Politechnika Wrocławska

Wstęp do programowania

INP001126W

rok akademicki 2021/22

semestr zimowy

Wykład 4

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji (1)

- Porównania ciągów znakowych
- Znaki jako liczby
- Dane logiczne

Na podstawie: T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Politechnika Wrocławska

Plan prezentacji (2)

- Wprowadzenie do pętli
- Pętla **`while`**
- Pętla **`for`**
- Złożone operatory przypisania
- Polecenia **`break`** oraz **`continue`**

Na podstawie: T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Porównywanie ciągów znakowych

example_if_07.py

File Edit Format Run Options Window Help

```
#Program pokazuje porównanie ciągów teksowych.

def main():
    name1 = 'Maria'
    name2 = 'Marek'

    # Operator porównania sprawdza, czy dwa ciągi znaków
    # są sobie równe - czy są takie same.
    # W zależności od wyniku porównania instrukcja warunkowa
    # wybierze jedną z dwóch ścieżek.
    if name1 == name2:
        print('Imiona są takie same')
    else:
        print('Imiona nie są takie same')

main()
```

Porównywanie ciągów znakowych

example_if_08.py

File Edit Format Run Options Window Help

```
#Program pokazuje porównanie ciągów teksowych.

def main():
    # Użytkownik podaje hasło
    password = input('Podaj hasło: ')

    # Instrukcja warunkowa sprawdza, czy hasło jest poprawne.
    if password == '3.1415':
        print('Hasło jest poprawne.')
    else:
        print('Podane hasło nie jest poprawne.')

main()
```

Politechnika Wrocławska

Porównywanie ciągów znakowych

- Operator porównania działa również dla ciągów tekstowych

Politechnika Wrocławska

Znaki jako liczby

Symbole znakowe są przechowywane w pamięci komputera jako liczby.

Najbardziej podstawowym system kodowania jest system ASCII (American Standard Code for Information Interchange)

- literom od „A” do „Z” odpowiadają liczby od 65 do 90,
- literom od „a” do „z” odpowiadają liczby od 97 do 122,
- cyfrom od „0” do „9” odpowiadają liczby od 48 do 57,
- znakowi spacji odpowiada liczba 32,
- znakowi nowej linii odpowiada liczba 10.

Kody ASCII nie obejmują polskich znaków diakrytycznych.

Politechnika Wrocławska

Znaki jako liczby

- Rozszerzeniem zestawu znaków ASCII na znaki używane w różnych językach jest zestaw znaków Unicode
- Do zapisywania znaków Unicode wykorzystuje się różne kodowania
- Popularnym kodowaniem jest kodowanie UTF-8 (Unicode Transformation Format), gdzie 8 oznacza, że do kodowania wykorzystywane są liczby 8-bitowe

https://docs.python.org/3/howto/unicode.html

Politechnika Wrocławska

Znaki jako liczby

- Funkcja **`ord()`** dla podanego znaku zwraca jego kod liczbowy
- Funkcja **`chr()`** dla podanego kodu liczbowego zwraca odpowiadający mu znak

Porównywanie ciągów znakowych

example_if_09.py

File Edit Format Run Options Window Help

```
#Program pokazuje porównanie ciągów teksowych.

def main():
    name1 = 'Maria'
    name2 = 'Marek'

    # Operator większości sprawdza, czy ciąg pierwszy jest
    # większy niż drugi (w porządku alfabetycznym).
    if name1 >  name2:
        print('Imię: ', name1, '' , sep = '"', end = '')
        print(' jest większe niż imię: ', name2, '.', sep = '"')

    #    M    a    r    i   a
    #   77   97  114  105  97

    #    M    a    r    e   k
    #   77   97  114  101 107

    # M==M a==a r==r  i>e

main()
```

Politechnika Wrocławska

Porównywanie ciągów znakowych

- Porównywanie ciągów tekstowych sprowadza się do porównywania kodów kolejnych znaków ciągu

Politechnika Wrocławska

Logiczny typ danych

- Dana typu logicznego może przyjmować jedną z dwóch wartości: prawda (**`True`**) lub fałsz (**`False`**).

```
hungry = True
sleepy = False
```

Logiczny typ danych

Przykład

example_bool_01.py

File Edit Format Run Options Window Help

```
#Program pokazuje sprawdzanie parzystość liczby.
#Parzystość liczby jest związana z podzielnością przez 2.

def main():
    x = 14

    # Jeśli liczba dzieli się przez 2 bez reszty
    # (reszta z dzielenia przez 2 równa się 0),
    if x % 2 == 0:
        #to liczba jest parzysta
        print('Liczba', x, 'jest parzysta')
    else:
        #w p. p. jest nieparzysta
        print('Liczba', x, 'jest nieparzysta')

main()
```

Logiczny typ danych

Przykład

example_bool_02.py

File Edit Format Run Options Window Help

```
#Program pokazuje sprawdzanie parzystość liczby.
#Wynik sprawdzenia (wartość logiczna)
#zapisywany jest w zmiennej even.

def main():
    x = 14

    # Zmienna even zapamiętuje, czy liczba jest parzysta.
    even = x % 2 == 0

    if even:
        #jest parzysta
        print('Liczba', x, 'jest parzysta')
    else:
        #w p. p. jest nieparzysta
        print('Liczba', x, 'jest nieparzysta')

    # Drugi raz wykorzystujemy obliczoną zmienną.
    if even:
        print('Ciągle parzysta')

main()
```

