Politechnika Wrocławska

Programowanie proceduralne

INP001210WL

rok akademicki 2020/21

semestr letni

Wykład 3

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji (1)

- Wprowadzenie do list
- Operator powtórzenia
- Listy i pętla `for`
- Indeksowanie elementów
- Długość listy
- Modyfikowanie list
- Konkatenacja list
- Wycinek listy

T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Politechnika Wrocławska

Plan prezentacji (2)

- Wyszukiwanie elementu listy
- Wybrane metody listy
- Kopiowanie listy
- Przetwarzanie danych
- Listy i pliki
- Listy zagnieżdżone

- Krotki

T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Politechnika Wrocławska

Wprowadzenie do list

- Lista to obiekt zawierający wiele elementów danych.
- Lista jest modyfikowalna – jej zawartość może być zmieniona w trakcie działania programu.

Politechnika Wrocławska

Wprowadzenie do list

- Przykład polecenia tworzącego listę:
  `parzyste = [2, 4, 6, 8, 10]`
- Listę można wyświetlić funkcją **`print()`**
  `print(parzyste)`

```
>>> parzyste = [2, 4, 6, 8, 10]
>>> print(parzyste)
[2, 4, 6, 8, 10]
>>>
```

Politechnika Wrocławska

Wprowadzenie do list

- Lista może zawierać ciągi tekstowe:

`kursy = ['Algebra', 'Analiza', 'Fizyka']`

```
>>> kursy = ['Algebra','Analiza','Fizyka']
>>> print(kursy)
['Algebra', 'Analiza', 'Fizyka']
>>>
```

Politechnika Wrocławska

Wprowadzenie do list

- Lista może zawierać elementy o różnych typach danych:

```
zawodnik = ['Bartosz Zmarzlik',78,2.410]
```

```
>>> zawodnik = ['Bartosz Zmarzlik',78,2.410]
>>> print(zawodnik)
['Bartosz Zmarzlik', 78, 2.41]
>>>
```

Politechnika Wrocławska

Wprowadzenie do list

- Istnieje wbudowana funkcja **`list()`** przeznaczona do konwertowania różnych typów obiektów na postać listy

`liczby = list(range(5))`

```
>>> print(range(5))
range(0, 5)
>>>
>>> liczby = list(range(5))
>>> print(liczby)
[0, 1, 2, 3, 4]
>>>
```

Politechnika Wrocławska

Operator powtórzenia

- Operator ***** zastosowany do listy i liczby całkowitej nazywany jest operatorem powtórzenia
- Operator powtórzenia powoduje utworzenie listy zawierającej wiele powtórzonych kopii

lista * n

```
>>> liczby = [0] * 5
>>> print(liczby)
[0, 0, 0, 0, 0]
>>>
```

Politechnika Wrocławska

Listy i pętla `for`

- Wykorzystując pętle **`for`** można prowadzić iterację przez listę

```
>>> liczby = [99, 100, 101, 102]
>>> for n in liczby:
        print(n)

99
100
101
102
>>>
```

Politechnika Wrocławska

Listy i pętla `for`

- Taka lista może zawierać elementy różnych typów

```
>>> zawodnik = ['Bartosz Zmarzlik',78,2.410]
>>> for element in zawodnik:
        print(element)

Bartosz Zmarzlik
78
2.41
>>>
```

Politechnika Wrocławska

Indeksowanie

- Dostęp do poszczególnych elementów listy można uzyskać używając indeksów

```
>>> lista = [10, 20, 30, 40]
>>> lista[0]
10
>>> lista[2]
30
>>> print(lista[1],lista[3])
20 40
```

Politechnika Wrocławska

Indeksowanie

- Użycie nieprawidłowego indeksu elementu powoduje zgłoszenie wyjątku

```
>>> print(lista)
[10, 20, 30, 40]
>>> lista[4]
Traceback (most recent call last):
  File "<pyshell#117>", line 1, in <module>
    lista[4]
IndexError: list index out of range
```

