Типы данных в Python

1. Числовые

int

In [ ]:
```
x = 5
print(x)
```

In [ ]:
```
print(type(x))
```

In [ ]:
```
a = 4 + 5
b = 4 * 5
c = 5 / 4

print(a, b, c)
```

In [ ]:
```
print(-5 / 4)
```

In [ ]:
```
print(-(5 / 4))
```

In [ ]:
```
print(8/-2)
```

long

In [ ]:
```
x1 = 1000000
print(type(x1))
```

In [ ]:
```
x = 5 * x1 * x1 * x1 * x1 + 1
print(x)
print(type(x))
```

float

In [ ]:
```
y = 5.7
print(y)
print(type(y))
```

In [ ]:
```
a = 4.2 + 5.1
b = 4.2 * 5.1
c = 5.0 / 4.0

print(a, b, c)
```

In [ ]:
```
a = 5
b = 4
print(float(a) / float(b))
```

In [ ]:
```
print(5.0 / 4)
print(5 / 4.0)
```

In [ ]:
```
print(float(a) / b)
```

In [ ]:
```
0.7-0.5
```

bool

In [ ]:
```
a = True
b = False

print(a)
print(type(a))

print(b)
print(type(b))

#xeq = true
```

In [ ]:
```
print(a + b)
print(a + a)
print(b + b)
```

In [ ]:
```
print(int(a), int(b))
```

In [ ]:
```
print(True and False)
print(True and True)
print(False and False)
```

2. None

In [ ]:
```
z = None
print(z)
print(type(z))
```

In [ ]:
```
print(int(z))
```

3. Строковые

str

In [ ]:
```
x = "abc"
print(x)
print(type(x))
```

In [11]:
```
a = 'Ivan'
b = "Ivanov"
s = a + " " + b
print(s)
```

```
Ivan Ivanov
```

In [7]:
```
print(a.upper())
print(a.lower())
```

```
IVAN
ivan
```

In [8]:
```
print(len(a))
```

```
4
```

In [9]:
```
print(bool(a))
print(bool(""))
```

```
True
False
```

In [12]:
```
print(int(a))
```

```
---------------------------------------------------------------------
-------
ValueError                                Traceback (most recent cal
l last)
<ipython-input-12-ea96a59d6d6b> in <module>()
----> 1 print(int(a))

ValueError: invalid literal for int() with base 10: 'Ivan'
```

In [ ]:
```
print(a)
print(a[0])
print(a[1])
print(a[0:3])
```

In [6]:
```
print(a[0:4:2])
print(a[::-1])  # переворачивает строку или список
```

```
Ia
navI
```

In [ ]:
```
firstname, lastname = 'Иван Иванов'.split(' ')  # разбивает строку и с
print(firstname)
print(lastname)
```

unicode

In [ ]:
```
# outdated section
x = u"abc"
print x
print type(x)
```

In [3]:
```
x = u'Элеонора Михайловна'
print x, type(x)
y = x.encode('utf-8')
print y, type(y)
z = y.decode('utf-8')
print z, type(z)
q = y.decode('cp1251')
print q, type(q)
```

```
Элеонора Михайловна <type 'unicode'>
Элеонора Михайловна <type 'str'>
Элеонора Михайловна <type 'unicode'>
РР»РµPsPSPsCЂР° РњРёС…Р°Р№Р»PsPIPSP° <type 'unicode'>
```

In [ ]:
```
print y[1:]
print len(y), type(y)
print len(x), type(x)
```

In [ ]:
```
y = u'Иван Иванович'.encode('utf-8')
print y.decode('utf-8')
```

In [ ]:
```
splitted_line = "Ivanov Ivan Ivanovich".split(' ')
print splitted_line
```

In [ ]:
```
print type(splitted_line)
```

3. Массивы

list

In [ ]:
```
# Всем привет! Мы как массивы!
saled_goods_count = [33450, 34010, 33990, 33200]
print(saled_goods_count)
print(type(saled_goods_count))
```

In [ ]:
```
#print()"---".join(saled_goods_count)
lst = ['Ivanov', 'Petrov', 'Trump']
print(', '.join(lst))
#lst = ["Ivanov", "Ivan", "Ivanovich"]
#print("--".join(lst))
```

