

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ  
ЦИФРОВОГО РАЗВИТИЯ**

**Отчет по лабораторной работе №1 по дисциплине:  
технологии распознавания образов**

Выполнил:

студент группы ПИЖ-б-о-21-1

Прокопов Дмитрий Владиславович

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2023 г.

## ВЫПОЛНЕНИЕ

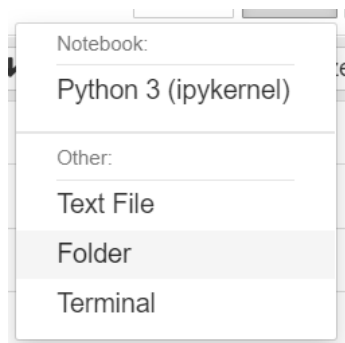


Рис. 1.1 – “Folder” создание новой папки

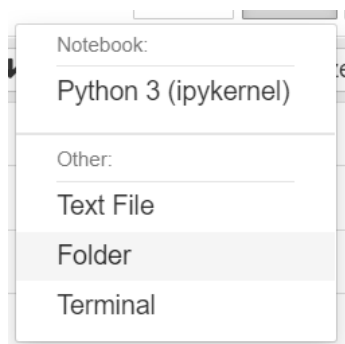


Рис. 1.2 – “Python 3(ipykernel)” создание нового ноутбука

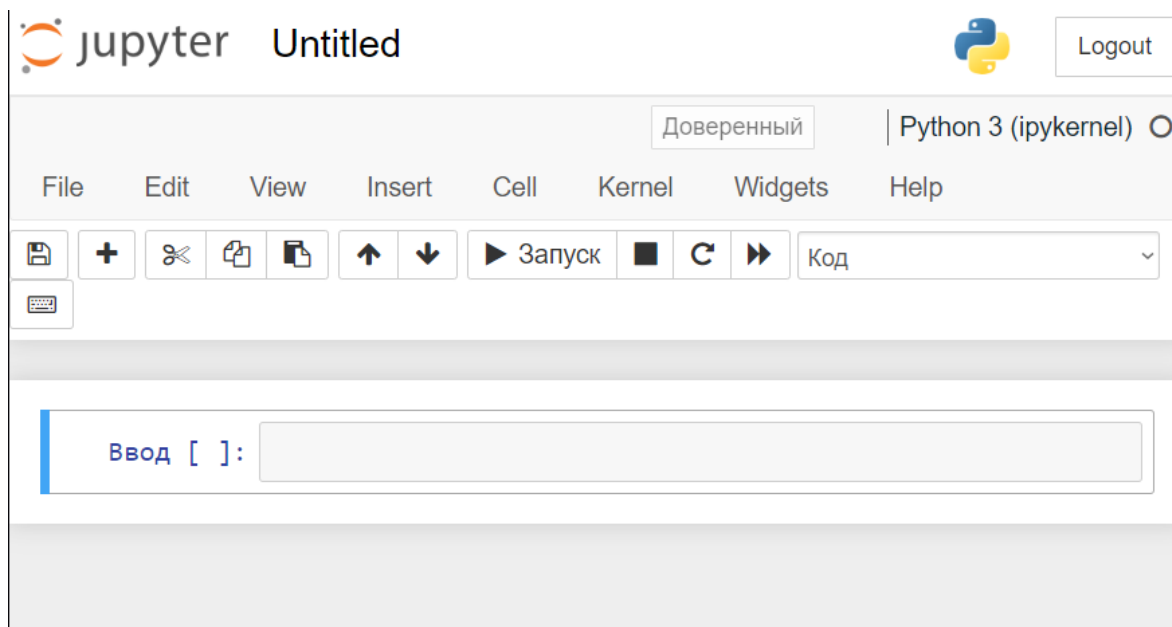


Рис. 1.3 – Созданный ноутбук

```
Ввод [1]: 3+2
Out[1]: 5
```

Рис. 1.4 – пример работы в ноутбуке №1

```
Ввод [4]: a=5  
b=3  
print(a+b)
```

8

```
Ввод [5]: n=7  
for i in range(n):  
    print(i+10)
```

10  
11  
12  
13  
14  
15  
16

Рис. 1.5 – пример работы в ноутбуке №2

```
Ввод [ ]: from matplotlib import pylab as plt  
%matplotlib inline
```

```
Ввод [9]: x=[i for i in range(50)]  
y=[i**2 for i in range(50)]  
plt.plot(x,y)
```

Out[9]: [matplotlib.lines.Line2D at 0x1a4e0960ee0]

