Politechnika Wrocławska

Podstawy programowania W11OPA-SI0072G
Wstęp do programowania W11FTE-SI0141WL
Wstęp do programowania W11IKW-SI0080WL

rok akademicki 2024/25
semestr letni

Wykład 3

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 221

Politechnika Wrocławska

Plan prezentacji

- Funkcja print() – kontynuacja
- Formatowanie liczb
- Argumenty: pozycyjne, nazwane, domyślne
- Funkcje: zwracanie wartości

- Algorytmy iteracyjne

- Pętle

Politechnika Wrocławska

Funkcja print()

print03.py

```
1  #Program pokazuje wypisywanie kilku wierszy przy użyciu funkcji print()
2  def main():
3      print('Raz') #po wypisaniu napisu kursor jest przenoszony
4      print('Dwa') #do nowego wiersza
5      print('Trzy')
6
7  main()
8
9
```

Console 1/A

```
Raz
Dwa
Trzy
```

Politechnika Wrocławska

Funkcja print()

print03.py print04.py

```
1  #Program pokazuje jak zmienić domyślne zachowanie funkcji print()
2  #Napisy będą wyświetlone w tym samym wierszu, rozdzielone spacjami
3  def main():
4      print('Raz', end=' ') #po wypisaniu napisu kursor nie jest przenoszony
5      print('Dwa', end=' ') #do nowego wiersza, jest wstawiana spacja
6      print('Trzy')
7
8  main()
9
```

Console 1/A

```
Raz Dwa Trzy
```

Politechnika Wrocławska

Funkcja print()

print03.py print04.py print05.py

```
1  #Program pokazuje jak zmienić domyślne zachowanie funkcji print()
2  #Napisy będą wyświetlone w tym samym wierszu.
3  def main():
4      print('Raz', end='') #po wypisaniu napisu kursor pozostaje w miejscu
5      print('Dwa', end='') #kolejne wywołanie wypisuje następny napis
6      print('Trzy')
7
8  main()
9
```

Console 1/A

```
RazDwaTrzy
```

Politechnika Wrocławska

Funkcja print()

print03.py print04.py print05.py print06.py

```
1  #Program pokazuje domyślne zachowanie funkcji print()
2  # wywoływanej z kilkoma argumentami.
3  def main():
4      print('Raz','Dwa','Trzy') #napisy są rozdzielone spacjami
5
6  main()
7
```

Console 1/A

```
Raz Dwa Trzy
```

Politechnika Wrocławska

Funkcja print()

print03.py print04.py print05.py print06.py print07.py

```
1  #Program pokazuje jak zmienić domyślne zachowanie funkcji print()
2  # wywoływanej z kilkoma argumentami.
3  def main():
4      print('Raz','Dwa','Trzy',sep='***') #napisy są rozdzielone gwiazdkami
5
6  main()
7
```

Console 1/A

```
Raz***Dwa***Trzy
```

Politechnika Wrocławska

Funkcja print()

Znaki sterujące

print08.py print09.py

```
1  #Program pokazuje wykorzystanie znaków sterujących \n, \t, \', \", \\
2  def main():
3      print('To jest napis') #Funkcja print() wyświetli napis
4
5      #Znak sterujący \n powoduje przeniesienie kursora do nowej linii
6      print('To\njest\nnapis')
7
8      #Znak sterujący \t powoduje przejście do następnego położenia tabulatora
9      print('To\tjest\tnapis')
10     #Może to być pomocne przy formatowaniu nagłówka tabeli
11     print('pn\twt\tśr\tcz\tpt')
12
13     #Znaki specjalne \' oraz \" pozwalają uzyskać apostrof i cudzysłów
14     print('Apostrof: \' ')
15     print('Cudzysłów: \" ')
16
17     #Skoro znaki sterujące wykorzystują ukośnik
18     #To jak uzyskać ukośnik?
19     print('Ukośnik: \\')
20
21 main()
```

Politechnika Wrocławska

Funkcja print()

Znaki sterujące

print08.py print09.py

```
1  #Program pokazuje wykorzystanie znaków sterujących \n, \t, \', \", \\
2  def main():
3      print('To jest napis') #Funkcja print() wyświetli n
4
5      #Znak sterujący \n powoduje przeniesienie kursora d
6      print('To\njest\nnapis')
7
8      #Znak sterujący \t powoduje przejście do następnego
9      print('To\tjest\tnapis')
10     #Może to być pomocne przy formatowaniu nagłówka tab
11     print('pn\twt\tśr\tcz\tpt')
12
13     #Znaki specjalne \' oraz \" pozwalają uzyskać apost
14     print('Apostrof: \' ')
15     print('Cudzysłów: \" ')
16
17     #Skoro znaki sterujące wykorzystują ukośnik
18     #To jak uzyskać ukośnik?
19     print('Ukośnik: \\')
20
21 main()
```

Console 1/A

```
To jest napis
To
jest
napis
To  jest    napis
pn  wt  śr  cz  pt
Apostrof: '
Cudzysłów: "
Ukośnik: \
```

Politechnika Wrocławska

Funkcja print()

Konkatencja napisów

print08.py print09.py

```
1  #Program pokazuje konkatencję ciągów tekstowych.
2
3  def main():
4      #Dwa napisy zostaną połączone w jeden.
5      print('To jest ' + 'jeden ciąg tekstowy.')
6
7      #Trzy napisy zostaną połączone, mimo tego,
8      #że znajdują się w kilku liniach.
9      print('To jest bardzo długi napis, ' +
10           'który trudno byłoby zmieścić ' +
11           'w jednej linii.')
12
13 main()
```

Console 1/A

```
To jest jeden ciąg tekstowy.
To jest bardzo długi napis, który trudno byłoby zmieścić w jednej linii.
```

Politechnika Wrocławska

Formatowanie liczb

print10.py | print10a.py | print11.py | print11a.py

```
1  #Program pokazuje domyślne formatowanie
2  #liczby zmiennoprzecinkowej.
3
4  def main():
5      kredyt = 5000.
6      rata_miesieczna = kredyt / 12
7      print('Miesięczna rata wynosi', rata_miesieczna, 'zł.')
8
9  main()
10
```

Console 1/A

```
Miesięczna rata wynosi 416.6666666666667 zł.
```

Politechnika Wrocławska

Formatowanie liczb

print10.py | print10a.py | print11.py | print11a.py

```
1  #Program pokazuje domyślne formatowanie
2  #liczby zmiennoprzecinkowej - użycie f-stringa
3
4  def main():
5      kredyt = 5000.
6      rata_miesieczna = kredyt / 12
7      print(f'Miesięczna rata wynosi {rata_miesieczna} zł.')
8
9  main()
10
```

