Typy danych w języku Python

Typy proste
- Logiczny (bool)
- Całkowity (int)
- Zmiennopozycyjny (float)
- Zespolony (complex)
- Napisowy (str)

Typy strukturalne
- Zbiór (set)
- Krotka (tuple)
- Lista (list)
- Słownik, tabela (dict)
- Plik (file)

1

Zbiory

- Nieuporządkowana kolekcja elementów
- Elementy zbioru mogą być różnych typów
- Podstawowe operacje na zbiorach:

```
s = set()              # twórz zbiór pusty
s2 = set([2,3,5])      # twórz zbiór z listy
len(s)                 # moc zbioru
s.add(el)              # wstaw element do zbioru
s.remove(el)           # usuń istniejący element ze zbioru
s.discard(el)          # usuń element ze zbioru
s.clear()              # opróżnij zbiór
if el in s:            # czy należy
if el not in s:        # czy nie należy
s1.union(s2)           # suma mnogościowa zbiorów s1 i s2
s1.intersection(s2)    # część wspólna zbiorów  s1 i s2
```

2

Listy

- Uporządkowane kolekcja elementów
- Elementy listy mogą być różnych typów
- Elementy listy mogą być zmieniane
- Rozmiar listy może się zmieniać
- Elementy dostępne poprzez indeksowanie
- Operacje na listach analogiczne do operacji na napisach a+b, a*3, a[0], a[-1], a[1:], len(a)

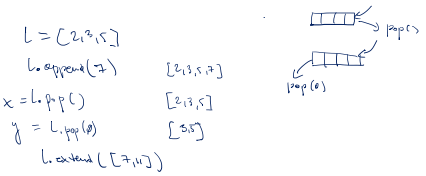
Przykład:

```
lista1 = [ ]
lista2 = [2,3,5,7]
lista3 = [23,"ala",[1,2,3]]
```

3

Podstawowe operacje na listach

- lista.append(element) – dołącz element do listy
- lista.pop() – pobierz element z listy
- lista.extend(lista) – dołącz kilka elementów
- lista.sort() – posortuj listę w miejscu
- lista2 = sorted(lista) – stwórz posortowaną listę
- lista.reverse() – odwróć kolejność elementów



4

Wyrażenia listowe

```
[ expr for x in collection ]
[ expr for x in collection if warunek]

kwadraty = [ i**2 for i in range(100) ]
kwadraty_zakonczone_6 = [ i**2 for i in range(100) if i**2%10==6 ]

osoby = [ ('ola',18), ('ula',16), ('ala',19), ... ]
pełnoletni = [os for (os,wiek) in osoby if wiek>=18 ]

lx = [ x for x in range(-10,11) ]
ly = [ f(x) for x in lx ]

la = [ 2,3,5 ]
lb = [ 7,11,13 ]

iloczyn_kartezjanski = [ (a,b) for a in la for b in lb ]

LL = [ [3,5,7], [2,4,8] ]
[ x**2 for L in LL for x in L]
```

5

Tablice w Pythonie

```
t =[]
for i in range(100): t.append(0)
t[6] = 23

t  = [ 0 for _ in range(100) ]

t ={}
for i in range(100): t[i] = 0
t[6] = 23

from numpy import *
t =zeros(100,float)
t[6] = 0.1428
```

```
t =[]
for i in range(8):
  t.append([])
  for j in range(8): t[i].append(0)
t[6][5] = 23

t = {}
for i in range(8):
  for j in range(8): t[i,j] = 0
t[5,6] = 29

from numpy import *
t =zeros((8,8),float)
t[6][6] = 0.1428
```

6

Krotki

- Krotka (tuple) analogiczna do listy ale bez możliwości zmian zawartości

Przykłady:

```
k1 = ( )
k2 = ( 'ala', 'ola', 'ula' )

a,b,c = k2
lista = list(k2)
k3 = tuple(lista)

x1,x2 = rk(a,b,c)   # funkcja zwraca krotkę
```

7

Krotki

Przykład:

```
rm = (0,0,3,3,6,1,4,6,2,5,0,3,5)
dni = ('niedziela','poniedziałek','wtorek','sroda','czwartek','piatek','sobota')

d  = int(input("dzien 1-31      : "))
m  = int(input("miesiąc 1-12    : "))
r  = int(input("rok 1900-2099 : "))
dt = d + rm[m] + (r-1900) + (r-1900)//4
if r%4==0 and m<3: dt=dt-1
dt = dt%7

print(dni[dt])
```

8

Słowniki

- Słownik, mapa, tabela, tablica asocjacyjna, tablica z haszowaniem
- Nieuporządkowana kolekcja par klucz/wartość
- Klucz najczęściej jest liczbą lub napisem
- Klucz musi być unikalny
- Wartość może być dowolnego typu

Przykład:

```
t1 = { }
t2 = { 'ala':6, 'ola':12, 'jan':23 }
```

9

Słownik – funkcje, metody

- h[klucz] – wartość dla klucza
- h[klucz]=wartosc – przypisanie wartości
- del h[klucz] – usuń wartość ze słownika
- h.clear() – usuwa słownik
- len(h) – liczba pozycji w słowniku
- h.get(klucz,default) – wartość dla klucza lub default
- h.keys() – lista kluczy
- h.values() – lista wartości
- h.items() – lista par
- h.copy() – kopia słownika

10

Wyrażenia słownikowe

```
{ key : value for (key,value) in collection }

osoby = [ ('ola',18), ('ula',16), ('ala',19) ]

dic1 = {key : value for (key,value) in osoby}
dic2 = {key : value for (value,key) in dic1.items()}

t = ( i**3 for i in range(10) )

for i in t:
  print(i)
```

11

Pliki

- f = open(*filename* [, *tryb* [, *buffersize*])
  - tryb: "r", "w", "a" ; default "r"
  - buffersize: 0=unbuffered; 1=line-buffered; buffered
- metody:
  - read([*nbytes*])
  - readline() – pojedyńcza linia
  - readlines() – wszystkie linie jako lista
  - write(*string*)
  - writelines(*list*)
  - seek(*pos*)
  - flush()
  - close()

12

Pliki - przykład

Przykład:

```
t = {}

f=open("pap.txt","r")
for line in f:
    line=line.strip('\n').lower()
    for z in line:
        t.setdefault(z,0)
        t[z]+=1
    # end for
# end for
f.close()

for k in t:
    print(k,t[k])
```

13

Pliki - przykład

Można inaczej:

```
from collections import defaultdict

t = defaultdict(int)
with open("pap.txt","r") as f:
    for line in f:
        line=line.strip('\n').lower()
        for z in line:
            t[z]+=1
        # end for
    # end for
# end with

for k in t:
    print(k,t[k])
```

14

Pliki - przykład

Plik w postaci:
Nowak Jan ; Informatyki ; profesor ; 180

```
from collections import namedtuple

Osoba = namedtuple('Osoba', 'katedra stanowisko pensum')
osoby = {}
with open(„dane.csv","r") as f:
    for line in f:
        lista=line.strip('\n').split(';')
        osoby[lista[0]] = Osoba( lista[1], lista[2], lista[3] )
    # end for
# end with

for naz in osoby:
    print( naz, t[naz].katedra, t[naz].stanowisko, t[naz].pensum )
```

15