# Python

После установки конды, создания и входа в окружение можно пользоваться питоном либо в Jupyter Notebook либо в PyCharm, либо из командной строки.

## List

Статья про то как устроен список в питоне: http://www.laurentluce.com/posts/python-list-implementation/

In [ ]:
```
# Можно хранить объекты любых типов
[1, 2, 3, 'hello', False, 'we', ['qw', 1]]
```

In [ ]:
```
arr = [0, 1, 2, 3, 4]
```

### Основные операции со списком:

In [ ]:
```
arr.append(5) # Добавление элемента в конец списка
arr
```

In [ ]:
```
arr.extend([6, 7, 8]) # Расширение списка другим списком
arr
```

In [ ]:
```
# Выделение(срез) подсписка.
# Обратите внимание на отрицательный индекс - отсчет с конца списка
arr[3:-2:2]
```

#### Присваивание

Присваивание - копируется ссылка на список но сам список не копируется

In [ ]:
```
arr1 = arr
arr1[0] = -1
arr
```

### Копирование

После выполнения среза возвращается новый список с нужными элементами - происходит копирование. То есть скопировать список можно так:

In [ ]:
```
arr_copy = arr[:]
arr_copy[1] = -1
arr
```

arr.copy() приведет к тому же самому результату

In [ ]:
```
arr_copy = arr.copy()
```

Но копируются только элементы списка, и если мы сделаем элементом списка другой список, то он не будет копироваться полностью, а скопируется только указатель на него

In [ ]:
```
arr.append([1,2,3])
arr
```

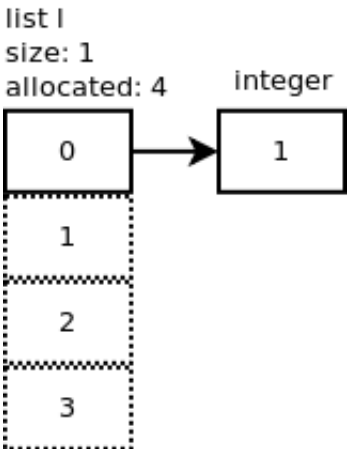
In [ ]:
```
arr_copy = arr[:]
arr_copy[-1][0] = -1
arr
```

### Глубокое копирование

Чтобы избежать этой проблемы можно использовать библиотеку copy

In [ ]:
```
import copy
arr_deepcopy = copy.deepcopy(arr)
arr_deepcopy[-1][1] = -1
arr
```

## О том как устроен список




In [ ]:
```
import sys
```

In [ ]:
```
sys.getsizeof([])
```

In [ ]:
```
sys.getsizeof([1])
```

In [ ]:
```
def print_list_size(l):
    list_str = f"sizeof({l})"
    print(f"{list_str:<23} == {sys.getsizeof(l)} bytes")

l = []
l.append(1)
print_list_size(l) # 88 - 56 == 32 == 8 * 4 - после append выделяется памя
l.append(2)
print_list_size(l) # Новая память не выделяется так как пока что хватает вы
l.append(3)
print_list_size(l)
l.append(4)
print_list_size(l)
l.append(5)
print_list_size(l)
```

Больше операций со списками тут:

https://docs.python.org/3/tutorial/datastructures.html

## Dict - ассоциативный массив. Данные в виде пар (ключ, значение)

In [ ]:
```
d = dict() # как unordered_map в C++
d = {'ab': 1, 'srt': 2, 'qwe': 3, 'qdf': 4}
d
```

In [ ]:
```
d[[1,2,3]] = 'abc'
```

In [ ]:
```
d[(1, 2, 3)] = 'abc'
d
```

Важно помнить что порядок ключей в словаре может отличаться от того, в котором их туда добавляли! Начиная с python3.7 порядок сохраняется, но код не должен рассчитывать на то что элементы словаря или множества будут выдаваться в опредленном порядке. Если нужен порядок есть collections.OrderedDict

In [ ]:
```
d['ab'] = 'e'
d
```

In [ ]:
```
from collections import OrderedDict
x = OrderedDict([(1, 2), (3, 4)])
x[5] = 6
print(x.items())
```

In [ ]:
```
for key, value in d.items():
    print(f'key: {key}, value: {value}')
```