Console 1/A

```
Miesięczna rata wynosi 416.6666666666667 zł.
```

Politechnika Wrocławska

Formatowanie liczb

print10.py | print10a.py | print11.py | print11a.py

```
1  #Program pokazuje formatowanie liczby zmiennoprzecinkowej
2  #przy użyciu funkcji format().
3
4  def main():
5      kredyt = 5000.
6      rata_miesieczna = kredyt / 12
7      #liczba zmiennoprzecinkowa (f) jest zaokrąglona
8      #do dwóch (.2f) cyfr po przecinku
9      print('Miesięczna rata wynosi',
10           format(rata_miesieczna, '.2f'),
11           'zł.')
```

Console 1/A

```
Miesięczna rata wynosi 416.67 zł.
```

Politechnika Wrocławska

Formatowanie liczb

print10.py | print10a.py | print11.py | print11a.py

```
1  #Program pokazuje formatowanie
2  #liczby zmiennoprzecinkowej - użycie f-stringa
3
4  def main():
5      kredyt = 5000.
6      rata_miesieczna = kredyt / 12
7      #liczba zmiennoprzecinkowa (f) jest zaokrąglona
8      #do dwóch (.2f) cyfr po przecinku
9      print(f'Miesięczna rata wynosi {rata_miesieczna: .2f} zł.')
10
11 main()
```

Console 1/A

```
Miesięczna rata wynosi 416.67 zł.
```

Politechnika Wrocławska

Formatowanie liczb

print10.py | print10a.py | print11.py | print11b.py

```
1  #Program pokazuje formatowanie
2  #liczby zmiennoprzecinkowej - użycie f-stringa
3
4  def main():
5      kredyt = 5000.
6      rata_miesieczna = kredyt / 12
7      #liczba zmiennoprzecinkowa (f) jest zaokrąglona
8      #do dwóch (.2f) cyfr po przecinku
9      print(f'{rata_miesieczna=: .2f} zł.')
10
11 main()
```

Console 1/A

```
rata_miesieczna= 416.67 zł.
```

Politechnika Wrocławska

Formatowanie liczb

Przykłady

print10.py | print10a.py | print11.py | print11a.py | print12.py

```
#Program pokazuje efekt wyświetlenia kilku liczb
#zmiennoprzecinkowych w kolumnie według wspólnego formatu.
#Liczby są wyrównane względem przecinka dziesiętnego.

def main():
    num1 = 127.899
    num2 = 3465.148
    num3 = 3.776
    num4 = 264.821
    num5 = 88.081
    num6 = 799.999

    form = '7.2f'

    print(format(num1,form))
    print(format(num2,form))
    print(format(num3,form))
    print(format(num4,form))
    print(format(num5,form))
    print(format(num6,form))

main()
```

Politechnika Wrocławska

Formatowanie liczb

Przykłady

print10.py | print10a.py | print11.py | print11a.py | print12.py

```
1  #Program pokazuje efekt wyświetlenia kilku liczb
2  #zmiennoprzecinkowych w kolumnie według wspólnego formatu.
3  #Liczby są wyrównane względem przecinka dziesiętnego.
4
5  def main():
6      num1 = 127.899
7      num2 = 3465.148
8      num3 = 3.776
9      num4 = 264.821
10     num5 = 88.081
11     num6 = 799.999
12
13     form = '7.2f'
14
15     print(format(num1,form))
16     print(format(num2,form))
17     print(format(num3,form))
18     print(format(num4,form))
19     print(format(num5,form))
20     print(format(num6,form))
21
22 main()
```

```
 127.90
3465.15
   3.78
 264.82
  88.08
 800.00
```

Politechnika Wrocławska

Formatowanie liczb

Opcje formatowania

```
format_spec     ::=  [[fill]align][sign]["z"]["#"]["0"][width][grouping_option]["." precision]
fill            ::=  <any character>
align           ::=  "<" | ">" | "=" | "^"
sign            ::=  "+" | "-" | " "
width           ::=  digit+
grouping_option ::=  "_" | ","
precision       ::=  digit+
type            ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | '
```

https://docs.python.org/3/library/string.html#formatspec

Politechnika Wrocławska

Formatowanie liczb

Metoda `str.format()`

- Łańcuchy znakowe można również formatować wykorzystując metodę **`format()`** klasy **`str`**

print13.py

```
1  #Program pokazuje przykłady wykorzystania
2  #metody str.format() do formatowania napisów
3
4  def main():
5      # dostęp do argumentów według pozycji
6      print('{0}, {1}, {2}'.format('a', 'b', 'c'))
7      print('{}, {}, {}'.format('a', 'b', 'c'))
8      print('{2}, {1}, {0}'.format('a', 'b', 'c'))
9      print('{0}{1}{0}'.format('abra', 'cad'))   # indeksy argumentów mogą się powtarzać
10     input()
11
```

Console 1/A

```
a, b, c
a, b, c
c, b, a
abracadabra
```

Politechnika Wrocławska

Formatowanie liczb

Metoda `str.format()`

- Łańcuchy znakowe można również formatować wykorzystując metodę **`format()`** klasy **`str`**

print13.py

```
12      # dostęp do argumentów po nazwach
13      print('Coordinates: {latitude}, {longitude}'.format(latitude='37.24N', longitude='-115.81W'))
14      input()
15
16      # wyrównywanie i ustalanie szerokości pola
17      print('{:<30}'.format('wyrównanie do lewej'))
18      print('{:>30}'.format('wyrównanie do prawej'))
19      print('{:^30}'.format('wyśrodkowanie'))
20      print('{:*^30}'.format('wyśrodkowanie'))   # i wypełnienie '*'
21      input()
22
23      # pokazywanie znaku liczb dodatnich
24      print('{:+f}; {:+f}'.format(3.14, -3.14))  # pokazuj znak zawsze
```

Console 1/A

```
Coordinates: 37.24N, -115.81W

wyrównanie do lewej
          wyrównanie do prawej
        wyśrodkowanie
********wyśrodkowanie*********
```

Politechnika Wrocławska

Formatowanie liczb

Metoda `str.format()`

- Łańcuchy znakowe można również formatować wykorzystując metodę **`format()`** klasy **`str`**

print13.py

```
23      # pokazywanie znaku liczb dodatnich
24      print('{:+f}; {:+f}'.format(3.14, -3.14))  # pokazuj znak zawsze
25      print('{: f}; {: f}'.format(3.14, -3.14))  # spacja dla dodatnich
26      print('{:-f}; {:-f}'.format(3.14, -3.14))  # pokazuj tylko dla ujemnych
27      print('{:f}; {:f}'.format(3.14, -3.14))  # pokazuj tylko dla ujemnych
28      input()
29
30      # liczby całkowite w innych systemach
31      print("int: {0:d};  hex: {0:x};  oct: {0:o};  bin: {0:b}".format(42))
32      print("int: {0:d};  hex: {0:#x};  oct: {0:#o};  bin: {0:#b}".format(42)) #z prefixami
33      input()
34
35      #wartości procentowe
```

