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)

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

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
```

5

Krotki

Przykład:

```
rm = (-1,0,3,3,6,1,4,6,2,5,0,3,5)
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])
```

6

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 }
```

7

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

8

Tablice w Pythonie

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

```
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(100): t[i] = 0
t[6] = 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(100,float)
t[6] = 0.1428
```

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

9

Procedury i funkcje

Cel stosowania:

- dekompozycja problemu
- wielokrotne wykonanie
- poziomy abstrakcji
- oddzielna kompilacja
- możliwość użycia rekurencji

10

Procedury i funkcje

```
def name (arg1, arg2, ...):
    """ opis funkcji """
    instrukcje
    ...
# end
```

```
return                  # procedura
return expression       # funkcja
```

11

Procedury i funkcje

```
def nwd(a, b):
    """Funkcja oblicza największy wspólny dzielnik.
       Nie jest najszybsza ale działa."""
    while a != b:
        if a>b: a=a-b
        else: b=b-a
    return a
# end def

>>> help(nwd)
Help on function nwd in module my_lib:
nwd(a, b)
    Funkcja oblicza największy wspólny dzielnik.
    Nie jest najszybsza ale działa.

>>> nwd(24, 30)
6
```

12

Przekazywanie parametrów

- Argumenty typów niemodyfikowalnych (np. integer, string, krotka) są przekazywane przez wartość.
- Przykład
- ```
  def cube(x):
      x = x*x*x
      return x
  # end def

  n=2
  w=cube(n)
  print(n,w)        # 2 8
  ```

13

Przekazywanie parametrów

- Argumenty typów modyfikowalnych (np. zbiory, listy, słowniki) są przekazywane przez referencję.
- Przykład
- ```
  def zeruj(lista):
      for i in range(len(lista)):
          lista[i] = 0
      return
  # end def

  l=[2,3,5,7]
  zeruj(l)
  print(l)        # [0,0,0,0]
  ```

14

Przekazywanie wielu argumentów

```
>>> def f(*args): print(args)
...
> f()
()
> f(1)
(1,)
> f(1, 2, 3, 4)
(1, 2, 3, 4)


> def f(**args): print(args)
...
> f()
{}
> f(a=1, b=2)
{'a': 1, 'b': 2}
```

15

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* [, *how*]), tell()
  - flush(), close()
  - fileno()

17

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])
```

18

Wyjątki

```
def f(x):
    return 1/x

def f2(x):
    try:
        return 1/x
    except ZeroDivisionError:
        print(”Nie wolno dzielić przez zero!!!”)
        return 0

f2(0)
```

19

Wyjątki

```
try:
  f=open("pap.txt","r")

  for line in f:
    print(line)

  f.close()

except IOError:
  print("Coś jest nie tak!")
```

20

Biblioteki standardowe

- Core:
  - os, sys, string, getopt, StringIO, struct, pickle, ...
- Wyrażenia regularne:
  - re
- Internet:
  - socket, rfc822, httplib, htmllib, ftplib, smtplib, ...
- Różne:
  - pdb (debugger), profile+pstats
  - Tkinter (Tcl/Tk interface), audio, *dbm, ...

21

Przykłady funkcji

- sys.argv – lista argumentów
- sys.platform – identyfikacja systemu operacyjnego
- sys.exit() – zakończenie programu

- os.getcwd() – bieżący katalog
- os.mkdir() – twórz katalog
- os.rmdir() – usuń katalog
- os.system() – wykonaj polecenie

22