In [ ]:
```
d.keys()
```

In [ ]:
```
d.values()
```

In [ ]:
```
for key, value in zip(d.keys(), d.values()):
    print(f'key: {key}, value: {value}')
```

## Dict comprehensions

In [ ]:
```
{i: i * 2 for i in range(10)}
[i ** 2 for i in range(10)]
```

## Метод get() - Если ключ есть в словаре то возвращается значение

## соответствующее этому ключу, иначе второй аргумент

In [ ]:
```
d.get('ab', 2)
```

In [ ]:
```
d.get('abd', 2)
```

## defaultdict - Такой же словарь, только при обращении по новому ключу

## можно использовать то, что значением является заданный объект

In [ ]:
```
from collections import defaultdict
def_dict = defaultdict(list)
def_dict
```

In [ ]:
```
list()
```

In [ ]:
```
def_dict['a'].append(1)
def_dict
```

## Set

In [ ]:
```
s = set() # Как unoredered set в c++
s = {1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5}
s
```

In [ ]:
```
4 in s
```

In [ ]:
```
s.add(8)
s
```

In [ ]:
```
a = {x for x in 'abracadabra' if x not in 'abc'}
a
```

In [ ]:
```

```

## Строки

In [ ]:
```
'asdfljkasdf'
```

### Метод join строит строку соединяя элементы списка через разделитель

In [ ]:
```
s = ' '.join([f'word{i+1}' for i in range(10)])
s
```

In [ ]:
```
s[1:10] # строка это список строк из одного символа
```

### Метод split(sep) делить строку по указанному разделителю

In [ ]:
```
print(s.split(' '))
```

# Загрузка текстов

In [191…
```
!ls filimdb_evaluation/FILIMDB/
```

```
dev-b.labels          dev.texts             train.labels
dev-b.texts           test-b.texts          train.texts
dev.labels            test.texts            train_unlabeled.texts
```

In [192…
```
from collections import defaultdict
with open(f'filimdb_evaluation/FILIMDB/train_unlabeled.texts', 'r', encodir
    text = f.read()

words = [
    word
    for sentence in text.split('\n')
    for word in sentence.split(' ')
]
```

In [193…
```
len(words)
```

Out[193…
```
11722415
```

In [194…
```
words[:10]
```

Out[194…
```
['As',
 'was',
 'said',
 'above,',
 'this',
 'is',
 'basically',
 '90210',
 'on',
 'Acid.']
```

# Удаление стоп слов

In [ ]:
```
import random
# Пусть для примера стоп слова это 100 случайных слов из корпуса
stopwords = set(random.sample(words, 100))
len(stopwords)
```

## Поиск элемента в list vs. set

In [ ]:
```
array = list(range(10 ** 7))
```

In [ ]:
```
%%timeit
150000 in array
```

In [ ]:
```
_set = set(range(10 ** 7))
```

In [ ]:
```
%%timeit
150000 in _set
```

## Loops vs. list comprehensions

Используйте генерацию списков вместо циклов - она быстрее и занимает меньше строк кода

In [ ]:
```
%%timeit
filtered_words = []
for word in words:
    if word not in stopwords:
        filtered_words.append(word)
```

In [ ]:
```
%%timeit
filtered_words = [
    word
    for word in words
    if word not in stopwords
]
```

# Пример создания словаря с частотами

In [ ]:
```
%%timeit
vocab = defaultdict(int)
for w in words:
    vocab[w] += 1
```

In [ ]:
```
%%timeit
vocab = dict()
for w in words:
    try:
        vocab[w] += 1
    except:
        vocab[w] = 1
```

In [ ]:
```
%%timeit
vocab = dict()
for w in words:
    vocab[w] = vocab.get(w, 0) + 1
```

In [ ]:
```
%%timeit
vocab = dict()
for w in words:
    if w in vocab:
        vocab[w] += 1
    else:
        vocab[w] = 1
```

In [ ]:

In [ ]:

# Полезные сслыки

Несколько различных кратких введений в python:

1. https://docs.python.org/3/tutorial/
2. https://www.learnpython.org/
3. https://en.wikibooks.org/wiki/A_Beginner's_Python_Tutorial
4. Статья об особенностях jupyter notebook: https://habr.com/ru/company/wunderfund/blog/316826/

Существуют также интерактивные курсы. К примеру, https://www.codecademy.com - курс learn python