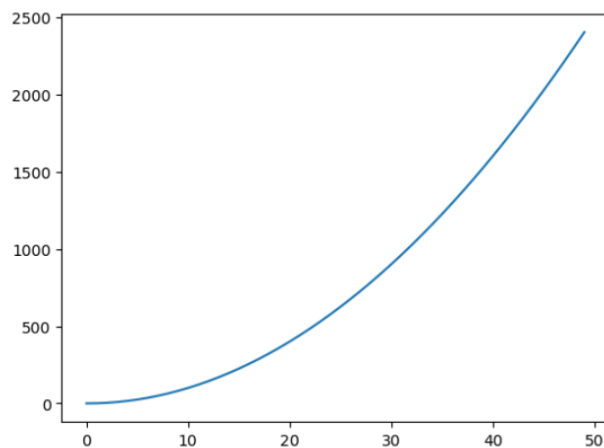


Рис. 1.6 – вывод изображения в ноутбуке

```
Ввод [10]: %lsmagic
```

```
Out[10]: Available line magics:
%alias %alias_magic %autoawait %autocall %automagic %autosave %bookmark
%cd %clear %cls %colors %conda %config %connect_info %copy %ddir %debu
g %dhist %dirs %doctest_mode %echo %ed %edit %env %gui %hist %history
%killbgscripts %ldir %less %load %load_ext %loadpy %logoff %logon %logs
tart %logstate %logstop %ls %lsmagic %macro %magic %matplotlib %mkdir
%more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinf2
%pip %popd %pprint %precision %prun %psearch %psource %pushd %pwd %pyc
at %pylab %qtconsole %quickref %recall %rehashx %reload_ext %ren %rep
%rerun %reset %reset_selective %rmdir %run %save %sc %set_env %store %
sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos
%xdel %xmode
```

```
Available cell magics:
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javasc
ript %%js %%latex %%markdown %%perl %%prun %%pypy %%python %%python2 %
python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%wr
itefile
```

Automagic is ON, % prefix IS NOT needed for line magics.

Рис. 1.7 – вывод списка магических команд

```
Ввод [11]: %env TEST = 5
```

```
env: TEST=5
```

```
Ввод [14]: %%time
import time
for i in range(50):
    time.sleep(0.1)
```

```
Wall time: 5.36 s
```

Рис. 1.8 – пример выполнения магических команд

## Сравнение алгоритмов пузырьковой и быстрой сортировки

### Быстрая сортировка

QuickSort является существенно улучшенным вариантом алгоритма сортировки с помощью прямого обмена, известном в том числе своей низкой эффективностью. Принципиальное отличие состоит в том, что в первую очередь производятся перестановки на наибольшем возможном расстоянии и после каждого прохода элементы делятся на две независимые группы.

### Сортировка пузырьком

Алгоритм сортировка пузырьком - простейший, но эффективен он лишь для небольших массивов.

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется перестановка элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

```
import random
from random import randint
import time
from matplotlib import pylab as plt
%matplotlib inline
```

```
def quicksort(nums):
    if len(nums) <= 1:
        return nums
    else:
        q = random.choice(nums)

        less_nums = []
        more_nums = []
        equal_nums = []

        for i in nums:
            if i < q:
                less_nums.append(i)

            elif i > q:
                more_nums.append(i)

            else:
                equal_nums.append(i)

        return quicksort(less_nums) + equal_nums + quicksort(more_nums)
```

```
def bubel(arr, N):
    i = 0
    while i < N - 1:
        j = 0
        while j < N - 1 - i:
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
            j += 1
        i += 1
    return(arr)
```

```
x = []
bub = []
qic = []
nums = 10

while nums < 20000:
    x.append(nums)
    lst = [randint(0, 100000) for i in range(nums)]
    nums = nums * 2

    start = time.time()
    quicksort(lst)
    qic.append(time.time() - start)

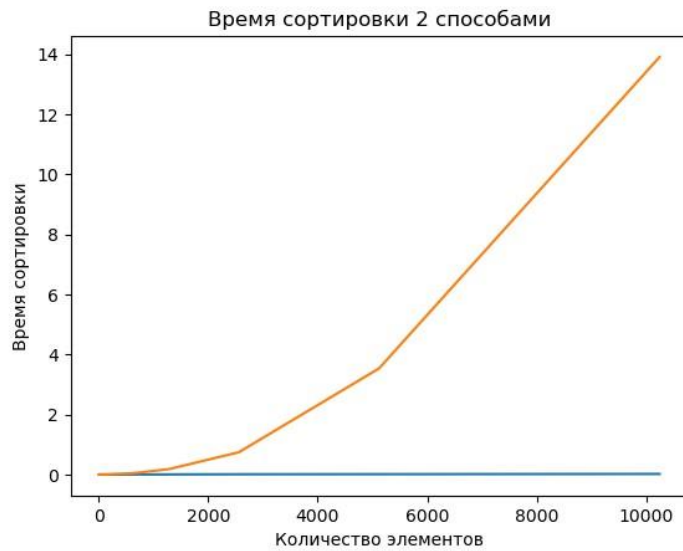
    nums = 10

while nums < 20000:
    lst = [randint(0, 100000) for i in range(nums)]

    start = time.time()
    bubel(lst, nums)
    bub.append(time.time() - start)
    nums = nums * 2
```

