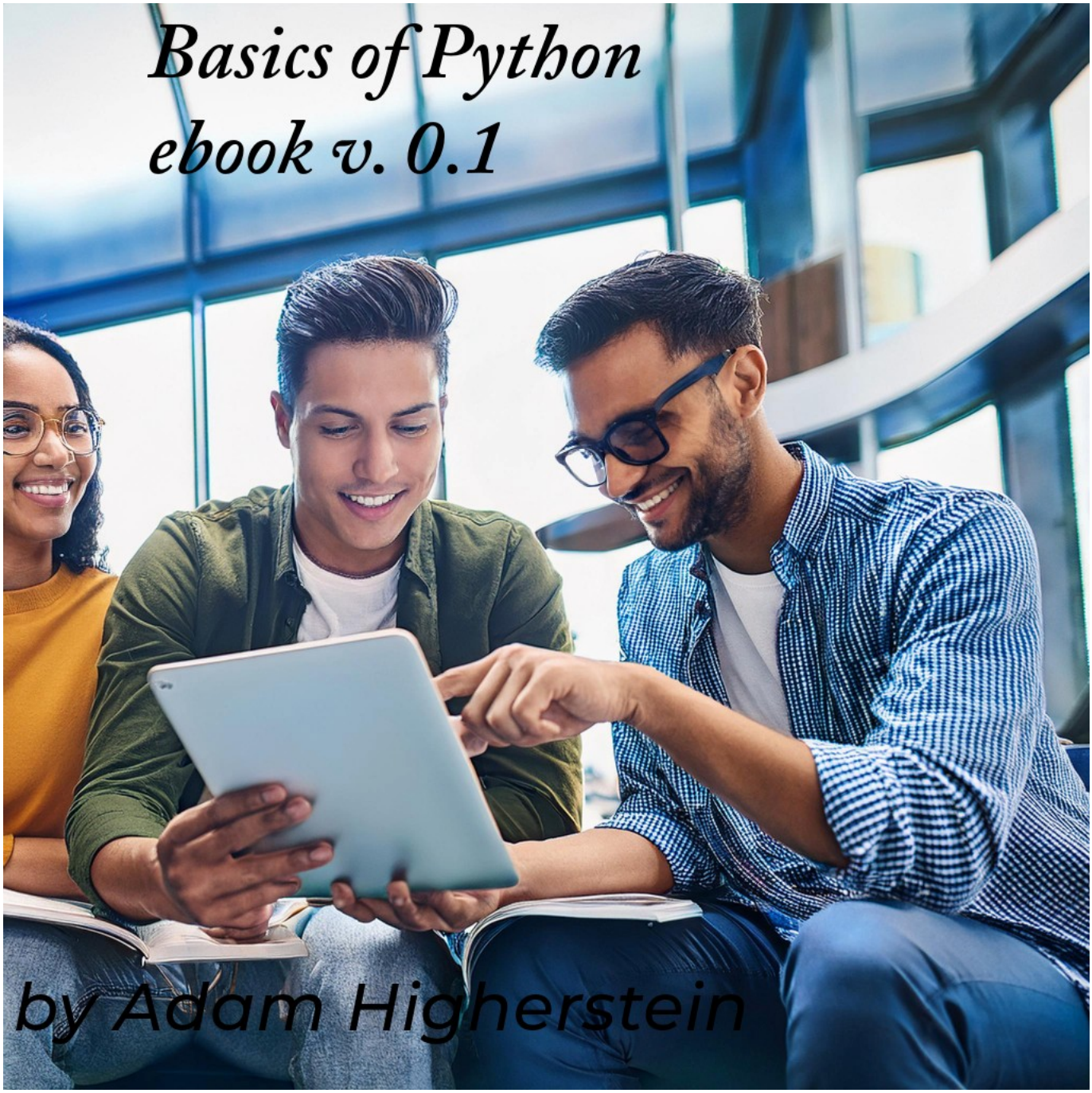

Basics of Python
ebook v. 0.1
by Adam Higherstein

Basics in programming

with

Python

Free eBook

by Adam

Table of Contents

Files....1
Reading a file....1
Removing white spaces....2
Read line by line....3
Lines to a list....4
Searching from the file....5
Writing to a file....6
Appending to a file....7
Opening modes....8
Shortly about manipulating csv files....8
Loading csv file....8
Saving csv file....8
Reading and writing....9
Exceptions....9
Example: reading a file and getting error number....11
Introduction to exceptions....11
What is an Exception....11
Examples of exceptional situations....12
Benefits....12
Classic example: division by zero....12
Here we catch different exceptions....12
Exception class hierarcy is shown here....13
More about exceptions....14
Plotting charts....15
Matplotlib....15
Examples....15
More info to chart....17
Example: bar chart....18
Example: Best fit curve....19
Final words....21

