PYTHON – ŁAGODNE WPROWADZENIE

Statystyka biomedyczna

Artur Klepaczko

CO TO JEST PYTHON?

- Język programowania wysokiego poziomu, którego ideą przewodnią jest czytelność kodu - składnia cechuje się przejrzystością i zwięzłością.

- Python wspiera różne paradygmaty programowania: m.in. obiektowy i funkcyjny.

- Posiada dynamiczny system typowania i automatyczne zarządzanie pamięcią.

- Jest często używany jako język skryptowy.

TYPY DANYCH

Typ	Opis	Przykład
Python 3: `str` Python 2: `unicode`	Napis w Unikodzie (niezmienny)	Python 3: `'Wikipedia'` lub `"Wikipedia"` Python 2: `u'Wikipedia'` lub `u"Wikipedia"`
Python 3: `bytes` Python 2: `str`	Napis w ASCII (niezmienny)	Python 3: `b'Wikipedia'` lub `b"Wikipedia"` Python 2: `'Wikipedia'` lub `"Wikipedia"`
`list`	Lista (zmienna, zawartość, długość)	`[4.0, 'string', True]`
`tuple`	Krotka (niezmienna)	`(4.0, 'string', True)`
`set`	Zbiór różnych elementów (zmienny)	Python 3: `{4.0, 'string', True}` Python 2: `set([4.0, 'string', True])`
`frozenset`	Zbiór różnych elementów (niezmienny)	Python 3: `frozenset({4.0, 'string', True})` Python 2: `frozenset([4.0, 'string', True])`
`dict`	Słownik, czyli tablica asocjacyjna (zmienny).	`{'key1': 1.0, 3: False}`
`int` (oraz `long` w Python 2)	Liczba całkowita o dowolnej wartości	`42`
`float`	Liczba zmiennoprzecinkowa	`3.1415927`
`complex`	Liczba zespolona	`3+2.7j`
`bool`	Prawda lub fałsz	`True` `False`
`type(None)`	Nic (odpowiednik `null`)	`None`

ZMIENNE

- Dwa podstawowe typy liczbowe to int oraz float (całkowity i rzeczywisty)
- Nazwa float bierze się ze sposobu zapisu liczby w pamięci komputera za pomocą reprezentacji zmiennoprzecinkowej
- Napis (str lub unicode) inicjowany jest za pomocą pojedynczego lub podwójnego cudzysłowu
- Znak % wewnątrz napisu oznacza, że następujący bezpośrednio po nim łańcuch znaków stanowi format wydruku zmiennej znajdującej się po zamykającym cudzysłowie

```
In [1]: calkowita = 5

In [2]: rzeczywista = 4.6

In [3]: rzeczywista2 = float(7)

In [4]: print calkowita, rzeczywista, rzeczywista2
5 4.6 7.0

In [5]: print '%.20f' % rzeczywista
4.59999999999999964473

In [6]: print '%d + %1.2f =\n\t = %1.2f' % (calkowita, rzeczywista2, calkowita+rzeczywista2)
5 + 7.00 =
         = 12.00
```

ŁAŃCUCHY ZNAKÓW

```
In [1]: s1 = "Statystyka"

In [2]: s2 = "biomedyczna"

In [3]: s3 = s1 + " " + s2

In [4]: s4 = s3.upper()

In [5]: print s4
STATYSTYKA BIOMEDYCZNA

In [6]: s5 = s3.replace(s1, 'Inżynieria')

In [7]: print s5
Inżynieria biomedyczna

In [8]: s6 = '6'

In [9]: s6int = int(s6)

In [10]: print s6int
6

In [11]: s6float = float(s6)

In [12]: print s6float
6.0
```

- Operator „+” jest przeciążony dla argumentów typu str
- Inne przydatne operacje na łańcuchach: strip(), split(), join(). len()
- Pełna lista operacji dla danego obiektu, np. typu str, jest dostępna po wydaniu polecenia dir(s1)

LISTY

```
In [1]: a = [1, 3, 5, 7]

In [2]: b = range(5)

In [3]: print b
[0, 1, 2, 3, 4]

In [4]: a = range(1, 9, 2)

In [5]: print a
[1, 3, 5, 7]

In [6]: c = a + b

In [7]: print c
[1, 3, 5, 7, 0, 1, 2, 3, 4]

In [8]: d = a*2

In [9]: print d
[1, 3, 5, 7, 1, 3, 5, 7]

In [10]: e = [[0, 1], [2, 3], [4, 5]]

In [11]: e[1][1]
Out[11]: 3

In [12]: e[4][1]
Traceback (most recent call last):

  File "<ipython-input-12-a3b0dc142bf1>", line 1, in <module>
    e[4][1]

IndexError: list index out of range
```