Politechnika Wrocławska

Indeksowanie

- Indeksy ujemne pozwalają na wskazywanie pozycji na liście od końca

```
>>> lista[-1]
40
>>> lista[-4]
10
>>> lista[-5]
Traceback (most recent call last):
  File "<pyshell#120>", line 1, in <module>
    lista[-5]
IndexError: list index out of range
>>>
```

Politechnika Wrocławska

Długość listy

- Do sprawdzenia długości listy można wykorzystać funkcję **`len()`**

```
>>> lista = [10, 20, 30, 40]
>>> len(lista)
4
>>>
```

Politechnika Wrocławska

Długość listy

- Wykorzystując funkcję **`len()`** można przejść przez listę pętlą **`while`**

```
>>> lista = [10, 20, 30, 40]
>>> index = 0
>>> while index < len(lista):
        print(lista[index])
        index += 1

10
20
30
40
>>>
```

Politechnika Wrocławska

Lista jest modyfikowalna

- Wartość elementów listy może być zmieniana

lista[index] = nowa_wartosc

```
>>> liczby = [1, 2, 3, 4, 5]
>>> print(liczby)
[1, 2, 3, 4, 5]
>>> liczby[2] = 77
>>> print(liczby)
[1, 2, 77, 4, 5]
>>>
```

Politechnika Wrocławska

Lista jest modyfikowalna

- Takie przypisanie działa tylko dla pozycji, które już są na liście

```
>>> liczby = [1, 2, 3, 4, 5]
>>> liczby[5] = 77
Traceback (most recent call last):
  File "<pyshell#60>", line 1, in <module>
    liczby[5] = 77
IndexError: list assignment index out of range
>>>
```

Politechnika Wrocławska

Lista jest modyfikowalna

01_tworzenie_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje uzupełnienie danymi listy
# o ustalonej długości.

# Program prosi użytkownika o podanie
# temperatur powietrza odnotowanych przez
# 7 kolejnych dni.

#Stała NUM_DAYS określa liczbę dni, dla których
#będą pobierane dane od użytkownika.
```

Politechnika Wrocławska

Lista jest modyfikowalna

01_tworzenie_listy.py

File Edit Format Run Options Window Help

```
NUM_DAYS = 7

def main():
    #utworzenie listy o ustalonej liczbie
    #elementów - wszytskie równe zero
    temperatures = [0]*NUM_DAYS

    #pobranie od użytkownika temperatur z kolejnych dni
    print('Podaj temperatury odnotowane przez 7 dni.')
    index = 0
    while index < NUM_DAYS:
        print('Dzień nr ',index+1,': ',sep='',end='')
        temperatures[index] = float(input())
        index += 1

    #wyświetlenie pobranych danych
    print('Odnotowane temperatury')
    for t in temperatures:
        print(t)

main()
```

Politechnika Wrocławska

Konkatencja list

- Do połączenia (konkatencji) list używa się operatora +

```
>>> lista1 = [1, 3, 5, 7]
>>> lista2 = [2, 4, 6, 8]
>>> lista3 = lista1 + lista2
>>> lista3
[1, 3, 5, 7, 2, 4, 6, 8]
>>> lista1
[1, 3, 5, 7]
>>> lista2
[2, 4, 6, 8]
```

Politechnika Wrocławska

Konkatencja list

- Do połączenie list można użyć złożonego operatora przypisania **+=**

```
>>> lista1 = [1, 3, 5, 7]
>>> lista2 = [2, 4, 6, 8]
>>> lista3 = lista1 + lista2
>>> lista3
[1, 3, 5, 7, 2, 4, 6, 8]
>>> lista1
[1, 3, 5, 7]
>>> lista2
[2, 4, 6, 8]
>>> lista2 += lista1
>>> lista2
[2, 4, 6, 8, 1, 3, 5, 7]
```

Politechnika Wrocławska

Wycinek listy

- Z wykorzystaniem indeksu można uzyskać dostęp do pojedynczego elementu listy
- Python pozwala także tworzyć wycinek obejmujący pewien fragment listy

***nazwa_listy*[*początek*:*koniec*]**

Politechnika Wrocławska

Wycinek listy

- Przykładowo:

```
>>> days_names = ['pn', 'wt', 'śr', 'cz', 'pt', 'sb', 'nd']
>>> days_names[0:5]
['pn', 'wt', 'śr', 'cz', 'pt']
>>> days_names[2:5]
['śr', 'cz', 'pt']
>>>
```

Politechnika Wrocławska

Wycinek listy

- Pominięcie jednego z indeksów (początku lub końca) pozwala stworzyć wycinek listy od początku lub do końca:

```
>>> days_names[2:]
['śr', 'cz', 'pt', 'sb', 'nd']
>>> days_names[:5]
['pn', 'wt', 'śr', 'cz', 'pt']
>>>
```

Politechnika Wrocławska

Wycinek listy

- Pominięcie obu indeksów pozwala stworzyć kopię całej listy:

```
>>> days_names[:]
['pn', 'wt', 'śr', 'cz', 'pt', 'sb', 'nd']
>>>
```

Politechnika Wrocławska

Wycinek listy

- Wycinek może zawierać również elementy listy z podanym krokiem:

```
>>> days_names[0:7:2]
['pn', 'śr', 'pt', 'nd']
>>> days_names[0::2]
['pn', 'śr', 'pt', 'nd']
>>>
```

Politechnika Wrocławska

Wycinek listy

- Do wskazywania elementów wycinka można używać również ujemnych indeksów

```
>>> days_names[:-2]
['pn', 'wt', 'śr', 'cz', 'pt']
>>>
```

Politechnika Wrocławska

Wycinek listy

- Nieprawidłowy indeks nie spowoduje błędu:

```
>>> days_names[:10]
['pn', 'wt', 'śr', 'cz', 'pt', 'sb', 'nd']
>>> days_names[-10:]
['pn', 'wt', 'śr', 'cz', 'pt', 'sb', 'nd']
>>> days_names[5:2]
[]
>>>
```

Politechnika Wrocławska

Wyszukiwanie elementu listy

- Operator **`in`** pozwala sprawdzić, czy element znajduje się na liście
- Wywołanie:

  `element` **`in`** ***`lista`***

Wyszukiwanie elementu listy

02_operator_in.py

File Edit Format Run Options Window Help

```
# Program pokazuje sprawdzanie czy podany
# element znajduje się na liście.
# Program pokazuje użycie operatora in.

def main():
    #utworzenie listy kolorów podstawowych
    primary_colours = ['czerwony', 'zielony', 'niebieski']

    #pobranie nazwy koloru od użytkownika
    colour = input('Podaj nazwę koloru: ')

    #sprawdzenie, czy podany kolor jest na liście
    if colour in primary_colours:
        print('Kolor',colour,'jest kolorem podstawowym.')
    else:
        print('Kolor',colour,'nie jest kolorem podstawowym.')