Ниже приведен график времени работы сортировки пузырьком и быстрой сортировки

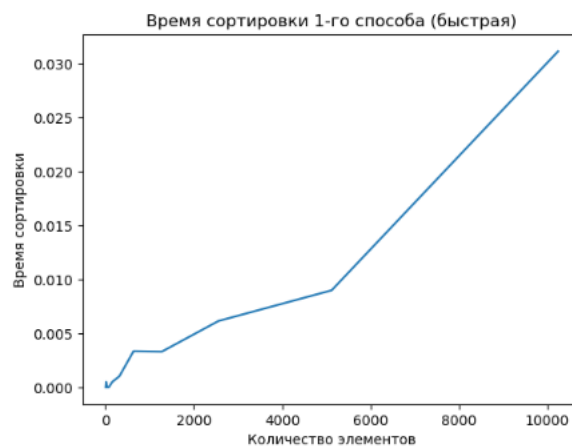
```
plt.title("Время сортировки 2 способами")
plt.xlabel("Количество элементов")
plt.ylabel("Время сортировки")
plt.plot(x, qic, x, bub)
plt.show()
```



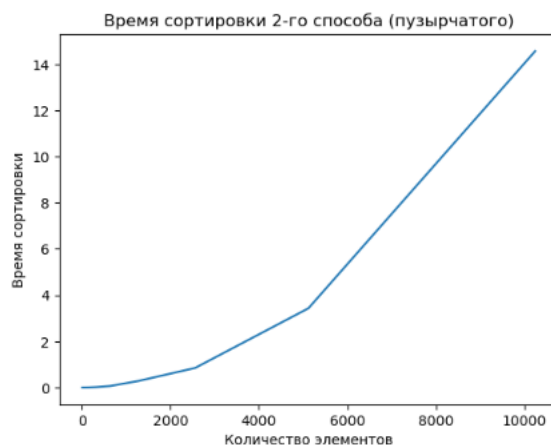
## Вывод

Исходя из графика, можно сказать, что время сортировки массива быстрой сортировкой почти не изменяется с увеличением количества элементов, в то время как время выполнения пузырьковой сортировки значительно возрастает с увеличением количества элементов

```
plt.title("Время сортировки 1-го способа (быстрая)")
plt.xlabel("Количество элементов")
plt.ylabel("Время сортировки")
plt.plot(x, qic)
plt.show()
```



```
plt.title("Время сортировки 2-го способа (пузырчатого)")
plt.xlabel("Количество элементов")
plt.ylabel("Время сортировки")
plt.plot(x, bub)
plt.show()
```



```
num=list(map(int, input("Введите число: ")))
```

Введите число: 444444

```
if sum(num[:3])==sum(num[3:]):  
    print('счастливый')  
else:  
    print('обычный')
```

счастливый

```
num=list(map(int, input("Введите число: ")))
```

Введите число: 123456

```
if sum(num[:3])==sum(num[3:]):  
    print('счастливый')  
else:  
    print('обычный')
```

обычный

## Пароль

```
pas='password6335'
```

```
k = pas.lower().count("andrei")  
if k == 0:  
    uq = set(pas)  
    lower_flag = False  
    upper_flag = False  
    digit_flag = False  
    for char in uq:  
        if char.islower():  
            lower_flag = True  
        if char.isupper():  
            upper_flag = True  
        if char.isdigit():  
            digit_flag = True  
    if len(uq) >= 4 and lower_flag and upper_flag and digit_flag:  
        print('strong')  
    else:  
        print('weak')  
else:  
    print('weak')
```

weak

## Числа Фибоначчи

```
f1 = f2 = 1  
amount = int(input())
```

6

```
print(f1, f2, end=' ')  
for i in range(2, amount):  
    f1, f2 = f2, f1 + f2  
    print(f2, end=' ')
```