- Indeksy elementów listy rozpoczynają się od 0
- Do elementów listy odwołujemy się za pomocą nawiasów [ ]
- W przypadku próby odwołania się przez indeks wykraczający poza zakres listy otrzymujemy wyjątek
- Listy mogą zawierać wartości różnych typów, a także inne listy

INDEKSOWANIE LIST

```
In [1]: a = range(10)

In [2]: a[3:6]
Out[2]: [3, 4, 5]

In [3]: a[3:-4]
Out[3]: [3, 4, 5]

In [4]: a[-7:6]
Out[4]: [3, 4, 5]

In [5]: a[:4]
Out[5]: [0, 1, 2, 3]

In [6]: a[-3:]
Out[6]: [7, 8, 9]

In [7]: a[0:9:2]
Out[7]: [0, 2, 4, 6, 8]

In [8]: a[::2]
Out[8]: [0, 2, 4, 6, 8]
```

- Do indeksowania całych fragementów list używamy znaku :
- Zakres elementów wybranych z tablicy obejmuje indeksy od początkowego (włącznie) do końcowego (z wyłączeniem)
- Można także ustalić krok wyboru elementów listy
- Ponadto listy są indeksowane od końca za pomocą liczb ujemnych (-1, -2,...)

OPERACJE NA LISTACH

```
In [1]: a = range(0,10,2) # a = [0, 2, 4, 6, 8]

In [2]: a.append(24) # a = [0, 2, 4, 6, 8, 24]

In [3]: a.extend([8, 10]) # a = [0, 2, 4, 6, 8, 24, 8, 10]

In [4]: a.insert(2, 8) # a = [0, 2, 8, 4, 6, 8, 24, 8, 10]

In [5]: a.count(8)
Out[5]: 3

In [6]: a.index(8)
Out[6]: 2

In [7]: a.pop(2) # a = [0, 2, 4, 6, 8, 24, 8, 10]
Out[7]: 8

In [8]: a.remove(8) # a = [0, 2, 4, 6, 24, 8, 10]

In [9]: b = a

In [10]: b.sort() # b = [0, 2, 4, 6, 8, 10, 24]

In [11]: a
Out[11]: [0, 2, 4, 6, 8, 10, 24]

In [12]: a.reverse()

In [13]: b
Out[13]: [24, 10, 8, 6, 4, 2, 0]
```

Uwaga! b jest referencją do tego samego obiektu listy

- append(val) - dołącza wartość do listy
- extend([ ]) - rozszerza listę o inną listę
- insert(idx, val) - wstawia do listy wartość jako nowy element o indeksie idx
- pop(idx) - zdejmuje z listy wartość o indeksie idx
- count(val) - wyznacza liczbę wystąpień wartości val w liście
- remove(val) - usuwa z listy pierwszą napotkaną wartość val
- sort() - sortuje elementy listy w porządku rosnącym
- reverse() - szereguje elementy listy w porządku odwrotnym

OPERACJE WARUNKOWE

```
In [1]: a = range(7)

In [2]: if 4 in a:
   ...:     idx = a.index(4)
   ...:     a.pop(idx)
   ...: elif len(a)<8:
   ...:     a.append(8)
   ...: else:
   ...:     print 'You never reach this statement.'
   ...:

In [3]: print a
[0, 1, 2, 3, 5, 6]

In [4]: a[0] == 0
Out[4]: True

In [5]: a[0] >= 0
Out[5]: True

In [6]: a[0] > 0
Out[6]: False
```

- Warunkowe wykonanie kodu odbywa się za pomocą klauzuli **if / elif / else**.
- W Pythonie zasięg klauzuli warunkowej (lub pętli) definiuje się za pomocą wcięć wierszy.
- Wykonanie poleceń następujących po **if** zależy od tego, czy zdanie warunkowe przyjmie wartość logiczną **True**, czy **False**.
- Wartości zerowe, **None** lub puste są traktowane jako **False**. Pozostałe są **True**.

PĘTLE FOR

```
In [1]: instruments = ['guitar', 'piano', 'violin']

In [2]: for item in instruments:
   ...:     item = item + 's'
   ...:     print item
   ...:
guitars
pianos
violins

In [3]: instruments
Out[3]: ['guitar', 'piano', 'violin']

In [4]: for i in range(len(instruments)):
   ...:     instrument = instruments[i]
   ...:     instrument = instrument + 's'
   ...:     instruments[i] = instrument
   ...:

In [5]: instruments
Out[5]: ['guitars', 'pianos', 'violins']
```

- Pętle **for** służą do seryjnego przetwarzania całych list; mówimy o *iterowaniu* po elementach listy
- Wewnątrz deklaracji pętli wprowadzamy tzw. zmienną iteracyjną
- Operacje wewnątrz pętli wykonujemy dla zmiennej iteracyjnej lub z jej użyciem (np. jako indeksu listy)