Files

Reading a file

Read the whole contents

file1.py - C:/kk/PYTHON2020/file1.py (3.7.5)

File Edit Format Run Options Window Help

```
# file reading
with open('persons1.txt') as file_object
 contents = file_object.read()
 print(contents)
```

Ln: 6 Col: 0

File is here

persons1.txt - Notepad

File Edit Format View Help

```
Mary
2020
Dick
1222
John
3456
Ted
9898
Kim
0101
```

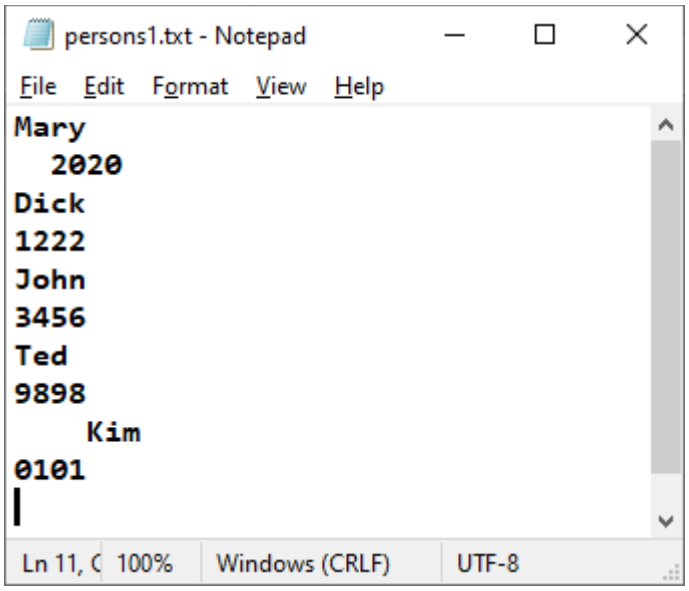
Ln 10, C 100% Windows (CRLF) UTF-8

Run result

```
Mary
2020
Dick
1222
John
3456
Ted
9898
Kim
0101
```

Removing white spaces

If there can be extra whitespaces in the textfile, you can use strip() function.




```
print(contents.strip())
```

Result

```
Mary
2020
Dick
1222
John
3456
Ted
9898
Kim
0101
```

Note
Now we had the file in the same directory as code file. It file exists in other place you have to type the file path.

Read line by line

file2.py - C:/kk/PYTHON2020/file2.py (3.7.5)

File Edit Format Run Options Window Help

```
filename = "persons1.txt"
with open(filename) as file_object:
  for line in file_object:
    print(line)
```

Ln: 6 Col: 0

```
Mary

  2020

Dick

1222

John

3456

Ted

9898
```

Also here you can use strip().

Lines to a list

file3.py - C:/kk/PYTHON2020/file3.py (3.7.5)

File Edit Format Run Options Window Help

```
filename = "persons1.txt"
with open(filename) as file_object:
  lines = file_object.readlines()

#print the 1. element
print(lines[0])
```

Ln: 8 Col: 0

Result

```
Mary
```

Searching from the file

Searching for a telephone number

Code

file4.py - C:/kk/PYTHON2020/file4.py (3.7.5)

File Edit Format Run Options Window Help

```
filename = "persons1.txt"
with open(filename) as file_object:
  lines = file_object.readlines()

strippedLines = []
for line in lines:
  line = line.strip()
  strippedLines.append(line)

for line in strippedLines:
  print(line)

person = "Ted"
# tell the phone number
index = -1
for line in strippedLines:
  index += 1
  if line == person:
    print(strippedLines[index + 1])
```

Ln: 20 Col: 35

First we remove extra whitespaces.

