Podstawy Informatyki - Język programowania Python
Przetwarzanie struktur danych – listy, krotki, słowniki
(opracował dr inż. Paweł Stąpór)

1. **Struktury danych: listy, krotki, słowniki**

Ćwiczenie 1. Listy tworzymy używając nawiasów kwadratowych, rozdzielając ich elementy przecinkami, przykłady

```
>>>Lista_1 = [1, 2, 3]
>>>Lista_2=[‘jeden’, ‘dwa’, ‘trzy’]
```

Listy mogą zawierać elementy innych typów

```
>>>Lista_3=[1, 2, 3, ‘jeden’, ‘dwa’, ‘trzy’]
```

I inne listy

```
>>>Lista_4[[1, 2, 3], Lista_3, [‘jeden’, ‘dwa’]]
```

Listy mogą być puste

```
>>>Lista_5 = []
```

Można odczytywać wybiórczo zawartość poszczególnych elementów listy

```
>>>Lista_4[1][0]
>>>Lista_3[0:2]
```

Listy można modyfikować w miejscu, powielać, skracać i wydłużać

```
>>>List_1[0]=Lista_2[0]
>>>Lista_1 = Lista_1*2
>>>Lista_3 = Lista_3[:3]
>>>Lista_3+=[’trzy’]
```

Metoda len(lista) zwraca długość listy

```
>>>len(Lista_1*2)
```

Metoda append( ) dołącza do listy pojedynczy element:

```
>>> Lista_1.append(33)
```

Metoda extend( ) dołącza do listy inną listę:

```
>>> Lista_1.extend([33,99])
```

Metoda count(w) liczy ile razy występuje na liście wartość w:

```
>>> Lista_1.count(33)
```

Metoda index(w) znajduje pierwszą pozycję listy na której występuje wartość w:

```
>>> Lista_1.index(33)
```

Metoda insert(i, w) wstawia na pozycję i listy wartość w:

```
>>> Lista_1.insert(5,77)
```

Metoda pop(i) zwraca wartość z pozycji i listy, po czym usuwa tę pozycję:

```
>>> Lista_1.pop(5)
```

Metoda remove(w) usuwa z listy pierwszą znalezioną na liście wartość w:

```
>>> Lista_1.remove(3)
```

Metoda reverse( ) odwraca kolejność elementów listy:

```
>>> Lista_1.reverse()
```

Metoda sort( ) porządkuje elementy listy w kolejności rosnącej:

```
>>> Lista_1.sort()
```

Funkcja range() służy do automatycznego tworzenia listy składającej się z ciągu arytmetycznego liczb

```
>>>range(10)
>>>range(1,10)
>>>range(1,10,2)
>>>range(9,0,-2)
```

Ćwiczenie 2. Krotki pod wieloma względami przypominają listy, w podobny sposób tworzymy je i sprawdzamy ich wartości. W odróżnieniu od list, krotki są sekwencjami niezmiennymi, co powoduje różnice w sposobie ich modyfikacji.

Krotki tworzymy używając nawiasów okrągłych, rozdzielając ich elementy przecinkami:

```
>>> krotka1=(1,2,3)
```

Krotki mogą zawierać elementy różnych typów:
>>> krotka2=(1.0, 2, "trzy")
W tym sekwencyjnych
>>> krotka3=(krotka1, lista1)
Można odczytywać wybiórczo zawartość poszczególnych elementów krotki:
>>> krotka1[1]
>>> krotka3[-1]
Lub ich ciągów:
>>> krotka1[1:]
>>> krotka1[::2]
Można je zmieniać tworząc nowy obiekt krotki o tej samej nazwie:
>>> krotka1*=2
>>> krotka1=krotka1[:3]
>>> krotka1(1, 2, 3)
>>> krotka1=krotka1+(4,)
Krotki nie można modyfikować w miejscu
>>>krotka1(1) = ‘a’
Nie obsługują takich metod jak listy, jedynymi są metody
>>>krotka1.index(1)
>>>krotka1.count(3)

Ćwiczenie 3. W dwóch omówionych dotąd typach sekwencji – listach, krotkach – dostęp do dowolnego elementu możliwy był poprzez podanie jego indeksu. Odmiennego rodzaju typem złożonym jest słownik. W słowniku dostęp do dowolnej wartości przechowywanej w słowniku możliwy jest poprzez podanie klucza do niej.

