIAL_CODES = [ (86, 'China'),
(91, 'India'),
(1, 'United States'),
(62, 'Indonesia'),
(55, 'Brazil'),
(92, 'Pakistan'),
(880, 'Bangladesh'),
(234, 'Nigeria'),
(7, 'Russia'),
(81, 'Japan'),
]
country_code = {country: code for code, country in DIAL_CODES}
print ("\nPrzykład 2.10 = ")
print (country_code)
print ({code: country.upper() for country, code in
country_code.items() if code < 66})
```

Wynik:

```
Przykład 2.10 =
{'China': 86, 'India': 91, 'United States': 1, 'Indonesia': 62,
'Brazil': 55, 'Pakistan': 92, 'Bangladesh': 880, 'Nigeria': 234,
'Russia': 7, 'Japan': 81}
{1: 'UNITED STATES', 62: 'INDONESIA', 55: 'BRAZIL', 7: 'RUSSIA'}
```

Przykład 2.11

```
#wypisuje liczbę oraz jej pierwiastek dla liczb z zakresu 1-10.
square_dict = dict()
for num in range(1, 11):
    square_dict[num] = num*num
print ("\nPrzykład 2.11 = ")
print(square_dict)
square_dict = {num: num*num for num in range(1, 11)}
print(square_dict)
```

Wynik:

```
Przykład 2.11 =
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Zad 2:

Przykład 1.1 wlasny

```
square1 = lambda x, y: x * y
x = [1, 2, 3, 4, 5, 1]
y = [6, 7, 8, 9, 1, 2]
print("\nPrzykład 1.1 wlasny = ", list(map(lambda num1, num2: num1 *
num2, x, y)))
```

Wynik:

```
Przykład 1.1 wlasny =  [6, 14, 24, 36, 5, 2]
```

Przykład 1.2 wlasny

```
x = [1, 2, 3, 4, 5, 1]
y = [6, 7, 8, 9, 1, 2]
def square(num1,num2):
    return num1 * num2
print("\nPrzykład 1.2 wlasny = " ,list(map(square, x, y)))
```

Wynik:

```
Przykład 1.2 wlasny =  [6, 14, 24, 36, 5, 2]
```

Przykład 1.3 wlasny

```
x = [1, 2, 3, 4, 5, 1]
y = [6, 7, 8, 9, 1, 2]
print("\nPrzykład 1.3 wlasny = " ,list(map(lambda num1, num2: num1 *
num2, x, y)))
```

Wynik:

```
Przykład 1.3 wlasny =  [6, 14, 24, 36, 5, 2]
```

Przykład 1.4 wlasny

```
product = 1
x = [5, 2, 3, 4]
for num in x:
    product = product - num
```

```
print("\nPrzykład 1.4 wlasny = " ,product)
```
Wynik:

```
Przykład 1.4 wlasny =  -13
```

Przykład 1.5 wlasny

```
from functools import reduce
product = reduce((lambda x, y: x - y), [5, 2, 3, 4])
print("\nPrzykład 1.5 wlasny = " ,product)
```
Wynik:

```
Przykład 1.5 wlasny =  -4
```

Przykład 1.6 wlasny

```
from functools import reduce
slowa = ["wawa", "ja", "aa", "mghjk", "Jenny", "Christy", "Monicahj",
"Vicky"]
naj_slowo = reduce(lambda slowo1, slowo2: slowo1 if len(slowo1) >
len(slowo2) else slowo2, slowa)
print("\nPrzykład 1.6 wlasny = ", naj_slowo)
```
Wynik:

```
Przykład 1.6 wlasny =  Monicahj
```

Przykład 1.7 wlasny

```
x = ("hello", "the", "superman", "ale", "ac")
new_tuples = []
for word in x:
    if len(word) < 4:
        new_tuples.append(word)
print("\nPrzykład 1.7 wlasny = ",new_tuples)
```
Wynik:

```
Przykład 1.7 wlasny =  ['the', 'ale', 'ac']
```

Przykład 1.8 wlasny

```
words = ["hello", "the", "superman", "ale", "ac"]
all_words = list(filter(lambda w: len(w) > 2, words))
print("\nPrzykład 1.8 wlasny = ",all_words)
```
Wynik:

```
Przykład 1.8 wlasny =  ['hello', 'the', 'superman', 'ale']
```