Console 1/A

```
+3.140000; -3.140000
 3.140000; -3.140000
3.140000; -3.140000
3.140000; -3.140000

int: 42;  hex: 2a;  oct: 52;  bin: 101010
int: 42;  hex: 0x2a;  oct: 0o52;  bin: 0b101010
```

Politechnika Wrocławska

Funkcje

Argumenty formalne i faktyczne

- Lista argumentów funkcji (umieszczona w nagłówku funkcji) nazywana jest listą argumentów formalnych (*formal arguments*)
- Wartości wyrażeń umieszczone w wywołaniu funkcji nazywane argumentami faktycznymi (*actual arguments*)

Politechnika Wrocławska

Funkcje

Argumenty nazwane

show_interest_01.py | show_interest_02.py | show_interest_03.py

```python
#Ten program pokazuje przykład wywołania funkcji
#z użyciem argumentów nazwanych.

#Funkcja główna wywołuje funkcję wyświetlającą obliczony wynik.
def main():
    #W tym wywołaniu argumenty przyjmują wartości na podstawie pozycji.
    show_interest(1000,0.03,3)

    #W tym wywołaniu argumenty przyjmują wartości zgodnie z nazwą.
    #Kolejność nie ma znaczenia.
    show_interest(rate = 0.03, periods = 3, deposit = 1000)

#Funkcja wyświetlająca wysokość odsetek należnych od zdeponowanej kwoty
#w zależności od czasu trwania depozytu oraz wysokości oprocentowania.
def show_interest(deposit, rate, periods):
    interest = deposit * rate * periods
    print('Wysokość odsetek wynosi',
          format(interest, '.2f')
          )

#Wywołanie funkcji głównej
main()
```

Politechnika Wrocławska

Funkcje

Argumenty nazwane

show_interest_01.py | show_interest_02.py | show_interest_03.py

```python
#Ten program pokazuje przykład wywołania funkcji
#z użyciem argumentów nazwanych oraz pozycyjnych.

#Funkcja główna wywołuje funkcję wyświetlającą obliczony wynik.
def main():
    #W tym wywołaniu argumenty przyjmują wartości na podstawie pozycji.
    show_interest(1000,0.03,3)

    #W tym wywołaniu argumenty przyjmują wartości zgodnie z nazwą.
    #Kolejność nie ma znaczenia.
    show_interest(deposit = 1000, rate = 0.03, periods = 3)
    show_interest(rate = 0.03, periods = 3, deposit = 1000)

    #W tym wywołaniu pierwszy argument przyjmuje wartość dzięki pozycji.
    #Kolejne na podstawie nazwy.
    show_interest(1000, periods = 3, rate = 0.03)

#Funkcja wyświetlająca wysokość odsetek
#należnych od zdeponowanej kwoty
#w zależności od czasu trwania depozytu
#oraz wysokości oprocentowania.
def show_interest(deposit, rate, periods):
    interest = deposit * rate * periods
    print('Wysokość odsetek wynosi',
          format(interest, '.2f')
          )

#Wywołanie funkcji głównej
main()
```

Politechnika Wrocławska

Funkcje

Argumenty nazwane

show_interest_01.py | show_interest_02.py | show_interest_03.py

```python
#Ten program pokazuje przykład wywołania funkcji
#z użyciem argumentów nazwanych oraz pozycyjnych.

#Funkcja główna wywołuje funkcję wyświetlającą obliczony wynik.
def main():
    #W tym wywołaniu argumenty przyjmują wartości na podstawie pozycji.
    show_interest(1000,0.03,3)

    #W tym wywołaniu argumenty przyjmują wartości zgodnie z nazwą.
    #Kolejność nie ma znaczenia.
    show_interest(deposit = 1000, rate = 0.03, periods = 3)
    show_interest(rate = 0.03, periods = 3, deposit = 1000)

    #W tym wywołaniu pierwszy argument przyjmuje wartość dzięki pozycji.
    #Kolejne na podstawie nazwy.
    show_interest(1000, periods = 3, rate = 0.03)

#Funkcja wyświetlająca wysokość odsetek
#należnych od zdeponowanej kwoty
#w zależności od czasu trwania depozytu
#oraz wysokości oprocentowania.
def show_interest(deposit, rate, periods):
    interest = deposit * rate * periods
    print('Wysokość odsetek wynosi',
          format(interest, '.2f')
          )

#Wywołanie funkcji głównej
main()
```

Politechnika Wrocławska

Funkcje

Argumenty o wartościach domyślnych

show_interest_01.py | show_interest_02.py | show_interest_03.py

```
1   #Ten program pokazuje przykład wywołania funkcji
2   #z użyciem argumentów domyślnych.
3
4   #Funkcja główna wywołuje funkcję wyświetlającą obliczony wynik.
5   def main():
6       #W tym wywołaniu argumenty przyjmują wartości domyślnych.
7       show_interest()
8
9   #Funkcja wyświetlająca wysokość odsetek
10  #należnych od zdeponowanej kwoty
11  #w zależności od czasu trwania depozytu
12  #oraz wysokości oprocentowania.
13  def show_interest(deposit = 1000, rate = 0.03, periods = 3):
14      interest = deposit * rate * periods
15      print('Wysokość odsetek wynosi',
16            format(interest, '.2f')
17            )
18
19  #Wywołanie funkcji głównej
20  main()
```

Politechnika Wrocławska

Funkcje

Zwracanie wartości None

return_none.py

```
1  """
2  Przykład pokazuje wykorzystanie typu NoneType,
3  gdy funkcja "nic nie zwraca" żadnej wartości.
4  """
5
6  def fun():
7      print("Komunikat z funkcji fun()")
8
9  def fun2():
10     print("Komunikat z funkcji fun2()")
11     return
12
13 print(fun())
14 print(fun2())
```

Console 1/A

```
Komunikat z funkcji fun()
None
Komunikat z funkcji fun2()
None
```

Politechnika Wrocławska

Funkcje

Zwracanie kilku wartości

return_none.py | return_tuple.py

```
1  #Przykładowy program wywołujący funkcję zwracającą kilka wartości.
2
3  #Funkcja główna programu
4  def main():
5      first_name, last_name = get_name()  #wczytanie imienia i nazwiska
6      #wyświetlenie powitania (operator + użyty do połączenia łańcuchów)
7      print('Witaj,', first_name + ' ' + last_name )
8
9  #Funkcja get_name()
10 #  wejście:       brak
11 #  przetwarzanie: funkcja prosi użytkownika o podanie imienia
12 #                 funkcja prosi użytkownika o podanie nazwiska
13 #  wyjście:       imię i nazwisko (jako dwa łańcuchy znakowe)
14 def get_name():
15     first = input('Podaj imię: ')
16     last  = input('Podaj nazwisko: ')
17     return first, last
18
19 main()
```

Console 1/A

```
Podaj imię: Ka
Podaj nazwisko: Tar
Witaj, Ka Tar
```

Politechnika Wrocławska

Algorytmy iteracyjne

- iteracja – powtarzanie pewnych instrukcji (ciągów instrukcji)
- przykładowo: wyznaczanie maksymalnego elementu w zbiorze wymaga przejrzenia wszystkich elementów zbioru

Politechnika Wrocławska

Algorytmy iteracyjne

Liniowe przeszukiwanie

- Czy we wskazanym zbiorze znajduje się poszukiwany element?