main()
```

Politechnika Wrocławska

Wybrane metody listy

metoda	opis
`append(`*`element`*`)`	dołącza element na końcu listy
`index(`*`element`*`)`	zwraca najniższy indeks, którego wartość odpowiada elementowi
`sort()`	porządkowanie elementów w kolejności rosnącej
`insert(`*`indeks, element`*`)`	wstawienie elementu na pozycji indeks
`remove(`*`element`*`)`	usuwa element z listy
`reverse()`	odwraca kolejność elementów na liście

Wybrane metody listy

append()

03_append.py

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody append()
# do dołączania elementów do listy.

# Program prosi użytkownika o wymienianie
# nazw kolorów jakie zna.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #dodanie nazwy do listy
        colours.append(colour)

        #pytanie czy kontynouwać wykonywanie pętli
        print('Czy znasz jeszcze jakieś kolory?')
        keep_going = input('Jeśli tak, wpisz t,'+
                           ' w przeciwnym razie inny znak: ')

    #wypisanie wszystkich kolorów
    print('Wszystkie kolory jakie podałeś:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

append()

03_append.py

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody append()
# do dołączania elementów do listy.

# Program prosi użytkownika o wymienianie
# nazw kolorów jakie zna.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #dodanie nazwy do listy
        colours.append(colour)

        #pytanie czy kontynouwać wykonywanie pętli
        print('Czy znasz jeszcze jakieś kolory?')
        keep_going = input('Jeśli tak, wpisz t,'+
                           ' w przeciwnym razie inny znak: ')

    #wypisanie wszystkich kolorów
    print('Wszystkie kolory jakie podałeś:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

append()

04_append_in.py

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody append()
# do dołączania elementów do listy.
# Przed dodaniem koloru do listy sprawdza, czy
# podany kolor się nie powtarza.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #jeśli podanej nazwy nie ma liście
        if colour not in colours:
            #to ją dołącz
            colours.append(colour)
        else:
            #w p. p. wyświetl komunikat i przerwij pętle
            print('Kolor',colour,'jest już na liście.')
            keep_going = 'n'

    #wypisanie wszystkich kolorów
    print('Na liście są następujące kolory:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

append()

04_append_in.py

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody append()
# do dołączania elementów do listy.
# Przed dodaniem koloru do listy sprawdza, czy
# podany kolor się nie powtarza.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #jeśli podanej nazwy nie ma liście
        if colour not in colours:
            #to ją dołącz
            colours.append(colour)
        else:
            #w p. p. wyświetl komunikat i przerwij pętle
            print('Kolor',colour,'jest już na liście.')
            keep_going = 'n'

    #wypisanie wszystkich kolorów
    print('Na liście są następujące kolory:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

index()

05_index.py -

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody index()
# do odnalezienia pozycji elementu na liście.
# Przed dodaniem koloru do listy sprawdza, czy
# podany kolor się nie powtarza. Jeśli się powtarza,
# to sprawdza, na której pozycji jest powtórzenie.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #jeśli podanej nazwy nie ma liście
        if colour not in colours:
            #to ją dołącz
            colours.append(colour)
        else:
            #w p. p. wyświetl komunikat i przerwij pętle
            print('Kolor ',colour,' jest już na liście. ',
                  'Na pozycji ',colours.index(colour),'.',sep='')
            keep_going = 'n'

    #wypisanie wszystkich kolorów
    print('Na liście są następujące kolory:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

index()

05_index.py -

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody index()
# do odnalezienia pozycji elementu na liście.
# Przed dodaniem koloru do listy sprawdza, czy
# podany kolor się nie powtarza. Jeśli się powtarza,
# to sprawdza, na której pozycji jest powtórzenie.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #jeśli podanej nazwy nie ma liście
        if colour not in colours:
            #to ją dołącz
            colours.append(colour)
        else:
            #w p. p. wyświetl komunikat i przerwij pętle
            print('Kolor ',colour,' jest już na liście. ',
                  'Na pozycji ',colours.index(colour),'.',sep='')
            keep_going = 'n'

    #wypisanie wszystkich kolorów
    print('Na liście są następujące kolory:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

`sort()`

06_sort.py

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody sort()
# do uporządkowania listy elementów.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #jeśli podanej nazwy nie ma liście
        if colour not in colours:
            #to ją dołącz
            colours.append(colour)
        else:
            #w p. p. wyświetl komunikat i przerwij pętle
            print('Kolor ',colour,' jest już na liście. ',
                  'Na pozycji ',colours.index(colour),'.',sep='')
            keep_going = 'n'

    #sortowanie listy w kolejności alfabetycznej
    colours.sort()

    #wypisanie wszystkich kolorów
    print('Na liście są następujące kolory:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

`sort()`

06_sort.py

File Edit Format Run Options Window Help