Przykład 1.9 wlasny

```
imiona = ["Arkadiusz", "Radoslaw", "Mateusz", "Maciej"]
nazwiska = ["Nowak ", "Mnich", "Kowalski"]
nick = ["Lud123", "Rango", "Rudy102"]
lista_osob = list(zip(imiona, nazwiska, nick))
print("\nPrzykład 1.9 wlasny = ",lista_osob)
```

Wynik:

```
Przykład 1.9 wlasny =  [('Arkadiusz', 'Nowak ', 'Lud123'),
('Radoslaw', 'Mnich', 'Rango'), ('Mateusz', 'Kowalski', 'Rudy102')]
```

Przykład 1.10 wlasny

```
import functools, operator
print("\nPrzykład 1.10 wlasny = ", functools.reduce(operator.add,
["aa", "xx", "ccc", "as"], "bb"))
```

Wynik:

```
Przykład 1.10 wlasny =  bbaaxxcccas
```

Przykład 1.11 wlasny

```
import functools
liczba = 520
for i in [12, 42, 50, 90]:
    liczba -= i
print("\nPrzykład 1.11 wlasny = ", liczba)
```

Wynik:

```
Przykład 1.11 wlasny =  326
```

Przykład 1.12 wlasny

```
import functools, operator
odd = functools.reduce(operator.sub, [21, 32, 50], 213)
add = functools.reduce(operator.add, [1, 2, 3, 5, 6], 1)
print("\nPrzykład 1.12 wlasny = ", odd, add)
```

Wynik:

```
Przykład 1.12 wlasny =  110 18
```

Zad 3:

Przykład 2.1 wlasny

```
print("\nPrzykład 2.1 wlasny = " ,[x + x for x in [1, 2, 3, 4, 5, 6, 7]])
```

Wynik:

```
Przykład 2.1 wlasny =  [2, 4, 6, 8, 10, 12, 14]
```

Przykład 2.2 wlasny

```
words = ["hello", "the", "superman", "ale", "ac"]
print("\nPrzykład 2.2 wlasny = ")
print("za pomoca programowania funk : ", (list(filter(lambda wr1: len(wr1) > 3, words))))
print("za pomoca list komprahetnych: ", [x for x in words if len(x) > 3])
```

Wynik:

```
Przykład 2.2 wlasny =

za pomoca programowania funk :  ['hello', 'superman']

za pomoca list komprahetnych:  ['hello', 'superman']
```

Przykład 2.3 wlasny

```
lista = [(x, y) for (y, x) in [(y, x) for y in range(5) for x in range(4) if x > 0 and y < 8]]
print("\nPrzykład 2.3 wlasny = " ,lista)
```

Wynik:

```
Przykład 2.3 wlasny =  [(1, 0), (2, 0), (3, 0), (1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3), (1, 4), (2, 4), (3, 4)]
```

Przykład 2.4 wlasny

```
polaczone = []
nazwiska_nick = [["Nowak ", "Mnich", "Kowalski"], ["Lud123", "Rango",
"Rudy102"]]
for x in range(len(nazwiska_nick[0])):
    polaczone_kolumna = []
    for kolumna in nazwiska_nick:
        polaczone_kolumna.append(kolumna[x])
    polaczone.append(polaczone_kolumna)
print("\nPrzykład 2.4 wlasny = " ,polaczone)
```

Wynik:

```
Przykład 2.4 wlasny =  [['Nowak ', 'Lud123'], ['Mnich', 'Rango'],
['Kowalski', 'Rudy102']]
```

Przykład 2.5 wlasny

```
niepolaczone = [ ['W', 'Z'], ['A', 'A'], ['R', 'W' ], ['S', 'A'] ]
polaczone = [[kolumna[x] for kolumna in niepolaczone] for x in
range(2)]
print("\nPrzykład 2.5 wlasny = " ,polaczone)
```

Wynik:

```
Przykład 2.5 wlasny =  [['W', 'A', 'R', 'S'], ['Z', 'A', 'W', 'A']]
```

Przykład 2.6 wlasny

```
words = ["hello", "the", "superman", "ale", "ac"]
print("\nPrzykład 2.6 wlasny = " )
print("przed: ", words)
slowa_wieksze = [r1 + r1 for r1 in words if len(r1) > 3]
print(slowa_wieksze)
```

Wynik:

```
Przykład 2.6 wlasny =

przed:  ['hello', 'the', 'superman', 'ale', 'ac']

['hellohello', 'supermansuperman']
```