Then we go trough the list. We know that telephone number is below the person's name. When person's name is found, we print the next line (next element of the list).

For "Ted" we get

```
9898
```

In reading we opened the file using mode "r".

Writing to a file

Now we open the file using mode "w". If file does not exist, it is created.

Example 1

file5.py - C:/kk/PYTHON2020/file5.py (3.7.5)

File Edit Format Run Options Window Help

```
filename = 'courses2020.txt'

with open(filename, 'w') as file_object:
  file_object.write("Python")
```

Ln: 4 Col: 29

Result

courses2020.txt

courses2020.txt - Notepad

File Edit Format View Help

```
Python
```

Writing several lines

file6.py - C:/kk/PYTHON2020/file6.py (3.7.5)

File Edit Format Run Options Window Help

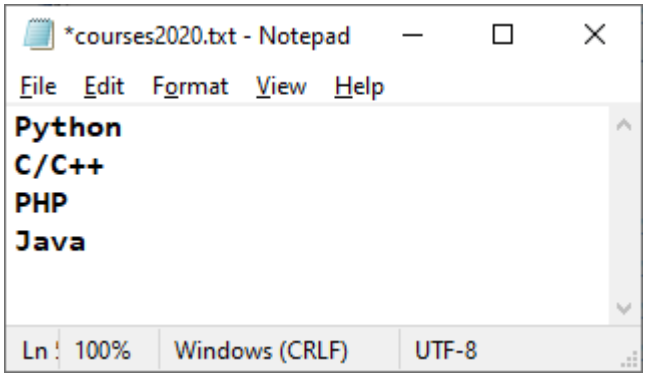
```
filename = 'courses2020.txt'

with open(filename, 'w') as file_object:
  file_object.write("Python")
  file_object.write("\n")
  file_object.write("C/C++")
  file_object.write("\n")
  file_object.write("PHP")
  file_object.write("\n")
  file_object.write("Java")
```

Ln: 11 Col: 2

Note:

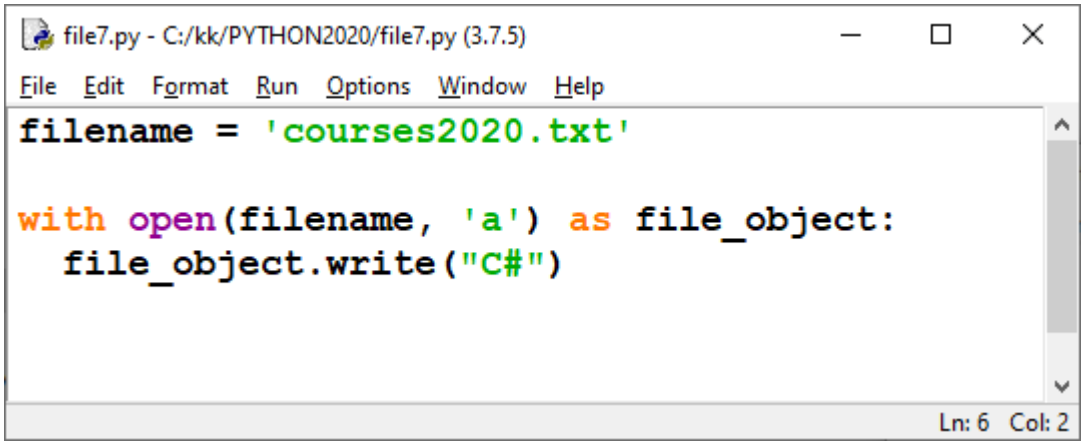
To get now line breaks, extra "\n" was printed, too.

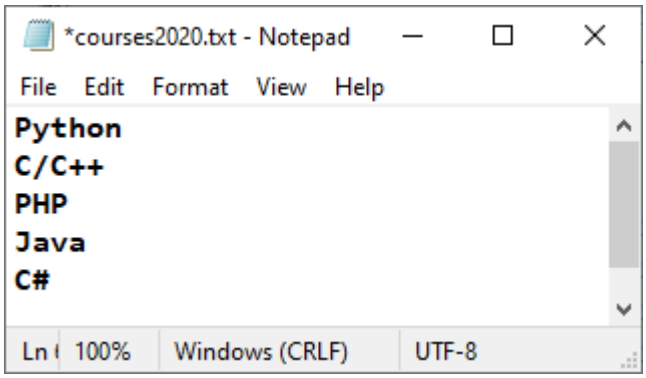
Result



Appending to a file

Mode is now "a".