Słownik składa się zatem ze zbioru kluczy i zbioru wartości, gdzie każdemu kluczowi przypisana jest pojedyncza wartość. Zależność między kluczem a jego wartością nazywana bywa odwzorowaniem. Klucz nie musi być liczbą, tak jak jest nią indeks, wystarczy, że jest typu niezmiennego. Można powiedzieć więc, że o ile lista czy krotka odwzorowuje liczby całkowite (indeksy) na obiekty dowolnego typu, o tyle słownik odwzorowuje obiekty dowolnego typu niezmiennego na obiekty dowolnego typu.

W języku Python do tworzenia słowników używamy nawiasów klamrowych, np.:
>>> tel = {"policja":997, "straz":998, "pogotowie":999}
Klucze nie muszą być tekstem, a wartości liczbami:
>>> bohater={"hans":"kloss","james":"bond"}
>>> ujemne={7:-7,3:-3}
Aby otrzymać wartość podanego klucza używamy nawiasów kwadratowych:
>>> tel ["policja"]
>>> bohater["hans"]
Możemy dopisać nowy klucz
>>> tel["taxi"]=919
Możemy zmodyfikować istniejącą wartość:
>>> tel["taxi"]=9622
Zapis
>>> tel2=tel
nie skopiuje zawartości tel do tel2 ale jedynie stworzy drugie odwołanie do tych wartości. A zatem operacja
>>> tel2["taxi"]=9666
zmieni zarówno wartości tel jak i tel2, gdyż są to te same dane pod dwoma różnymi nazwami:
>>> tel
Aby skopiować zawartość jednego słownika do drugiego używamy metody copy:

>>> tel2=tel.copy()
W ten sposób otrzymaliśmy dwa różne słowniki, początkowo o tych samych wartościach.
>>> tel["taxi"]=9622
>>> tel
>>> tel2
Aby sprawdzić zawartość określonego klucza w słowniku używamy operatora in:
>>> 'taxi' in tel
>>> 'taxi' in tel2
Można też użyć w tym celu metody __contains__():
>>> tel2.__contains__("pogotowie")
Aby uzyskać listę kluczy występujących w słowniku, używamy metody keys:
>>> tel.keys()
Aby uzyskać listę wartości występujących w słowniku, używamy metody values:
>>> tel.values()
Można w ten sposób sprawdzić występowanie określonych wartości:
>>> 999 in tel.values()
Aby usunąć cały słownik:
>>> del tel

Ćwiczenie 4.
Przykład pętli for przebiegającej po elementach listy, łańcucha, krotki i słownika

```
#Wyświetlenie elementów listy
Lista_zakupów = ["mielonka", "jajka", "szynka"]
for x in Lista_zakupów:
        print(x, end=' ')

#Suma po elementach listy
suma = 0
for x in [1, 2, 3, 4]:
    suma+=x
print('\nSuma=',suma)

#Iloczyn z elemntów listy
iloczyn = 1
for x in [1, 2, 3, 4]:
    iloczyn*=x
print('Iloczyn=',iloczyn)

#Iteracja po elementach łańcucha
S = 'drwal'
for x in S: print(x, end=' ')
print()

#Iteracja po elementach krotki
T = ("ja", "jestem", "drwalem")
for x in T: print(x, end=' ')

#Iteracja po elementach słownika
D = {'a':1, 'b':2, 'c':3}
for klucz in D:
    print(klucz, '=>', D[klucz])

print(list(D.items()))

for (klucz, wartość) in D.items():
    print(klucz, ' : ', wartość)
```

