



Разработка на софтуер

Лекция 3 –Въведение в Python

Милен Спасов

Python 3.1



- Инсталиране от: <https://www.python.org/downloads/release/python-3100/>
- Python 2.x vs. 3.x
 - Значително по-лесен, опростен и разбираем синтаксис
 - Използван от всички съвременни AI/ML технологии и системи
 - Голям брой библиотеки и ресурси
 - Голямо community и помощ при проблеми

Основни правила (1)

- Кодът отива в `.py` файлове
- Изпълнява се с `python example.py`
- Можем да пишем код интерактивно като пуснем `python` без аргументи
- Python е предсказуем, така че пробвайте, когато не сте сигурни


```
python
>>> 2 + 3
5
>>> a = 2
>>> b = a + 6
>>> a + b
10
>>> "hello" + ' ' + "world"
'hello world'
```
- В конзолата `help()` показва документацията на всяка функция, клас или тип


```
help(5)
help(some_function_name)
```
- Един ред код
 - Никога не завършва с `;`
 - Съдържа един израз
 - Всичко след `#` е коментар

Типове данни (1)

➤ Int

- Цели положителни и отрицателни числа
- Стандартни операции `+`, `-`, `*`, `/`, `%`, `**` (степенуване)
- Без максимален размер
- Може да пробваме `2 ** 4 ** 8`

➤ Float

- Числа с плаваща запетая
- Същите като целите числа по всички други показатели
- Поддържат специфичен запис `4.e7` ($= 4 * 10^7$)

➤ Complex

- Комплексни числа с реална и имагинерна част `2+3j`
- `type(2+3j)`

➤ Str

- Текстови низове с произволна дължина
- Единични или двойни кавички
- Използва се само Unicode
- Поддържат `\n`, `\t` и др.
- `"hello".upper()`
- `len("hey")`

Типове данни (2)

➤ Bool

- True или False
- Голямата буква има значение

➤ None

- Еквивалентът на null от други езици
- Когато функция не връща нищо, връща None
- Използваме го, за да кажем „нищо“ или „няма“

➤ За всички типове

- Всичко има тип
`type(5.5)`
`type("text")`
- Включително функциите
`type(len)`
- Всяка стойност е обект и има клас, включително функциите
- Всичко е обект, включително функциите и типовете
- Типът на всичко може да се провери с `type()`
- `type` е функция => `type` е обект => `type` си има тип
- `type(type(5.5)) = ?`

Променливи

- Можем да дадем стойност на име като така създаваме променлива

- Python е динамичен език и стойностите имат тип, но не и имената

```
>>> a = 4
>>> type(a)
<class 'int'>
```

- Стойности се присвояват по следния начин

```
>>> a = 10
>>> b = a
>>> a = 20
>>> print(b)
?
```

- Променливите могат да имат различен тип

```
>>> a = b = 200
>>> a = "test string"
>>> a
?
>>> b
?
```

Структури от данни (1)

- **list** (ordered, changeable, allows duplicates)
 - Списък = масив = array
 - Mutable и без фиксирана дължина
 - Бързо търсене по индекс, бавно по стойност
 - Може елементите да са от различен тип
 - Гарантиран ред

```
thislist = ["apple", "banana", "cherry"]
```

```
>>> my_list = []
>>> my_list.append("test")
>>> my_list.append(5)
>>> my_list[1] == 5
True
>>> len(my_list)
2
>>> del my_list[0]
>>> "test" in my_list
True
```

Структури от данни (2)

- **dict** (key-value pairs, unordered, changeable, does not allow duplicates)
 - Речник = hashtable = associative array
 - Асоциира ключ със стойност
 - Mutable и без фиксирана дължина
 - Редът не е гарантиран

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
```

```
>>> thisdict['year'] = 1965
```

```
>>> thisdict.get('owner')
```


Структури от данни (3)

➤ **tuple** (ordered, unchangeable, allows duplicates)

- tuple = кортеж
- Може да се ползва когато функцията връща повече от един резултат
- Immutable
- Гарантиран ред