Opening modes

Here are all modes listed

Text mode	Binary mode	Description
rt	rb	read
wt	wb	write
at	ab	append
r+t	r+b	read and update
w+t	w+b	write and update

Shortly about manipulating csv files

Just a couple of examples here: csv file are really important because we often use them when reading statistivcal data from different sources.

Loading csv file

```
from csv import reader
from csv import writer
import csv

# Loading a CSV file
filename ='combats_short.csv'
file1 = open(filename, "r")
lines = reader(file1)
print(filename)
for row in lines:
  print (row)
file1.close()
```

Saving csv file

```
# Saving a CSV file
file2  = open('testdata.csv', "w", newline='\n')
writer = writer(file2)
```

```
writer.writerows("Kauko")


file2.close()
```

Reading and writing

```
# read and print new file

filename ='testdata.csv'

file1 = open(filename, "r")

lines = reader(file1)

print(filename)

for row in lines:

  print (row)


file1.close()
```

Exceptions

Now it is a cood chance to discuss exceptions!

When using files several exceptional things may happen:
e.g. file not found

So, we start with an example: we try to open a file for reading that does not exist.

Code is here

except1.py - C:/kk/PYTHON2020/except1.py (3.7.5)

File Edit Format Run Options Window Help

```
try:
  stream = open("course2020.txt", "rt")
  # processing goes here
  stream.close()
except Exception as exc:
  print("Cannot open the file:", exc)
```

Ln: 8 Col: 0

Note that file name is not correct,

We run now this code.

```
Cannot open the file: [Errno 2] No such file or directory:
'course2020.txt'
```

SO, we get a message that file was not found. User has now to check the file name or its location and run the code again.

Our program did not crash thanks to catching that exception!

We put the main code to try block.

Then we executed the code.

Because the file was not found, an exception was raised.

That exception was caught by the catch block.

Then it ws handled: a message was printed.

Note this line

except Exception as exc:

We were ble to catch all possible exceptions, because class Exception is the base class of all exceptions.

If you want to get more detailed information about exceptions, you have to catch objects of child classes...

Example: reading a file and getting error number

except2.py - C:/kk/PYTHON2020/except2.py (3.7.5)

File Edit Format Run Options Window Help

```
try:
    cnt = 0
    s = open('coursess.txt', "rt")
    ch = s.read(1)
    while ch != '':
        print(ch, end='')
        cnt += 1
        ch = s.read(1)
    s.close()
    print("\n\nCharacters in file:", cnt)
except IOError as e:
    print("I/O error occurred: ", strerr(e.errno))
```

Ln: 13 Col: 0

We get

```
Traceback (most recent call last):
  File "C:/kk/PYTHON2020/except2.py", line 3, in <module>
    s = open('coursess.txt', "rt")
FileNotFoundError: [Errno 2] No such file or directory: 'coursess.txt'
```

Now the error did not occur during the reading process, but it is always possible.

Introduction to exceptions

What is an Exception

Exceptional event - typically an error that occurs during runtime

Cause normal program flow to be disrupted

Examples of exceptional situations

Divide by zero errors

Accessing the elements of an array beyond its range Invalid input

Hard disk crash

Opening a non-existent file Heap memory exhausted

Benefits

Separating Error-Handling code from “regular” business logic code

Propagating errors up the call stack Grouping and differentiating error types

Classic example: division by zero

except3.py - C:/kk/PYTHON2020/except3.py (3.7.5)

File Edit Format Run Options Window Help

```
firstNumber = int(input("Enter the first number: "))
secondNumber = int(input("Enter the second number: "))

try:
    print(firstNumber / secondNumber)
except:
    print("This operation cannot be done.")

print("THE END.")
```

Ln: 10 Col: 0

Trial

```
Enter the first number: 3
Enter the second number: 0
This operation cannot be done.
THE END.
```

Here we catch different exceptions

except4.py - C:/kk/PYTHON2020/except4.py (3.7.5)

File Edit Format Run Options Window Help

```
try:
    x = int(input("Enter a number: "))
    y = 1 / x
    print(y)
except ZeroDivisionError:
    print("You cannot divide by zero, sorry.")
except ValueError:
    print("You must enter an integer value.")
except:
    print("Oh dear, something went wrong...")

print("THE END.")
```

Ln: 14 Col: 0

Try it!