JESZCZE O PĘTLACH

```
In [4]: a = range(10)

In [5]: for i in range(10):
   ...:     print i
   ...:     if i > 6:
   ...:         break
   ...:     elif i > 3:
   ...:         continue
   ...:     a[i] = a[i] + 10
   ...:
0
1
2
3
4
5
6
7

In [6]: print a
[10, 11, 12, 13, 4, 5, 6, 7, 8, 9]
```

```
In [1]: a = range(10,20,2)

In [2]: aPlus2 = a + 2
Traceback (most recent call last):

  File "<ipython-input-2-78240e0808a5>", line 1, in <module>
    aPlus2 = a + 2

TypeError: can only concatenate list (not "int") to list


In [3]: aPlus2 = []

In [4]: for item in a:
   ...:     aPlus2.append(item + 2)
   ...:

In [5]: print aPlus2
[12, 14, 16, 18, 20]

In [6]: aPlus2 = [item+2 for item in a if item > 14]

In [7]: print aPlus2
[18, 20]
```

List comprehension

FUNKCJE

def – deklaracja funkcji

sygnatura funkcji

```
In [1]: def stringListToFloat(aStringList):
   ...:     aFloatList = []
   ...:     for i in range(len(aStringList)):
   ...:         aFloatList.append(float(aStringList[i]))
   ...:     return aFloatList
   ...:

In [2]: charList = ['1.0', '2.5', '0.5', '4.6']

In [3]: numList = stringListToFloat(charList)

In [4]: numList
Out[4]: [1.0, 2.5, 0.5, 4.6]
```

lista argumentów, kilka argumentów rozdzielonych przecinkami

return – wartość zwracana z funkcji

wywołanie funkcji przez sygnaturę

MODUŁY

W module definiujemy zmienne i funkcje, które następnie importujemy do środowiska wykonawczego.

typeconversion.py

```
1 charList = ['1.0', '2.5', '0.5', '4.6']
2
3 def stringListToFloat(aStringList):
4     aFloatList = []
5     for i in range(len(aStringList)):
6         aFloatList.append(float(aStringList[i]))
7     return aFloatList
8
```

```
In [1]: import typeconversion as tc

In [2]: print tc.charList
['1.0', '2.5', '0.5', '4.6']

In [3]: numList = tc.stringListToFloat(tc.charList)

In [4]: print numList
[1.0, 2.5, 0.5, 4.6]

In [5]: anotherCharList = ['3.0', '17.8', '2.3e5']

In [6]: anotherNumList = tc.stringListToFloat(anotherCharList)

In [7]: print anotherNumList
[3.0, 17.8, 230000.0]
```

Wzorzec **mapy** (programowanie *funkcyjne*) umożliwia pominięcie pętli **for**

```
1 charList = ['1.0', '2.5', '0.5', '4.6']
2
3 def stringListToFloat(aStringList):
4     aFloatList = map(float, aStringList)
5     return aFloatList
```

NUMPY

- Na podobnej zasadzie korzystamy ze „statystycznych” modułów Pythona: NumPy, SciPy i Matplotlib
- Obiekt **array** – n-wymiarowa tablica
- Bogaty zestaw funkcji: algebra liniowa, transformata Fouriera, liczby losowe
- Statystyka danych w tablicy (średnia, odchylenie standardowe, wartości minimalne i maksymalne)
- Tworzenie tablic za pomocą szeregu różnych funkcji (zeros, ones, linspace, logspace, mgrid)
- Szereg podstawowych funkcji matematycznych: trygonometryczne, logarytmiczne, wykładnicze, iloczyn skalarny i wektorowy, wartość bezwzględna, zaokrąglenia, itp.

SCIPY

- Bazuje na NumPy w zakresie struktur danych oraz podstawowych operacji matematycznych.
- Dostarcza algorytmów do
  - przetwarzania sygnałów
  - przetwarzania obrazów
  - analiza skupień
  - rachunku prawdopodobieństwa
  - optymalizacji funkcji, statystyki itp.

DSYTRYBUCJE

- Python udostępniany jest w wielorakich *dystrybucjach*
- Standardowo stanowi element składowy unix-owych systemów operacyjnych (OS X, Linux)
- Bardzo wygodne są pakiety oprogramowania zawierające interpreter Pythona oraz środowisko projektowe (edytor z wyróżnieniem składni i automatycznym uzupełnianiem kodu, monitor zmiennych, eksplorator plików)
- Popularne pakiety to m.in. Canopy oraz Anaconda (zawierająca środowisko Spyder2).

LIVE DEMO