- Dane: ciąg liczb w liście *l* rozmiaru *n*, poszukiwana liczba *x*
- Wynik: indeks pozycji w liście *l*, na której znajduje się element *x*, lub None jeśli nie ma poszukiwanego elementu

Politechnika Wrocławska

Algorytmy iteracyjne

Liniowe przeszukiwanie

- Dane: ciąg liczb w liście *l* rozmiaru *n*, poszukiwana liczba *x*
- Wynik: indeks pozycji w liście *l*, na której znajduje się element *x*, lub None jeśli nie ma poszukiwanego elementu

- dla *i* = 0, 1, …, *n*-1
  - jeśli *l*[*i*] == *x* zwróć *i*
- zwróć None

Politechnika Wrocławska

Algorytmy iteracyjne

Obliczanie średniej

- Dane: ciąg liczb w liście *l* rozmiaru *n*
- Wynik: średnia wszystkich elementów

Politechnika Wrocławska

Algorytmy iteracyjne

Znajdowanie elementu maksymalnego

- Dane: ciąg liczb w liście *l* rozmiaru *n*
- Wynik: wartość elementu maksymalnego

- wybierz jako maksimum (kandydata na maksimum) dowolny element
- dla wszystkich pozostałych elementów:
  - jeśli przeglądany element jest większy niż aktualne maksimum zaktualizuj maksimum
- zwróć maksimum

Politechnika Wrocławska

Algorytmy iteracyjne

Znajdowanie elementu maksymalnego

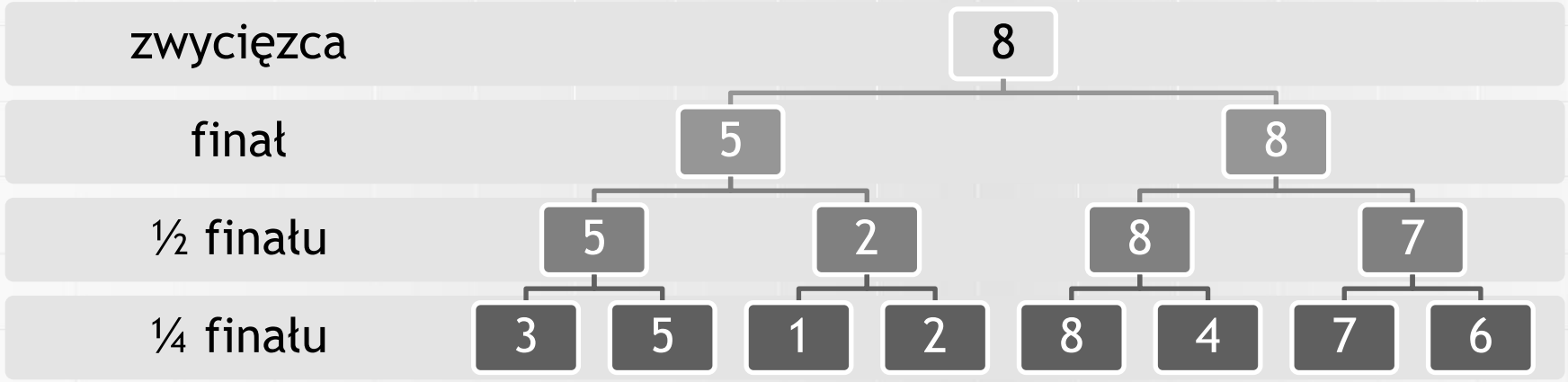
- Liczba porównań wynosi n-1
- Czy można mniej?

Politechnika Wrocławska

Algorytmy iteracyjne

Znajdowanie elementu maksymalnego

- Rozważmy metodę pucharową



Politechnika Wrocławska

Algorytmy iteracyjne

Kompletowanie podium

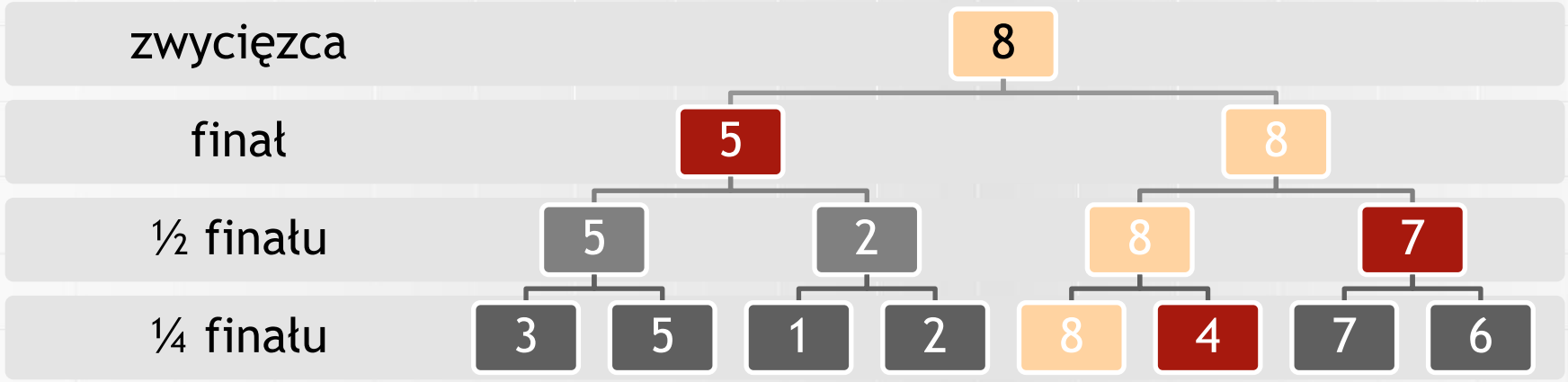
- Jak wybrać drugiego zawodnika turnieju?
- Druga drabinka?
  - n-1 porównań (wybór zwycięzcy)
  - n-2 porównania (wybór drugiego zawodnika)
  - łącznie $2n$-3
- Czy można mniej?