Python 3 defines 63 built-in exceptions,

Exception class hierarcy is shown here

BaseException

↑

Exception

↑

ArithmeticError

↑

ZeroDivisionError

More about exceptions

The try-except block can be extended in one more way - by adding a part headed by the finally keyword (it must be the last branch of the code designed to handle exceptions).

Note: these two variants (else and finally) aren't dependent in any way, and they can coexist or occur independently.

The finally block is always executed (it finalizes the try-except block execution, hence its name), no matter what happened earlier, even when raising an exception, no matter whether this has been handled or not.

Example

except5.py - C:/kk/PYTHON2020/except5.py (3.7.5)

File Edit Format Run Options Window Help

```
def reciprocal(n):
    try:
        n = 1 / n
    except ZeroDivisionError:
        print("Division failed")
        n = None
    else:
        print("Everything went fine")
    finally:
        print("It's time to say goodbye")
        return n

print(reciprocal(2))
print(reciprocal(0))
```

Ln: 15 Col: 0

```
Everything went fine
It's time to say goodbye
0.5
Division failed
It's time to say goodbye
None
```

Study more exceptions when needed!

They are must!

You have to use them!

Plotting charts

Matplotlib

Check that you have matplotlib package. Install or upgrade it.

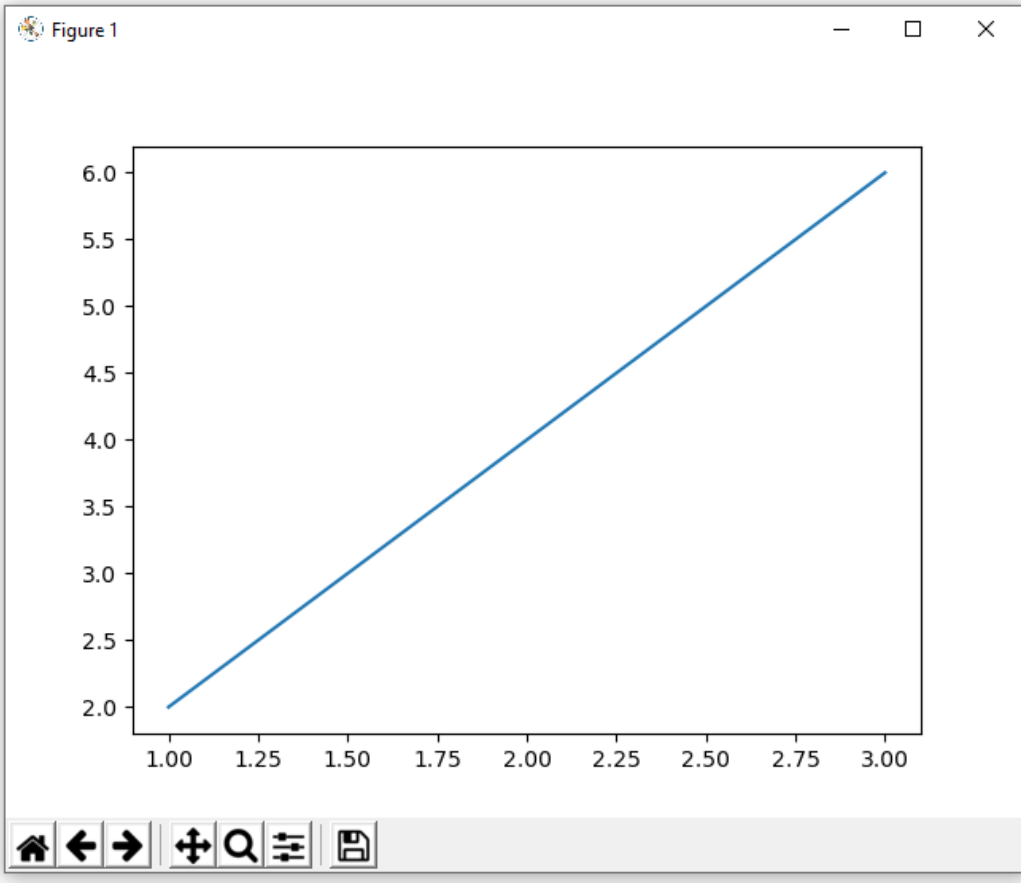
```
C:\Users\KaukoK>python -m pip install --upgrade pip
```