```
# Program pokazuje użycie metody sort()
# do uporządkowania listy elementów.

def main():
    #utworzenie pustej listy
    colours = []

    #zmienna kontroluje przebieg pętli while
    keep_going = 't'
    while keep_going == 't':
        #pobranie nazwy koloru od użytkownika
        colour = input('Podaj kolor: ')

        #jeśli podanej nazwy nie ma liście
        if colour not in colours:
            #to ją dołącz
            colours.append(colour)
        else:
            #w p. p. wyświetl komunikat i przerwij pętle
            print('Kolor ',colour,' jest już na liście. ',
                  'Na pozycji ',colours.index(colour),'.',sep='')
            keep_going = 'n'

    #sortowanie listy w kolejności alfabetycznej
    colours.sort()

    #wypisanie wszystkich kolorów
    print('Na liście są następujące kolory:')
    for colour in colours:
        print(colour)


main()
```

Wybrane metody listy

`insert()`

```
>>> lista = [] #utworzenie pustej listy
>>> lista.insert(0,'k') #wstawienie 'k' na pozycję 0
>>> print(lista)
['k']
>>> lista.insert(0,'g') #wstawienie 'g' na pozycję 0
>>> print(lista)
['g', 'k']
>>> #'k' zostało przesunięte
>>> lista.insert(0,'a')
>>> print(lista)
['a', 'g', 'k']
>>> lista.insert(1,'b')
>>> print(lista)
['a', 'b', 'g', 'k']
>>>
```

Wybrane metody listy

`insert()`

- Użycie indeksu przekraczającego długość listy nie spowoduje błędu
- Element zostanie wstawiony na koniec lub początek listy

```
>>> lista.insert(10,'z')
>>> print(lista)
['a', 'b', 'g', 'k', 'z']
>>> lista.insert(-10,'z')
>>> print(lista)
['z', 'a', 'b', 'g', 'k', 'z']
>>>
```

Politechnika Wrocławska

Wybrane metody listy

`remove()`

- Metoda **`remove()`** usuwa z listy pierwsze wystąpienie elementu

```
>>> print(lista)
['z', 'a', 'b', 'g', 'k', 'z']
>>> lista.remove('z')
>>> print(lista)
['a', 'b', 'g', 'k', 'z']
>>> lista.remove('z')
>>> print(lista)
['a', 'b', 'g', 'k']
>>> lista.remove('z')
Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    lista.remove('z')
ValueError: list.remove(x): x not in list
>>>
```

Wybrane metody listy

`reverse()`

- Metoda **`reverse()`** odwraca porządek elementów na liście

```
>>> print(lista)
['a', 'b', 'g', 'k']
>>> lista.reverse()
>>> print(lista)
['k', 'g', 'b', 'a']
>>>
```

Usuwanie elementu listy

`del`

- Do usunięcia elementu listy można użyć polecenia **`del`**

```
>>> print(lista)
['k', 'g', 'b', 'a']
>>> del lista[1]
>>> print(lista)
['k', 'b', 'a']
>>>
```

Znajdowanie minimum i maksimum

- Funkcje **`min()`** i **`max()`** zwracają wartość minimalnego i maksymalnego elementu z listy

```
>>> print(lista)
['k', 'b', 'a']
>>> min(lista)
'a'
>>> max(lista)
'k'
>>>
```

Politechnika Wrocławska

Kopiowanie listy

- Przypisanie jednej zmiennej do drugiej powoduje odwoływanie się obu zmiennych do tego samego obszaru pamięci

```
>>> list1 = ['a', 'b', 'c', 'd']
>>> list2 = list1
>>> print(list1)
['a', 'b', 'c', 'd']
>>> print(list2)
['a', 'b', 'c', 'd']
>>>
```

Politechnika Wrocławska

Kopiowanie listy

- Przypisanie jednej zmiennej do drugiej powoduje odwoływanie się obu zmiennych do tego samego obszaru pamięci