Przykład 2.7 wlasny

```
r1 = range(-10, 11)
parzyste = [liczba + liczba*2 for liczba in r1 if liczba%2==0]
print("\nPrzykład 2.7 wlasny = " ,parzyste)
```

Wynik:

```
Przykład 2.7 wlasny =  [-30, -24, -18, -12, -6, 0, 6, 12, 18, 24, 30]
```

Przykład 2.8 wlasny

```
print("\nPrzykład 2.8 wlasny = " ,[(x * x  if x > 6  else  x + x) for
x in [x for x in range(23)]  if x < 12])
```

Wynik:

```
Przykład 2.8 wlasny =  [0, 2, 4, 6, 8, 10, 12, 49, 64, 81, 100, 121]
```

Przykład 2.9 wlasny

```
imiona = ["arkadiusz", "radoslaw", "mateusz", "maciej", "karol" ]
print("\nPrzykład 2.9 wlasny = ")
print("przed: ", imiona)
print("1 sposob: ", [i[0].upper() + i[1:].upper() for i in imiona])
print("2 sposob: ", ([i.upper() for i in imiona]))
```

Wynik:

```
Przykład 2.9 wlasny =

przed:  ['arkadiusz', 'radoslaw', 'mateusz', 'maciej', 'karol']

1 sposob:  ['ARKADIUSZ', 'RADOSLAW', 'MATEUSZ', 'MACIEJ', 'KAROL']

2 sposob:  ['ARKADIUSZ', 'RADOSLAW', 'MATEUSZ', 'MACIEJ', 'KAROL']
```

Przykład 2.10 wlasny

```
WAGA = [
(76, 'Tomek'),
(65, 'Ania'),
(89, 'Romek'),
(93, 'Piotrek'),
(76, 'Sylwia'),
(60, 'Natalia'),
(80, 'Marcin'),
(69, 'Teresa'),
(86, 'Mikolaj')
]
waga_osob = {waga: ile for ile, waga in WAGA }
waga_osob
{ile: waga.upper() for waga, ile in waga_osob.items() if ile < 180}
print("\nPrzykład 2.10 wlasny = " ,waga_osob)
```

Wynik:

```
Przykład 2.10 wlasny =  {'Tomek': 76, 'Ania': 65, 'Romek': 89,
'Piotrek': 93, 'Sylwia': 76, 'Natalia': 60, 'Marcin': 80, 'Teresa':
69, 'Mikolaj': 86}
```

Przykład 2.11 wlasny

```
slownik = dict()
for licznik in range(1, 10):
    if licznik % 2:
        slownik[licznik] = licznik * licznik
    else:
        slownik[licznik] = licznik + licznik
print("\nPrzykład 2.11 wlasny = ", {licznik: licznik - licznik if
licznik % 2 else licznik + licznik for licznik in range(5, 15)})
```

Wynik:

```
Przykład 2.11 wlasny =  {5: 0, 6: 12, 7: 0, 8: 16, 9: 0, 10: 20, 11:
0, 12: 24, 13: 0, 14: 28}
```

Zad 4:

Obliczanie sumy elementów nieparzystych z wybranego zakresu:

```
print("\nProgramowanie funkcyjne = ")
from functools import reduce
list1 = [1,2,3,4,5,7,8,9,10]
print("\nWynik = ", reduce(lambda x,y: x +y,(list(filter(lambda x:  x
% 2 , list1[int(input(" Poczatek ")):int(input(" Koniec "))] )))))

print("\nComprehension list = ")
print("\nWynik = ", sum([ x for x in list1[int(input(" Poczatek
")):int(input(" Koniec "))] if x % 2]))
```

Wynik:

```
Obliczanie sumy elementów nieparzystych z wybranego zakresu =

Programowanie funkcyjne =

Poczatek 1

Koniec 10

Wynik =  24

Comprehension list =

Poczatek 1

Koniec 10

Wynik =  24
```

Wyliczające sumę elementów, których wartości spełniają zadany warunek:

```
print("\nProgramowanie funkcyjne = ")
print("\nWynik = ",reduce(lambda x,y: x + y,(list(filter(lambda x: x
<= 5 ,[23,1,4,5,9,2,13,22,31] )))))

print("\nComprehension list = ")
print("\nWynik = ", sum([num for num in [23,1,4,5,9,2,13,22,31] if num
<= 5 ]))
```

Wynik:

```
Programowanie funkcyjne =

Wynik =  12

Comprehension list =

Wynik =  12
```