Examples

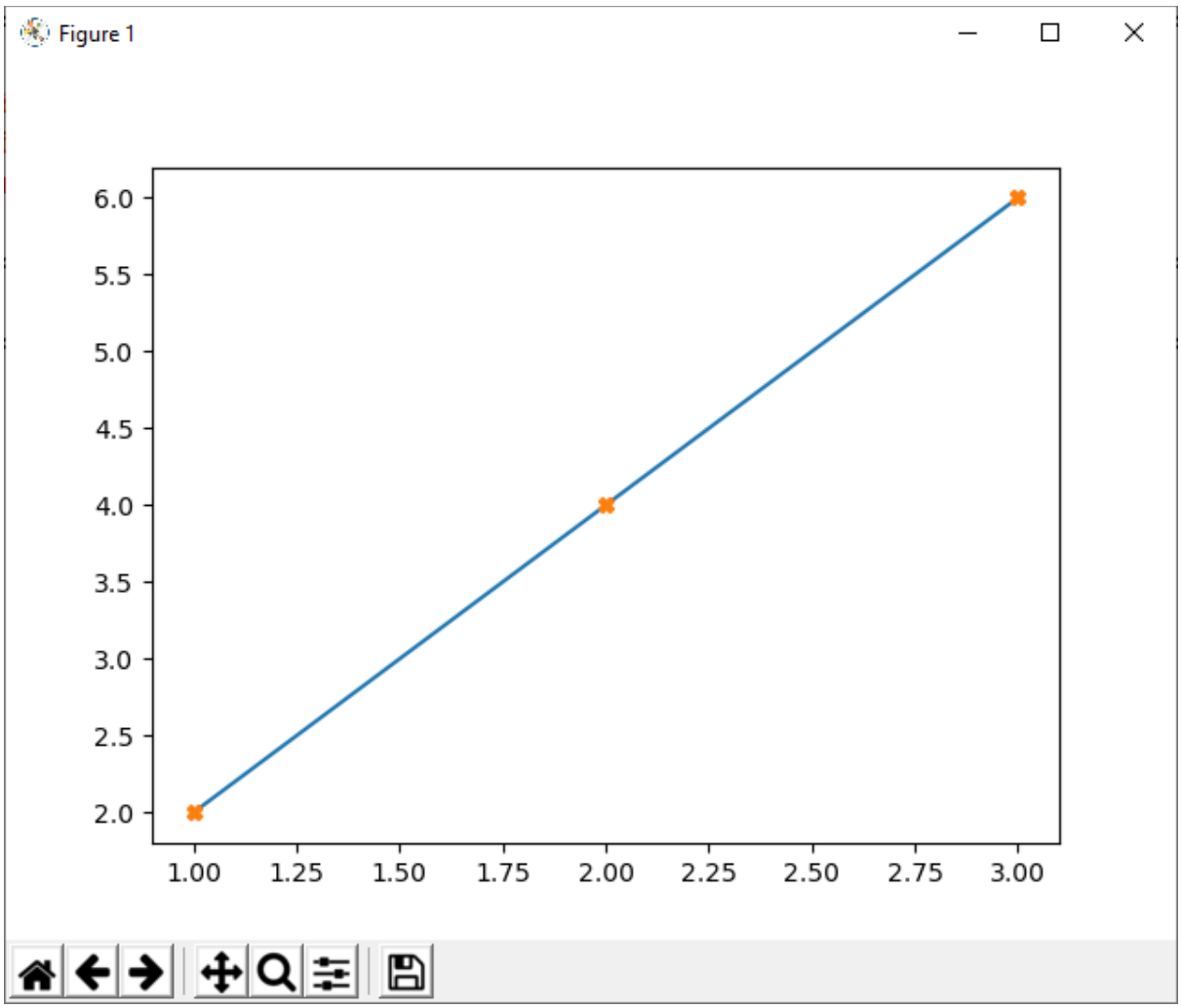
plot1.py - C:/kk/PYTHON2020/plot1.py (3.8.3)