Politechnika Wrocławska

Algorytmy iteracyjne

Kompletowanie podium

- Modyfikacja metody pucharowej

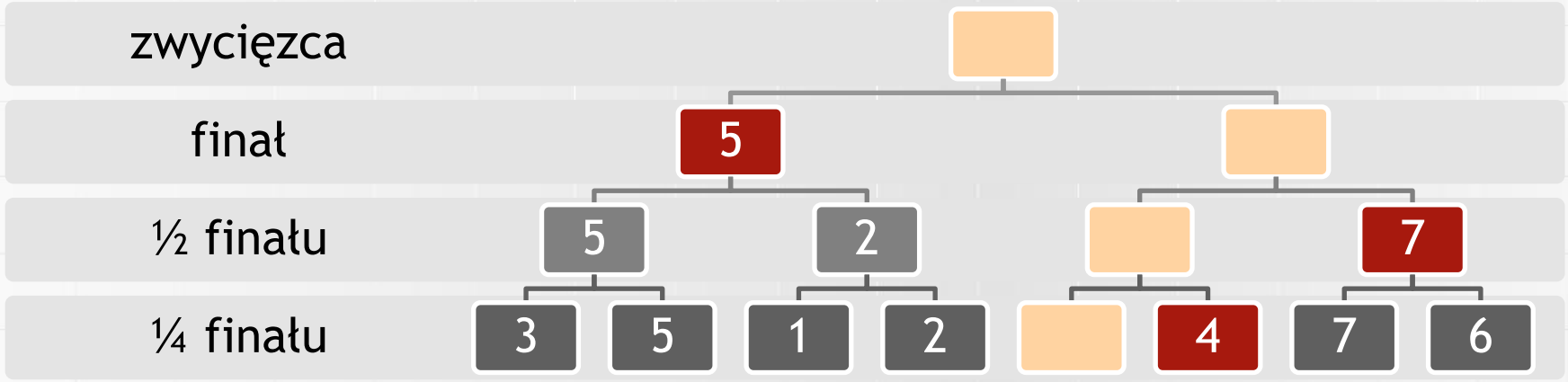


Politechnika Wrocławska

Algorytmy iteracyjne

Kompletowanie podium

- Modyfikacja metody pucharowej

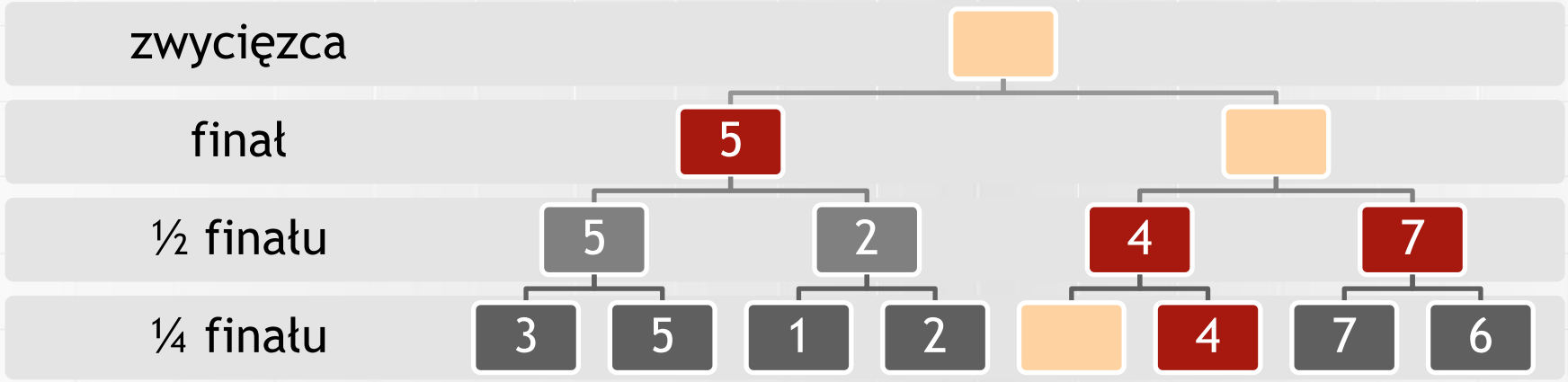


Politechnika Wrocławska

Algorytmy iteracyjne

Kompletowanie podium

- Modyfikacja metody pucharowej

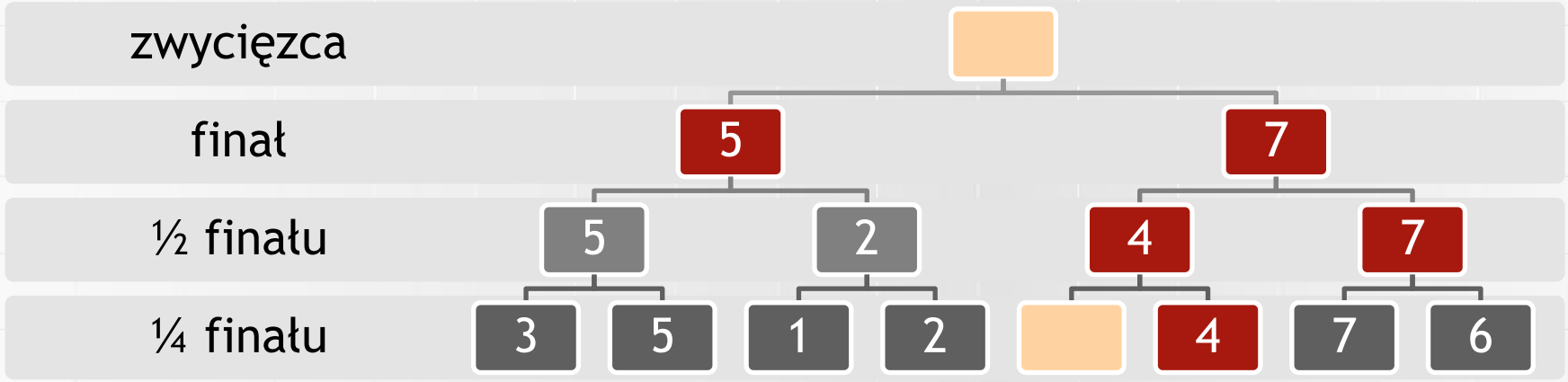


Politechnika Wrocławska

Algorytmy iteracyjne

Kompletowanie podium

- Modyfikacja metody pucharowej

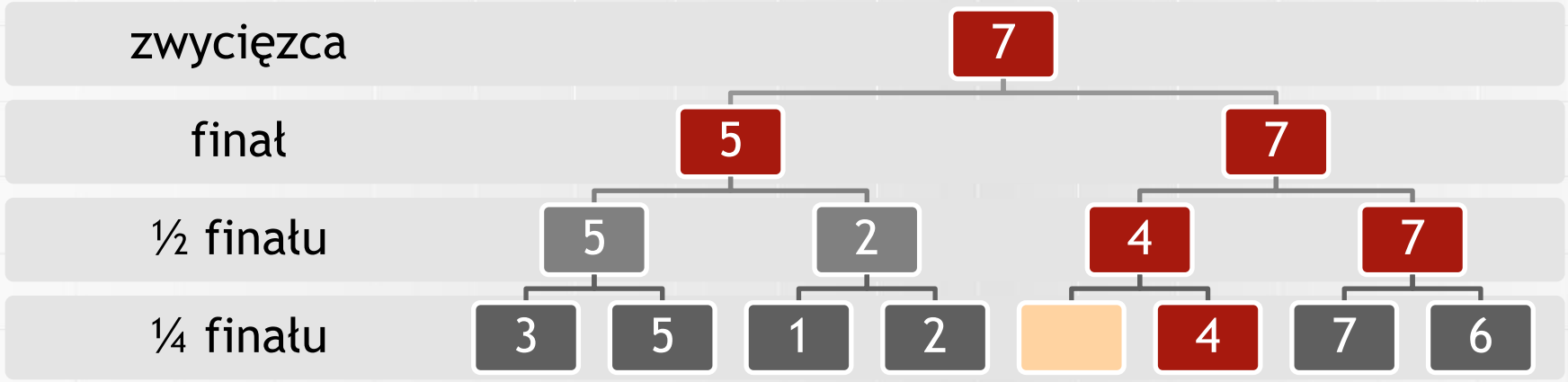


Politechnika Wrocławska

Algorytmy iteracyjne

Kompletowanie podium

- Modyfikacja metody pucharowej



Politechnika Wrocławska

Algorytmy iteracyjne

Kompletowanie podium

- Liczba porównań:
  - n-1 (wybór zwycięzcy),
  - $\log_2 n$-1 (wybór drugiego zawodnika)
  - łącznie $n + \log_2 n - 2$

Politechnika Wrocławska

Algorytmy iteracyjne

Jednoczesne znajdowanie maksimum i minimum

min	3							
	3	5	1	2	8	4	7	6
max		5						

Politechnika Wrocławska

Algorytmy iteracyjne

Jednoczesne znajdowanie maksimum i minimum

min	3		1					
	3	5	1	2	8	4	7	6
max		5		2				

Politechnika Wrocławska

Algorytmy iteracyjne

Jednoczesne znajdowanie maksimum i minimum

min	3		1			4		
	3	5	1	2	8	4	7	6
max		5		2	8			

Politechnika Wrocławska

Algorytmy iteracyjne

Jednoczesne znajdowanie maksimum i minimum

min	3		1			4		6
	3	5	1	2	8	4	7	6
max		5		2	8		7	

Politechnika Wrocławska

Algorytmy iteracyjne

Jednoczesne znajdowanie maksimum i minimum

min	3		1			4	7
	3	5	1	2	8	4	7
max		5		2	8		7

Politechnika Wrocławska

Algorytmy iteracyjne

Jednoczesne znajdowanie maksimum i minimum

min	3		1			4		6
	3	5	1	2	8	4	7	6
max		5		2	8		7	

Politechnika Wrocławska

Algorytmy iteracyjne

Jednoczesne znajdowanie maksimum i minimum

- Liczba porównań dla parzystych *n*:
  - n/2 porównań par elementów
  - n/2 – 1 porównań (wybór maksimum)
  - n/2 – 1 porównań (wybór minimum)
  - łącznie 3n/2 - 2

Politechnika Wrocławska

Wprowadzenie do pętli

- Typowym zadaniem wykonywanym przez programy komputerowe jest wielokrotne powtarzanie tej samej sekwencji poleceń (np. wykonywanie takich samych obliczeń dla zestawu danych)

Politechnika Wrocławska

Wprowadzenie do pętli

01_powtorzenia.py

```
1   #Program pokazuje wielokrotne powtórzenie tej samej sekwencji poleceń.
2   #
3   #Program pobiera od użytkownika przyspieszenie
4   #w ruchu jednostajnie przyspieszonym i oblicza drogę przebytą przez ciało
5   #we wskazanym czasie (zakłada się ruch bez prędkości początkowej).
6   #Program wykorzystuje funkcje get_acceleration(), get_time(),