Ćwiczenie 5. Przykład programu, który wczytuje elementy do listy **numbers** , wyznacza średnią dla tych liczb a następnie drukuje pozycje elementów listy większych od obliczonej średniej. Funkcja split() dzieli łańcuch według znaku domyślnego - znaku spacji.

```
string = input("Podaj listę a, np. 2 -2.8 3.4 4 \n")
numbers = list(map(float, string.split()))
print(numbers)

s = 0;

for x in numbers:
    s = s + x
n = len(numbers)
s = float(s)/n
print('średnia = {:.2f}'.format(s))
for i in range(n):
    if numbers[i]>s:
        print('i = ', i)
```

Zadanie 1. Dana jest lista liczb całkowitych. Opracować program *mini.py*, który wczytuje te liczby do listy *b*, wyznacza element minimalny a następnie drukuje pozycje (numery) elementów listy równych elementowi minimalnemu.
Przykład
Dane:
Lista liczb: [1 , 2, 1, 3, 5]
Wyniki: min= 1 pozycje: 0 2

Zadanie 2. Dana jest lista liczb całkowitych. Opracować program *lista.py*, który wczytuje te liczby do listy *c* a następnie wyznacza:

a) ile jest liczb ujemnych ,
b) sumę kwadratów liczb,
c) średnią z liczb dodatnich,

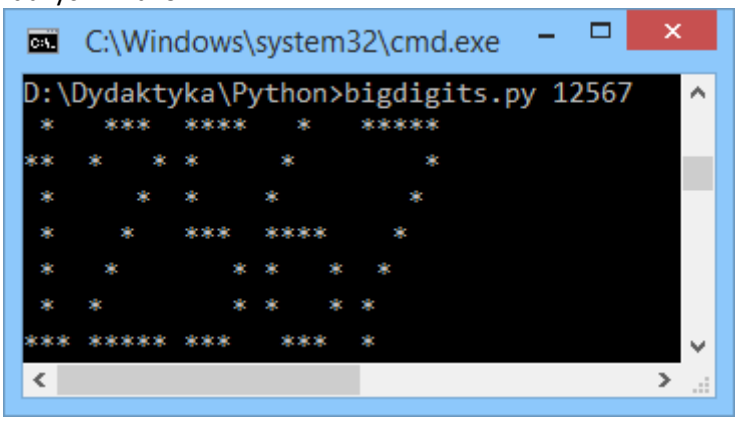
Przykład Dane:
Lista liczb: [- 2 , 7, 1, 4 , -1, 5]
Wyniki: Ujemnych jest :2
Suma kwadratów: 96
Średnia z dodatnich: 4.25

Zadanie 3. Woda zamarza przy 32 stopniach Fahrenheita, a wrze przy 212 stopniach Fahrenheita. Napisz program, który wyświetli tabelę przeliczeń stopni Celsjusza na stopnie Fahrenheita w zakresie od –20 do +40 stopni Celsjusza (co 5 stopni).

Zadanie 4. Listy x i y zawierają wyniki spotkań piłkarskich. Na liście x zapisana jest liczba goli zdobytych przez gospodarzy a na liście y – przez gości.

a) Wyznacz średnią liczbę goli zdobytych przez gości w spotkaniach zakończonych zwycięstwem gospodarzy
b) Wyznacz średnią liczbę bramek strzelonych w meczu
c) Wyznacz liczbę spotkań zakończonych remisem

Zadanie 5. Napisz program *bigdigits.py*, który dla danego ciągu liczb wyświetla liczby w postaci ‘dużych’ znaków.



Pobranie argumentu wywołania programu

```
import sys
digits = sys.argv[1]
```

Zadanie 6. Napisz program, który zamieni całkowitą liczbę dziesiętną na odpowiadającą jej liczbę rzymską. Wykorzystaj zdefiniowany słownik:
{1000:"M", 900:"CM", 500:"D", 400:"CD", 100:"C", 90:"XC", 50:"L",40:"XL", 10:"X", 9:"IX", 5:"V", 4:"IV", 1:"I"}