```
thistuple = (9.24, 8.33, 11.22)
```

```
>>> thistuple[1]
```

```
8.33
```

```
>>> thistuple[2] = 21 / 4
```

```
?
```

Структури от данни (4)

- **set** (unordered, unchangeable, does not allow duplicates)
 - Множество = списък без повтарящи се елементи
 - Редът не е гарантиран
 - Не поддържа индексирание
 - Може да се проверява за принадлежност

```
thisset = {2, 5, 7, 9}
```

```
>>> thisset
```

```
(2, 5, 7, 9)
```

```
>>> thisset.add(5)
```

```
>>> thisset
```

```
(2, 5, 7, 9)
```

```
>>> thisset.remove(5)
```

```
>>> thisset
```

```
(2, 7, 9)
```

```
>>> myset = [1, 2, 3, 4, 5]
```

```
>>> s = set(myset)
```

```
>>> s
```

```
(1, 2, 3, 4, 5)
```

```
>>> 5 in s
```

```
True
```

Mutable vs. Immutable

```
>>> a = 5
>>> a += 2
>>> a
7
```

- Този код не променя стойността на 5, а насочва “a” към друга стойност
- Immutable са числата, низовете, `tuple`-ите, `True`, `False`, `None`
- Всичко останало е mutable
- Като ключ за `dict` или елемент на `set` могат да се ползват само immutable стойности

Контролни структури

➤ If... elif... else

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

- Няма изненади в това как работи
- Не слагайте скоби около условията
- **and, or, not** вместо **&&, ||, !**
- Идентацията е важна

С булеви променливи:

```
a = True
if a:
    print("a is True")
```

Идентация

- Няма къдрави скоби
- Всеки блок код (тяло на if, функция и т.н.) се определя с идентацията му спрямо обгръщания го блок
- Всеки блок започва с двуточие на предишния ред
- Блокът свършва, когато се върнете на предишна идентация
- 4 празни места = нов блок
- Точно 4 и без табове
- Заместете таб с 4 празни места в редактора си

"PYTHON INDENTATION"

(CODE THAT WORKS)

```
n = [3, 5, 7]

def double_list(x):

    for i in range(0, len(x))
        x[i] = x[i] * 2
    return x

print double_list(n)
```

(CODE THAT FAILS)

```
n = [3, 5, 7]

def double_list(x):

    for i in range(0, len(x))
        x[i] = x[i] * 2
    return x

print double_list(n)
```



[HTTPS://TAPAS.IO/SERIES/GRUMPY-CODES](https://tapas.io/series/grumpy-codes)



CARDBOARDVOICE

Цикли

➤ while

```
while a > 5:
    a -= 1
    print(a)
```

➤ for

```
primes = [3, 5, 7, 11]
for n in primes:
    print(n ** 2)
```

- For е като **foreach** в други езици
- Обхожда колекции от данни
- Няма инициализация, стъпка и проверка

➤ for като в C или C++

```
for i in range(0, 20):
    # do something
```

- **break** и **continue** работят както в другите езици

Switch/case

- Няма обичайния switch/case от други езици
- Python 3 поддържа pattern matching, който може да се използва като мощен switch/case
- Без pattern matching също може да се постигне тази функционалност:

```
>>> def week(i):
    switcher={
    0:'Sunday',
    1:'Monday',
    2:'Tuesday',
    3:'Wednesday',
    4:'Thursday',
    5:'Friday',
    6:'Saturday'
    }
    return switcher.get(i,"Invalid day of week")
```

Thank You

Maake Asante Shukria Dhanyavadagalu
Vinaka Kōszönöm Kiitos Maana Dankon
감사합니다 Dankscheen Манна Dankon
Dank Je Mauruuru Biyan
Blagodaram Ngiyabonga Dziekuje Chokrane
Juspaxar Arigato Diolch i Chi
நன்றி Bedankt Terima Kasih Matondo
Ua Tsaug Rau Koj D'akujem धन्यवाद Tack
Grazas câm ơn bạn Grazi Mochchakkeram
Děkuji Nirringrazzjak Tingki
Suksama Rahmat Gratias Tibi
Misaotra Matur Nuwun Obrigado
谢谢 Xbala Welalin Di Ou Mèsi
Danke Merci Kia Ora Kop Khun Khap
Salamat Go Raibh Maith Agat Tuke
Najis Tuke Eskerrik Asko