7   #calculate_displacement().
8   def main():
9       a = get_acceleration()
10
11      #początek powtórzonej sekwencji
12      t = get_time()
13      s = calculate_displacement(a,t)
14      print('W czasie',t,'s','ciało przebyło drogę',s,'m.')
15      #koniec powtórzonej sekwencji
```

Politechnika Wrocławska

Wprowadzenie do pętli

01_powtorzenia.py

```
 8  def main():
 9      a = get_acceleration()
10
11      #początek powtórzonej sekwencji
12      t = get_time()
13      s = calculate_displacement(a,t)
14      print('W czasie',t,'s','ciało przebyło drogę',s,'m.')
15      #koniec powtórzonej sekwencji
16
17      #drugie powtórzenie
18      t = get_time()
19      s = calculate_displacement(a,t)
20      print('W czasie',t,'s','ciało przebyło drogę',s,'m.')
21
22      #trzecie powtórzenie
23      t = get_time()
24      s = calculate_displacement(a,t)
25      print('W czasie',t,'s','ciało przebyło drogę',s,'m.')
```

Politechnika Wrocławska

Wprowadzenie do pętli

01_powtorzenia.py

```
27  #I: brak
28  #P: pobranie od użytkownika przyspieszenia i konwersja na float
29  #O: wartość przyspieszenia
30  def get_acceleration():
31      return float(input('Podaj przyspieszenie [m/s^2]: '))
32
33  #I: brak
34  #P: pobranie od użytkownika czasu i konwersja na float
35  #O: czas
36  def get_time():
37      return float(input('Podaj czas [s]: '))
38
39  #I: przyspieszenie i czas
40  #P: obliczenie drogi według wzoru 1/2*a*t**2
41  #O: droga
42  def calculate_displacement(a, t):
43      return .5*a*t**2
44
45  main()
```

Politechnika Wrocławska

Pętla `while`

- Do wielokrotnego wykonania fragmentu kodu można wykorzystać pętle **`while`**

- Składnia:

```
while warunek:
    ciało pętli
```

Politechnika Wrocławska

Pętla while

02_petla_while.py

```
1  #Program pokazuje wielokrotne wykonanie tej samej sekwencji poleceń
2  #z wykorzystaniem pętli while.
3  #
4  #Program pobiera od użytkownika przyspieszenie
5  #w ruchu jednostajnie przyspieszonym i oblicza drogę przebytą przez ciało
6  #we wskazanym czasie (zakłada się ruch bez prędkości początkowej).
7  #Program wykorzystuje funkcje get_acceleration(), get_time(),
8  #calculate_displacement().
9
10 def main():
11     a = get_acceleration()
12
13     keep_going = 't'
14
15     while keep_going == 't':
16         t = get_time()
17         s = calculate_displacement(a,t)
18         print('W czasie',t,'s','ciało przebyło drogę',s,'m.')
19         keep_going = input('Czy chcesz obliczyć kolejną drogę?' +
20                            '(jeśli tak, wpisz t): ')
21
```