```
>>> list1 = ['a', 'b', 'c', 'd']
>>> list2 = list1
>>> print(list1)
['a', 'b', 'c', 'd']
>>> print(list2)
['a', 'b', 'c', 'd']
>>>
```

Politechnika Wrocławska

Kopiowanie listy

- Przypisanie jednej zmiennej do drugiej powoduje odwoływanie się obu zmiennych do tego samego obszaru pamięci

```
>>> list1 = ['a', 'b', 'c', 'd']
>>> list2 = list1
>>> print(list1)
['a', 'b', 'c', 'd']
>>> print(list2)
['a', 'b', 'c', 'd']
>>>
```

Politechnika Wrocławska

Kopiowanie listy

- Obie zmienne wskazują na ten sam obszar pamięci. Zatem zmiana wartości elementu jest widoczna przez obie zmienne.

```
>>> list1[1] = 'z'
>>> print(list1)
['a', 'z', 'c', 'd']
>>> print(list2)
['a', 'z', 'c', 'd']
>>>
```

Politechnika Wrocławska

Kopiowanie listy

- Do kopiowania listy można wykorzystać metodę **`copy()`**

```
>>> list1 = ['a','b','c','d']
>>> list2 = list1.copy()
>>> list1[1] = 'z'
>>> print(list1)
['a', 'z', 'c', 'd']
>>> print(list2)
['a', 'b', 'c', 'd']
>>>
```

Politechnika Wrocławska

Przetwarzanie elementów listy

07_przetwarzanie_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje przetwarzanie listy.
# Program oblicza średnią liczb z listy.

def main():
    #lista z przykładowymi danymi
    data = [1,2,3,5,8,13]

    #obliczenie sumy elementów z listy
    total = 0
    for value in data:
        total += value
    #obliczenie średniej elementów
    average = total / len(data)
    #wypisanie wyniku
    print('Średnia danych z listy to',average)

main()
```

Politechnika Wrocławska

Przetwarzanie elementów listy

08_przetwarzanie_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje przetwarzanie listy.
# Program oblicza średnią liczb z listy.

def main():
    #lista z przykładowymi danymi
    data = [1,2,3,5,8,13]
    #wywołanie funkcji average()
    #lista data przekazana jako argument
    print('Średnia danych z listy to',average(data))

#I: lista zawierająca dane do obliczeń
#P: funkcja oblicza sumę liczb na liście,
    # a następnie zwraca średnią
    # (iloraz sumy elementów i liczby elementów)
#O: średnia elementów listy
def average(data):
    total = 0
    for value in data:
        total += value
    return total / len(data)

main()
```

Przetwarzanie elementów listy

08_przetwarzanie_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje przetwarzanie listy.
# Program oblicza średnią liczb z listy.

def main():
    #lista z przykładowymi danymi
    data = [1,2,3,5,8,13]
    #wywołanie funkcji average()
    #lista data przekazana jako argument
    print('Średnia danych z listy to',average(data))

#I: lista zawierająca dane do obliczeń
#P: funkcja oblicza sumę liczb na liście,
    # a następnie zwraca średnią
    # (iloraz sumy elementów i liczby elementów)
#O: średnia elementów listy
def average(data):
    total = 0
    for value in data:
        total += value
    return total / len(data)

main()
```

Politechnika Wrocławska

Przetwarzanie elementów listy

09_przetwarzanie_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje przetwarzanie listy.
# Program oblicza średnią liczb z listy.

def main():
    data = get_data(5)
    print('Średnia danych z listy to',average(data))

#funkcja zwraca średnią z elementów listy data
def average(data):
    total = 0
    for value in data:
        total += value
    avg = total / len(data)
    return avg