Politechnika Wrocławska

Logiczny typ danych

- Wynik operacji logicznej można przechowywać w zmiennej typu logicznego

Logiczny typ danych

Przykład

example_bool_03.py

File Edit Format Run Options Window Help

```
#Program pokazuje sprawdzanie parzystość liczby.
#Do sprawdzania parzystości wykorzystywana jest
#funkcja is_even() zwracająca wartość logiczną.

def main():
    x = 14

    #wywołanie funkcji is_even() w warunku instrukcji if
    if is_even(x):
        print('Liczba', x, 'jest parzysta')
    else:
        print('Liczba', x, 'jest nieparzysta')

#I: liczba całkowita
#P: sprawdzane jest, czy reszta z dzielenia przez 2
#   równa się 0
#O: wartość logiczna:
#   True (jeśli liczba jest parzysta)
#   False (w p. p.)
def is_even(x):
    return x % 2 == 0

main()
```

Politechnika Wrocławska

Logiczny typ danych

- Dane typu logicznego mogą być również zwracane przez funkcje

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)

Wprowadzenie do pętli

01_powtorzenia.py

File Edit Format Run Options Window Help

```
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
16
```

Politechnika Wrocławska

Wprowadzenie do pętli

01_powtorzenia.py

File Edit Format Run Options Window Help

```
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.')
26
```

Politechnika Wrocławska

Wprowadzenie do pętli

01_powtorzenia.py

File Edit Format Run Options Window Help

```
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
```

Pętla while

02_petla_while.py

File Edit Format Run Options Window Help

```
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
```

Pętla `while`

Pętla nieskończona

03_nieskonczona_petla_while.py

File Edit Format Run Options Window Help

```
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

File Edit Format Run Options Window Help

```
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()
```

Pętla for

iterator *lista*

04_for_liczby.py

File Edit Format Run Options Window Help

```
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()
```

ciało pętli

Politechnika Wrocławska

Pętla for

05_for_liczby.py

File Edit Format Run Options Window Help

```
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()
```

Politechnika Wrocławska

Pętla `for`

- Pętla **`for`** może przechodzić po liście o dowolnych elementach

Politechnika Wrocławska

Pętla for

- Lista może zawierać również ciągi tekstowe

06_for_ciagi_tekstowe.py

File Edit Format Run Options Window Help

```
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 funkcję **`range()`**

07_for_liczby_range.py

File Edit Format Run Options Window Help

```
1  #Program pokazuje użycie funkcji range() do generowania listy.
2
3  #Program wyświetla liczby z listy: [0, 1, 2, 3, 4]
4  #Listę można stworzyć poleceniem funkcją range().
5
6  def main():
7      print('Wyświetlam liczby od 0 do 4')
8      for num in range(5):
9          print(num)
10
11 main()
```

Politechnika Wrocławska

Pętla for

08_for_range.py

File Edit Format Run Options Window Help

```
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()
10
```

Politechnika Wrocławska

Pętla for

08_for_range.py

File Edit Format Run Options Window Help

```
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()
10
```

Pętla for

09_for_range.py

File Edit Format Run Options Window Help

```
1  #Program pokazuje użycie pętli for i funkcji range()
2  #do wygenerowania tabeli danych.
3
4  #Program wyświetla tabelę prędkości w metrach na sekundę
5  #(od 10 do 50 co 10) z przeliczeniem na kilometry na godzinę.
6
7  RATIO = 3.6 # 1 m/s = (1/1000 km) / (1/3600 h) = 3.6 km/h
8
9  def main():
10     print('v[m/s]\tv[km/h]')
11     for v in range(10,60,10):
12         print(v,mps_to_kmph(v),sep='\t')
13
14 # I: prędkość w metrach na sekundę
15 # P: mnożenie prędkości przez przelicznik (1 m/s = 3.6 km/h)
16 # O: prędkość w kilometrach na godzinę
17 def mps_to_kmph(v): #meter per second to kilometers per hour
18     return v*RATIO
19
20 main()
```

Politechnika Wrocławska

Pętla for

10_for_range.py

File Edit Format Run Options Window Help

```
1  #Program pokazuje użycie pętli for i funkcji 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`

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

Politechnika Wrocławska

Pętla for

11_for_sumowanie_liczb.py

File Edit Format Run Options Window Help

```
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()
18
```

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`

Operatory złożone

12_zlozone_operatory.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykład użycia
#złożonego operatora dodawania.
#
#Program sumuje 5 liczb podanych przez użytkownika.

MAX = 5

def main():
    print('Program sumuje',MAX,'liczb podanych przez użytkownika.')

    total = 0.0
    for counter in range(MAX):
        number = float(input('Podaj liczbę: '))
        #złożony operator dodawania polecenie
        #total = total + number
        total += number

    print('Suma wynosi ',total)

main()
```

Politechnika Wrocławska

Pętle for i while

Porównanie

20_while_for_break_continue.py

File Edit Format Run Options Window Help

```
1  # Program pokazuje porównanie pętli while oraz for
2  # na przykładzie wypisywania ciagu tekstowego.
3
4  # Dodatkowo pokazuje także działanie instrukcji
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()
```

Politechnika Wrocławska

Pętle `for` i `while`

Porównanie

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

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

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

Politechnika Wrocławska

Instrukcje `break` i `continue`

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

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

Politechnika Wrocławska

Absolutne minimum (1)

- W jaki sposób porównywane są łańcuchy znakowe?
- Znaki są pamiętane (kodowane) jako liczby
- Jakie wartości może przyjmować logiczny typ danych?

Politechnika Wrocławska

Absolutne minimum (2)

- Kiedy stosuje się pętle?
- Jaka jest składnia pętli **`while`**?
- Jaka jest składnia pętli **`for`**?
- Jak wykorzystać funkcję **`range()`**?
- Jak wykorzystać operatory złożone?