Politechnika Wrocławska

Pętla `while`

Pętla nieskończona

03_nieskonczona_petla_while.py

```
1  #Program pokazuje pętlę nieskończoną.
2  #
3  #Program pobiera od użytkownika przyspieszenie
4  #w ruchu jednostajnie przyspieszonym i oblicza drogę przebytą przez ciało
5  #we wskazanym czasie (zakłada się ruch bez prędkości początkowej).
6  #Program wykorzystuje funkcje get_acceleration(), get_time(),
7  #calculate_displacement().
8
9  def main():
10     a = get_acceleration()
11
12     keep_going = 't' #nadano keep_going wartość 't'
13
14     while keep_going == 't': #warunek jest prawdziwy
15         t = get_time()
16         s = calculate_displacement(a,t)
17         print('W czasie',t,'s','ciało przebyło drogę',s,'m.')
18
19         #wartość keep_going nie zmienia się nigdzie w pętli
20         #czyli warunek keep_going == 't' będzie zawsze spełniony
21         #(wykonanie pętli można przerwać skrótem klawiaturowym Ctrl + C)
```

Politechnika Wrocławska

Pętla `while`

- Pętla **`while`**, która w ciele pętli nie zmienia wartości zmiennych wykorzystanych w warunku pętli, będzie pętlą nieskończoną

Politechnika Wrocławska

Pętla `for`

- Do wielokrotnego wykonania fragmentu kodu można wykorzystać również pętle **`for`**

- Składnia:

```
for iterator in lista_elementów:
    ciało pętli
```

Politechnika Wrocławska

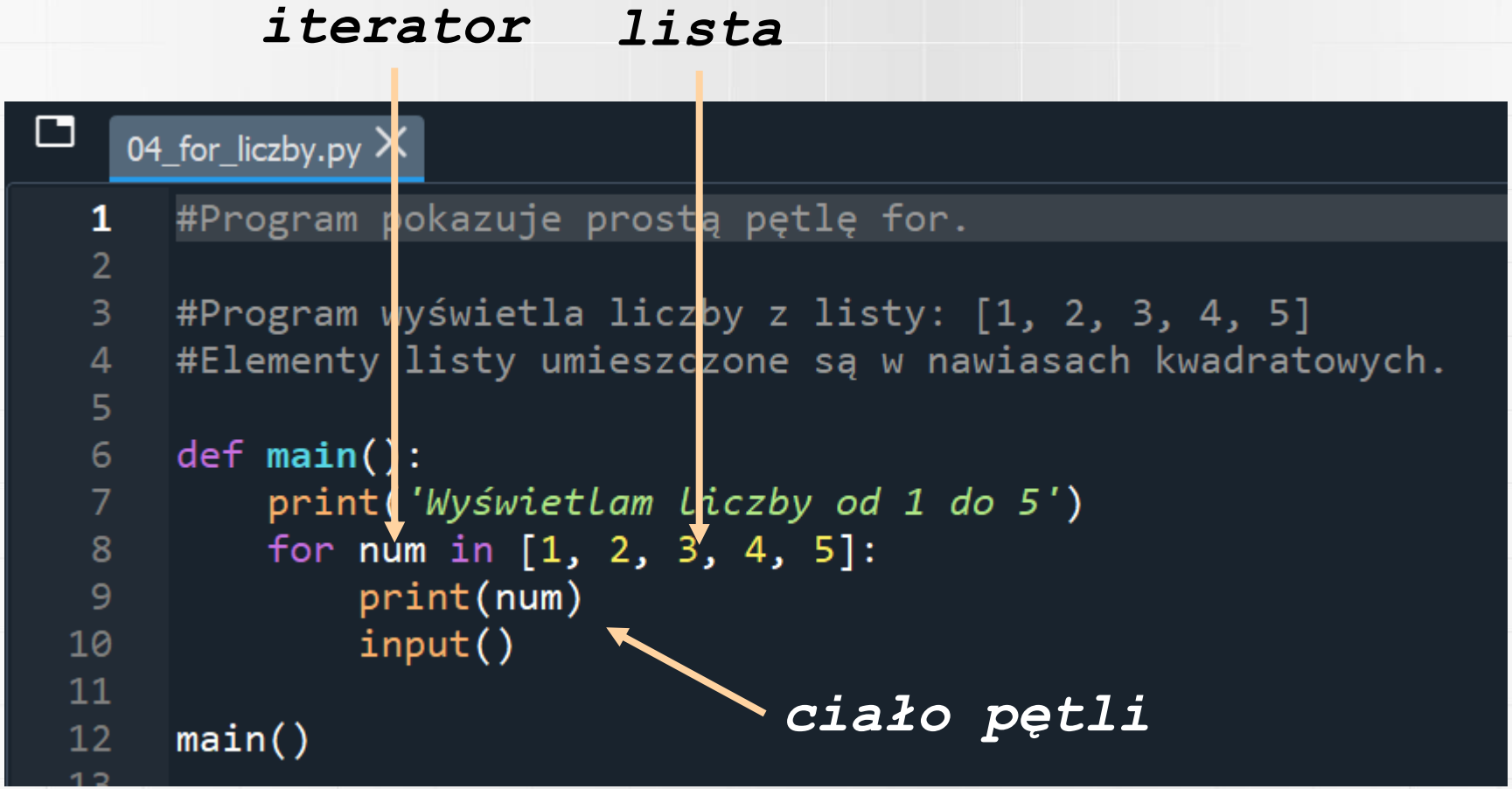
Pętla for

04_for_liczby.py

```
1  #Program pokazuje prostą pętlę for.
2
3  #Program wyświetla liczby z listy: [1, 2, 3, 4, 5]
4  #Elementy listy umieszczone są w nawiasach kwadratowych.
5
6  def main():
7      print('Wyświetlam liczby od 1 do 5')
8      for num in [1, 2, 3, 4, 5]:
9          print(num)
10         input()
11
12 main()
```

Politechnika Wrocławska

Pętla for



Politechnika Wrocławska

Pętla for

04_for_liczby.py | 05_for_liczby.py

```
1  #Program pokazuje prostą pętlę for.
2
3  #Program wyświetla liczby z listy: [1, 3, 5, 7, 9]
4
5  def main():
6      print('Wyświetlam liczby nieparzyste od 1 do 9')
7      for num in [1, 3, 5, 7, 9]:
8          print(num)
9
10 main()
11
```

Politechnika Wrocławska

Pętla `for`

- Lista może zawierać również ciągi tekstowe

04_for_liczby.py | 05_for_liczby.py | 06_for_ciagi_tekstowe.py

```
1  #Program pokazuje prostą pętlę for.
2
3  #Program wyświetla ciągi tekstowe z listy.
4  #Lista może zawierać ciągi tekstowe jako elementy.
5
6  def main():
7      print('Atomówki to:')
8      for name in ['Bajka', 'Bójka', 'Brawurka']:
9          print(name)
10
11 main()
```