#I: liczba n reprezentująca długość listy
#P: wczytanie n liczb rzeczywistych i dodanie ich
    #do listy
#O: wczytana lista danych
def get_data(n):
    data = []
    for i in range(n):
        data.append(float(input('Podaj liczbę: ')))
    return data


main()
```

Przetwarzanie elementów listy

09_przetwarzanie_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje przetwarzanie listy.
# Program oblicza średnią liczb z listy.

def main():
    data = get_data(5)
    print('Średnia danych z listy to',average(data))

#funkcja zwraca średnią z elementów listy data
def average(data):
    total = 0
    for value in data:
        total += value
    avg = total / len(data)
    return avg

#I: liczba n reprezentująca długość listy
#P: wczytanie n liczb rzeczywistych i dodanie ich
    #do listy
#O: wczytana lista danych
def get_data(n):
    data = []
    for i in range(n):
        data.append(float(input('Podaj liczbę: ')))
    return data


main()
```

Politechnika Wrocławska

Listy i pliki

10_zapis_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje zapisywanie elementów listy do pliku.
# Program wykorzystuje metodę writelines obiektu plikowego.

def main():
    courses = ['Analiza\n', 'Algebra\n', 'Fizyka\n']

    outfile = open('lista.txt','w')
    #funkcja writlines() zapisuje elementy listy w pliku
    outfile.writelines(courses)
    outfile.close()

main()
```

Politechnika Wrocławska

Listy i pliki

11_odczyt_listy.py

File Edit Format Run Options Window Help

```
# Program pokazuje odczyt listy z pliku.
# Program wykorzystuje metodę readlines obiektu plikowego.

def main():
    infile = open('lista.txt','r')
    #Funkcja readlines() odczytuje z pliku wiersze
    #i zapisuje je w liście
    courses = infile.readlines()
    infile.close()

    print(courses)

main()
```

Politechnika Wrocławska

Listy zagnieżdżone

```
>>> magiczny_kwadrat = [[6,1,8],[7,5,3],[2,9,4]]
>>> print(magiczny_kwadrat)
[[6, 1, 8], [7, 5, 3], [2, 9, 4]]
>>> magiczny_kwadrat[0]
[6, 1, 8]
>>> magiczny_kwadrat[0][0]
6
>>> magiczny_kwadrat[2][1]
9
>>>
```

Politechnika Wrocławska

Krotka

- Krotka to niemodyfikowalna sekwencja
- Do zapisania krotki używa się nawiasów okrągych

```
>>> tuple1 = (0,1,2)
>>> tuple1[1] = 2
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    tuple1[1] = 2
TypeError: 'tuple' object does not support item assignment
>>>
```

Politechnika Wrocławska

Krotka

- Krotka pozwala na odwoływanie się do elementów przez index
- Krotka może być argumentem funkcji **`len()`**, **`min()`**, **`max()`**
- Można tworzyć wycinki z krotki
- Krotka działa z operatorami **`in`**, **`+`**, **`*`**

Politechnika Wrocławska

Krotka

- Funkcje **`list()`** i **`tuple()`** pozwalają przeprowadzić konwersję między krotką i listą

```
>>> krotka = (1,2,3)
>>> lista = list(krotka)
>>> print(lista)
[1, 2, 3]
>>> type(lista)
<class 'list'>
>>>
```

```
>>> lista = [4,5,6]
>>> krotka = tuple(lista)
>>> print(krotka)
(4, 5, 6)
>>> type(krotka)
<class 'tuple'>
>>>
```

Politechnika Wrocławska

Podsumowanie (1)

- Operator powtórzenia `*`
- Listy i pętla `for`
- Indeksowanie elementów `[]`
- Długość listy `len()`
- Modyfikowanie list
- Konkatenacja list `+`
- Wycinek listy `[a:b]`

Politechnika Wrocławska

Podsumowanie (2)

- Wyszukiwanie elementu listy **`in`**
- Wybrane metody listy
- Kopiowanie listy **`copy()`**
- Przetwarzanie danych
- Listy i pliki
- Listy zagnieżdżone
  `[[1,2,3],[4,5,6],[7,8,9]]`

- Krotki **`tuple()`** `(1,2,3)`