In [ ]:
```
# duck typing
mixedList = ['Ivan Ivanovich', 'Medium', 500000, 12, True]
print(mixedList)
```

In [ ]:
```
print(mixedList[0])
print(mixedList[1])
print(mixedList[3])
```

In [ ]:
```
print(mixedList[0:5])
```

In [ ]:
```
print(mixedList[:5])
```

In [ ]:
```
print(mixedList[1:])
```

In [ ]:
```
print(mixedList[2:5])
```

In [ ]:
```
print(mixedList[:-1])
```

In [ ]:
```
mixedList.append('One more element in list')
print(mixedList)
```

In [ ]:
```
del mixedList[-2]
```

In [ ]:
```
print(mixedList)
```

tuple

In [13]:
```
# Неизменяемый тип данных
mixedTuple = ()
mixedTuple = ('Ivan Ivanovich', 'Medium', 500000, 12, True)
print(type(mixedTuple))
```

```
<type 'tuple'>
```

In [ ]:
```
mixedTuple[2:5]
```

In [ ]:
```
#mixedTuple.append('one more element')

#mixedTuple = mixedTuple + ('Where am I?',)
mixedTuple

# Что с ним делать?
```

In [ ]:
```
# для работы с набором переменных неявно используются кортежи
a,b = 50, 100
print(a,b)
a,b = b,a  # обмен значений
print(a,b)
```

4. Множества и словари

set

In [ ]:
```
names = {'Ivan', 'Petr', 'Konstantin'}
print(type(names))
print(('Ivan' in names))
```

In [ ]:
```
print('Mikhail' in names)
names.add('Mikhail')
print(names)
```

In [ ]:
```
names.remove('Mikhail')
print(names)
```

In [ ]:
```
names.add(('Vladimir', 'Vladimirovich'))
print(names)
```

In [ ]:
```
a = range(10000)
b = range(10000)
b = set(b)
```

In [ ]:
```
print(a[:5])
print(a[-5:])
```

dict

In [ ]:
```
words_frequencies = dict()
words_frequencies['I'] = 1
words_frequencies['am'] = 1
words_frequencies['I'] += 1

print(words_frequencies)
```

In [ ]:
```
print(words_frequencies['I'])
```

In [ ]:
```
words_frequencies = {'I': 2, 'am': 1}
print(words_frequencies)
```

In [ ]:
```
yet_another_dict = {'abc': 3.4, 5: 7.8, u'123': None}
print(yet_another_dict)
```

In [ ]:
```
# Кортеж ключ !?!?!??!???"!Н1112

yet_another_dict[(1,2,5)] = [4, 5, 7]
print(yet_another_dict)
```

In [ ]:
```
# Использование списка как ключа!?
```

In [ ]:
```
# Использование set?
```

In [ ]:
```
e = dict()
s = frozenset((1,2,3))
e[s] = 3.76
e
```

Синтаксис Python

In [1]:
```
x = True
if x:
    print("OK")
else:
    print("NOT OK")
```

```
OK
```

In [ ]:
```
for i in range(10):
    print(i)
```

In [ ]:
```
for i in range(1,10):
   #print(i)
    print(i,)
# учим xrange()
```

In [ ]:
```
# конструктор списка (list comprehension)
w = [ x ** 2 for x in range(1,11) ]
print(w)
```

In [ ]:
```
# список элементов
w = [ x ** 2 for x in range(1,11) if x % 2 == 0]
```

In [ ]:
```
# конструктор генераторов
w = ( x ** 2 for x in range(1,11) if x % 2 == 0 )
print(type(w)) # не храним в памяти во время работы цикла
```

In [ ]:
```
w1=list(w)
print(w1)
```

In [ ]:
```
s = 0
while True: # Это условие всегда выполняется
    s += 1
    if (s % 2 == 0): # Через раз
        print("Continue") # вместо s будет выведено "continue"
        continue # и код ниже в таком случае не будет исполнен в этой
    print(s)
    if s > 10: # С помощью этой конструкции
        break # <<бесконечный цикл досрочно прерывается>>
```