Politechnika Wrocławska

Pętla `for`

- Do generowania listy można wykorzystać klasę **`range()`**

07_for_liczby_range.py

```python
#Program pokazuje użycie klasy range() do generowania listy.

#Program wyświetla liczby: [0, 1, 2, 3, 4]
#Wartości są generowane przez obiekt klasy range().

def main():
    print('Wyświetlam liczby od 0 do 4')
    for num in range(5):
        print(num)

main()
```

https://docs.python.org/3/library/stdtypes.html#range

Politechnika Wrocławska

Pętla for

08_for_range.py

```
1  #Program pokazuje prostą pętlę for.
2
3  #Program wyświetla ten sam ciąg tekstowy 5 razy.
4
5  def main():
6      for x in range(5):
7          print('Python jest fajny!')
8
9  main()
```

Politechnika Wrocławska

Pętla for

10_for_range.py

```
1  #Program pokazuje użycie pętli for i klasy range()
2  #z danymi pochodzącymi od użytkownika.
3
4  #Program wyświetla liczby całkowite od podanej liczby początkowej
5  #mniejsze od zadanego ograniczenia z podanym krokiem.
6
7  def main():
8      print('Program wyświetla liczby całkowite od podanej liczby początkowej')
9      print('mniejsze od zadanego ograniczenia z podanym krokiem.')
10     init = int(input('Podaj liczbę początkową: '))
11     end  = int(input('Podaj ograniczenie: '))
12     step = int(input('Podaj krok: '))
13     for i in range(init, end, step):
14         print(i)
15
16 main()
```

Politechnika Wrocławska

Pętla `for`

- Konstruktor klasy **`range()`** może przyjmować trzy argumenty:
  - start – początek zakresu,
  - stop – koniec zakresu,
  - step – krok.

Politechnika Wrocławska

Pętla for

11_for_sumowanie_liczb.py

```
1  #Program pokazuje przykład użycia pętli for.
2  #
3  #Program sumuje 5 liczb podanych przez użytkownika.
4
5  MAX = 5
6
7  def main():
8      print('Program sumuje',MAX,'liczb podanych przez użytkownika.')
9
10     total = 0.0
11     for counter in range(MAX):
12         number = float(input('Podaj liczbę: '))
13         total  = total + number
14
15     print('Suma wynosi ',total)
16
17 main()
```

Politechnika Wrocławska

Operatory złożone

Operator	Przykład użycia	Odpowiednik polecenia
`+=`	`x += 5`	`x = x + 5`
`-=`	`y -= 2`	`y = y - 2`
`*=`	`z *= 10`	`z = z * 10`
`/=`	`a /= b`	`a = a / b`
`%=`	`c %= 3`	`c = c % 3`

Politechnika Wrocławska

Operatory złożone

12_zlozone_operatory.py

```
1   #Program pokazuje przykład użycia
2   #złożonego operatora dodawania.
3   #
4   #Program sumuje 5 liczb podanych przez użytkownika.
5
6   MAX = 5
7
8   def main():
9       print('Program sumuje',MAX,'liczb podanych przez użytkownika.')
10
11      total = 0.0
12      for counter in range(MAX):
13          number = float(input('Podaj liczbę: '))
14          #złożony operator dodawania polecenie
15          #total = total + number
16          total += number
17
18      print('Suma wynosi ',total)
19
20  main()
```

Politechnika Wrocławska

Pętle for i while

Porównanie

20_while_for_break_continue.py

```python
# Program pokazuje porównanie pętli while oraz for
# na przykładzie wypisywania ciagu tekstowego.

# Dodatkowo pokazuje także działanie instrukcji
# break oraz continue w obu pętlach.

def main():
    text = 'napis'

    print('while_example(text)')
    while_example(text)
    input()

    print('for_example_1(text)')
    for_example_1(text)
    input()

    print('for_example_2(text)')
    for_example_2(text)
    input()
```

Politechnika Wrocławska

Pętle for i while

Porównanie

20_while_for_break_continue.py

```
1  # Program pokazuje porównanie pętli while oraz for
2  # na przykładzie wypisywania ciagu tekstowe
3
4  # Dodatkowo pokazuje także działanie instru
5  # break oraz continue w obu pętlach.
6
7  def main():
8      text = 'napis'
9
10     print('while_example(text)')
11     while_example(text)
12     input()
13
14     print('for_example_1(text)')
15     for_example_1(text)
16     input()
17
18     print('for_example_2(text)')
19     for_example_2(text)
20     input()
```

20_while_for_break_continue.py

```
21
22     print('while_break_example(text)')
23     while_break_example(text)
24     input()
25
26     print('for_break_example(text)')
27     for_break_example(text)
28     input()
29
30     print('while_continue_example(text)')
31     while_continue_example(text)
32     input()
33
34     print('for_continue_example(text)')
35     for_continue_example(text)
36
37
```

Politechnika Wrocławska

Pętle `for` i `while`

Porównanie

20_while_for_break_continue.py

```
39  def while_example(text):
40      i = 0
41      while i < len(text):
42          print(i, text[i])
43          i = i+1
44
45  def for_example_1(text):
46      for c in text:
47          print(c)
48
49  def for_example_2(text):
50      for i in range(len(text)):
51          print(i, text[i])
52          #i = i+1
53
```

Politechnika Wrocławska

Pętle `for` i `while`

Porównanie

- W pętli **`while`** instrukcje sterujące przebiegiem pętli nie są w jednym miejscu
- W pętli **`while`** trzeba samodzielnie zmieniać licznik pętli
- W pętli **`while`** można zmieniać licznik pętli
- Pętla **`for`** sprawdza się przy znanej liczbie powtórzeń, pętla **`while`** sprawdza się przy nieznanej liczbie powtórzeń

Politechnika Wrocławska

Instrukcje `break` i `continue`

- Polecenie **`break`** kończy wykonanie pętli

20_while_for_break_continue.py

```
54  def while_break_example(text):
55      i = 0
56      while i < len(text):
57          if text[i] == 'p':
58              break
59          print(text[i])
60          i = i+1
61
62  def for_break_example(text):
63      for c in text:
64          if c == 'p':
65              break
66          print(c)
67
```

Politechnika Wrocławska

Instrukcje `break` i `continue`

- Polecenie **`continue`** kończy wykonanie iteracji pętli

20_while_for_break_continue.py

```
68  def while_continue_example(text):
69      i = 0
70      while i < len(text):
71          if text[i] == 'p':
72              i = i+1
73              continue
74          print(text[i])
75          i = i+1
76
77  def for_continue_example(text):
78      for c in text:
79          if c == 'p':
80              continue
81          print(c)
82
```

Politechnika Wrocławska

Podsumowanie

- Funkcja print() – kontynuacja
- Formatowanie liczb
- Argumenty: pozycyjne, nazwane, domyślne
- Funkcje: zwracanie wartości

- Algorytmy iteracyjne

- Pętle