1 1 2 3 5 8

## Исследование

```

import csv
from matplotlib import pylab as plt
import statistics
%matplotlib inline

def ShortFloat(a, digits=0):
    return f"{a:.{digits}f}"

year = []
count_people = []
perc_people = []
temp_y = 0
cp = 0
pp = 0

with open('late-night-preferential-runway-use-1.csv', 'r', newline='', encoding='utf-8') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    next(reader)
    for k in reader:
        if k[1] != "N/A":
            if int(k[1]) != temp_y:
                year.append(int(k[1]))
                temp_y = int(k[1])
                count_people.append(cp)
                pp = pp / 12
                pp = float(ShortFloat(pp, 2))
                perc_people.append(pp)
                cp = 0
                pp = 0
            else:
                cp += int(k[3])
                r = k[4].find('%')
                temp_pp = ""
                temp_pp = str(k[4])
                temp_pp = temp_pp[:r]
                pp += int(temp_pp)

aver_cpeop = 0
for peoples in count_people:
    aver_cpeop += peoples
aver_cpeop = aver_cpeop / len(count_people)
print(f'Среднее кол-во/год уехавших за границу людей: {aver_cpeop}')

aver_ppeop = 0
for percents in perc_people:
    aver_ppeop += percents
aver_ppeop = aver_ppeop / len(perc_people)
print(f'Средний % людей/год уехавших за границу: {aver_ppeop}')

stocp = statistics.stdev(count_people)
stoppp = statistics.stdev(perc_people)
print(f'Стандартное отклонение в кол-ве людей: {stocp}')
print(f'Стандартное отклонение в % людей: {stoppp}')

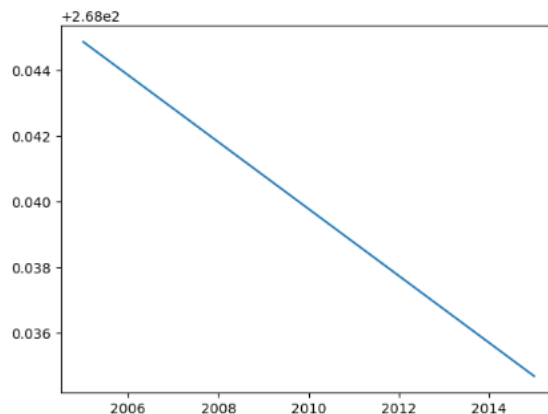
s_year = sum(year)
s_pp = sum(perc_people)
sqy = 0
yearpp = 0
for i, item in enumerate(year):
    sqy += s_year ** 2
    yearpp += year[i] * perc_people[i]
a = (s_pp * s_year - yearpp) / (s_year ** 2 - sqy)
b = (yearpp - sqy * a) / s_year
print(f"Уравнение линейной зависимости: y = {a}x + {b}")

y = []
for elem in year:
    y.append(a * elem + b)

plt.plot(year, y)

```

Среднее кол-во/год уехавших за границу людей: 870.1818181818181  
 Средний % людей/год уехавших за границу: 22.507272727272724  
 Стандартное отклонение в кол-ве людей: 452.00349958419974  
 Стандартное отклонение в % людей: 11.503561266763514  
 Уравнение линейной зависимости: y = -0.001017899035311498x + 270.0857476707372  
 [«matplotlib.lines.Line2D at 0x1c11cасеса0»]



## Ответы на контрольные вопросы:

1. Как осуществляется запуск Jupyter notebook?

Запуск Jupyter notebook осуществляется через приложения «Jupyter Notebook (anaconda3)»



## 2. Какие существуют типы ячеек в Jupyter notebook?

Бывают трех типов: Code, Markdown и Raw

## 3. Как осуществляется работа с ячейками в Jupyter notebook?

В ячейку кода вводится код, после чего он запускается. Вывод появляется в ячейках markdown

## 4. Что такое "магические" команды Jupyter notebook? Какие "магические" команды Вы знаете?

Магией в Jupyter Notebook называются дополнительные команды, выполняемые в рамках оболочки, которые облегчают процесс разработки и расширяют возможности. Например: %ismagic, %env, %%time, %timeit

## 5. Самостоятельно изучите работу с Jupyter notebook и IDE PyCharm и Visual Studio Code.

Для PyCharm Professional нужно создать новый проект. В этом проектесоздайте новый файл ipynb, выбрав File > New > Jupyter notebook. Это должно открыть новый файл записной книжки.

Для VS Code надо открыть Command Palette с помощью сочетания клавиш CTRL+SHIFT+P(Windows) или Command+SHIFT+P(macOS) и запустить команду «Python: Create Blank New Jupyter Notebook». Если у вас уже есть файл Jupyter notebook, это так же просто, как просто открыть этот файл в VS Code. Он автоматически откроется с новым редактором Jupyter.