In [ ]:
```
def sq(x):
    return x ** 2
```

In [ ]:
```
print(map(sq, range(10)))
```

In [ ]:
```
sq = lambda x: x**2
```

In [ ]:
```
print(map(lambda x: x**2, range(10)))
```

Чтение данных из файла

In [2]:
```
with open('sentenses.txt', 'r') as file:
    #print file.read()
    print(file.read().splitlines()) #[0]
```

```
---------------------------------------------------------------------
-------
IOError                                   Traceback (most recent cal
l last)
<ipython-input-2-9bdfb2c6f58b> in <module>()
----> 1 with open('sentenses.txt', 'r') as file:
      2     #print file.read()
      3     print(file.read().splitlines()) #[0]

IOError: [Errno 2] No such file or directory: 'sentenses.txt'
```

Pandas

In [ ]:
```
import pandas as pd
import numpy as np

print(pd.__version__)
print(np.__version__)
```

In [ ]:
```
d = {'col1': 123, 'col2': 234}
# df = DataFrame(data=d, index=index)
df2 = pd.DataFrame(np.random.randn(10, 5))
df3 = pd.DataFrame(np.random.randn(10, 5), columns=['a', 'b', 'c', 'd'
print(df3)
```

In [ ]:
```
import requests
url = 'https://api.hh.ru/vacancies/'
resp = requests.get(url + '18374197')
vac = resp.json()
vac['area']
```

In [ ]:
```
data = []
for i in range(18399084, 18399394):
    data.append(requests.get(url + str(i)).json())
df = pd.DataFrame(data)
```

In [ ]:
```
df.head()
```

In [ ]:
```
df.head().T
```

In [ ]:
```
df.experience
```

In [ ]:
```
df[df.salary.isnull()].head(10)
```

In [ ]:
```
print(df.shape)
print(df[df.salary.isnull()].shape)

# удаляем закрытые вакансии
df = df[df.billing_type.notnull()]
```

In [ ]:
```
del df['alternate_url'], df['response_url'], df['suitable_resumes_url'
```

In [ ]:
```
import math
def f(x):
    return x['name']
```

In [ ]:
```
df['billing_type'].apply(f).head(10)
```

In [ ]:
```
df['billing_type'].apply(lambda x: x['name']).head(10)
```

In [ ]:
```

```

In [ ]:
```

```

Упрощенная задача машинного обучения

Найти две строки, наиболее похожие на первую

- Читаем из файла
- Получаем все слова
- Убираем пустые
- Убираем дублирование, сохранив порядок слов в тексте. Сохраняем первое вхождение по правилу allWords[<индекс первого вхождения>]) = <слово>
- Заполняем матрицу по правилу $a_{i,j}$ = количество вхождений j-го слова из allWords в i-ю строку
- Находим косинусное расстояние между нулевой строкой матрицы (вхождения в первую строку строку текстового файла)
- Находим три максимально похожие строки на первую (минимальные косинусные расстояния)

In [14]:
```
# косинусное расстояние между строками
import scipy
from scipy.spatial import distance
scipy.spatial.distance.cosine(matrix[0], matrix[1])
```

```
---------------------------------------------------------------------
-------
NameError                                 Traceback (most recent cal
l last)
<ipython-input-14-8d17c452f75e> in <module>()
      2 import scipy
      3 from scipy.spatial import distance
----> 4 scipy.spatial.distance.cosine(matrix[0], matrix[1])

NameError: name 'matrix' is not defined
```

$$1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2}.$$

In [ ]:
```
import numpy as np
#matrix = (23,255)
#print matrix
len(matrix)
```

Где еще можно познакомиться с Python

- https://www.coursera.org/courses?query=Python (https://www.coursera.org/courses?query=Python)
- https://www.codeacademy.com (https://www.codeacademy.com)
- http://www.pythontutor.ru (http://www.pythontutor.ru)
- http://www.learnpythonthehardway.org (http://www.learnpythonthehardway.org)
- http://snakify.org (http://snakify.org)
- https://www.checkio.org (https://www.checkio.org)