File Edit Format Run Options Window Help

```
from pylab import plot, show

x_numbers = [1, 2, 3]
y_numbers = [2, 4, 6]

plot(x_numbers, y_numbers)

show()
```



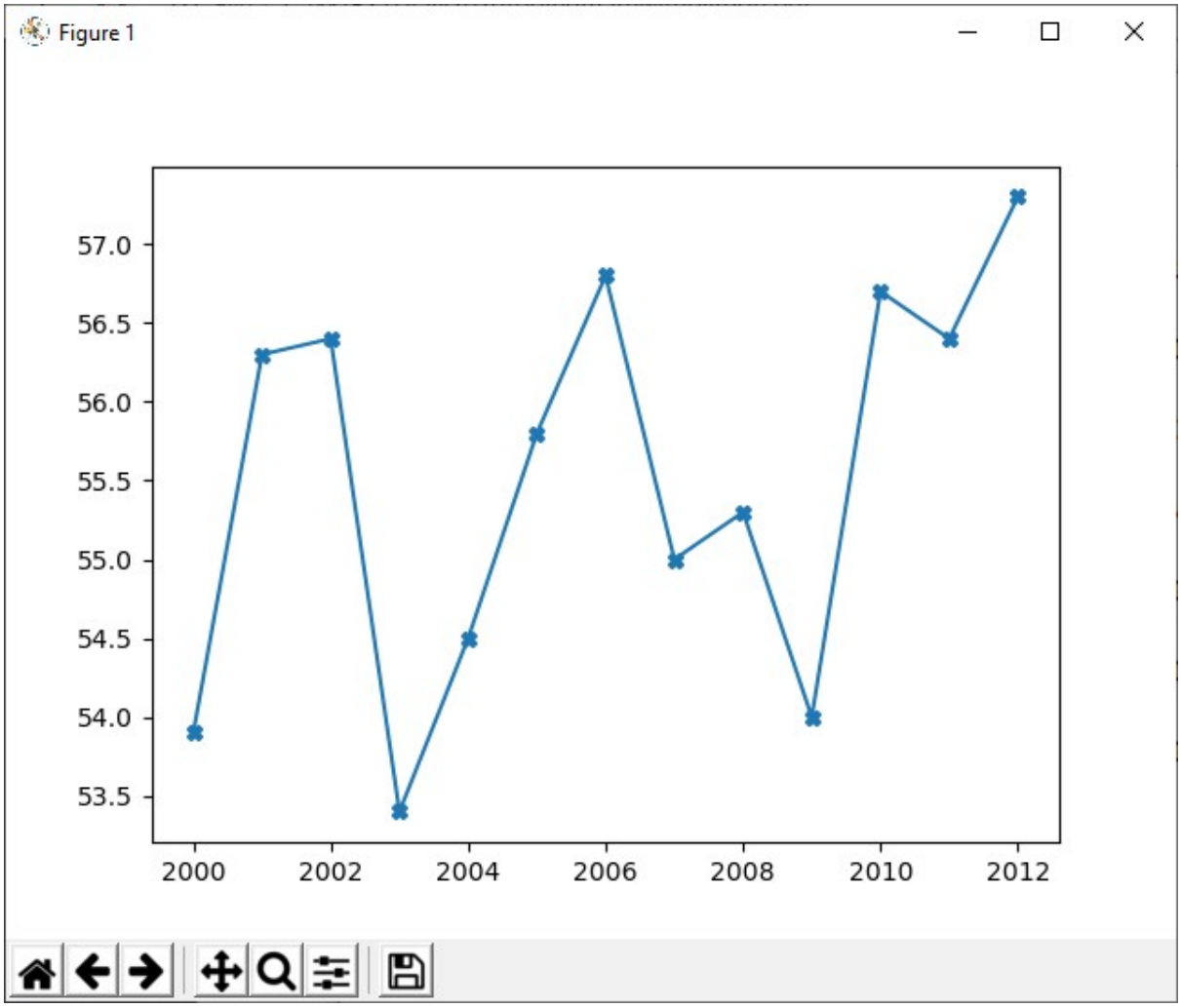
Marks with



Code is

plot(x_numbers, y_numbers, 'X')

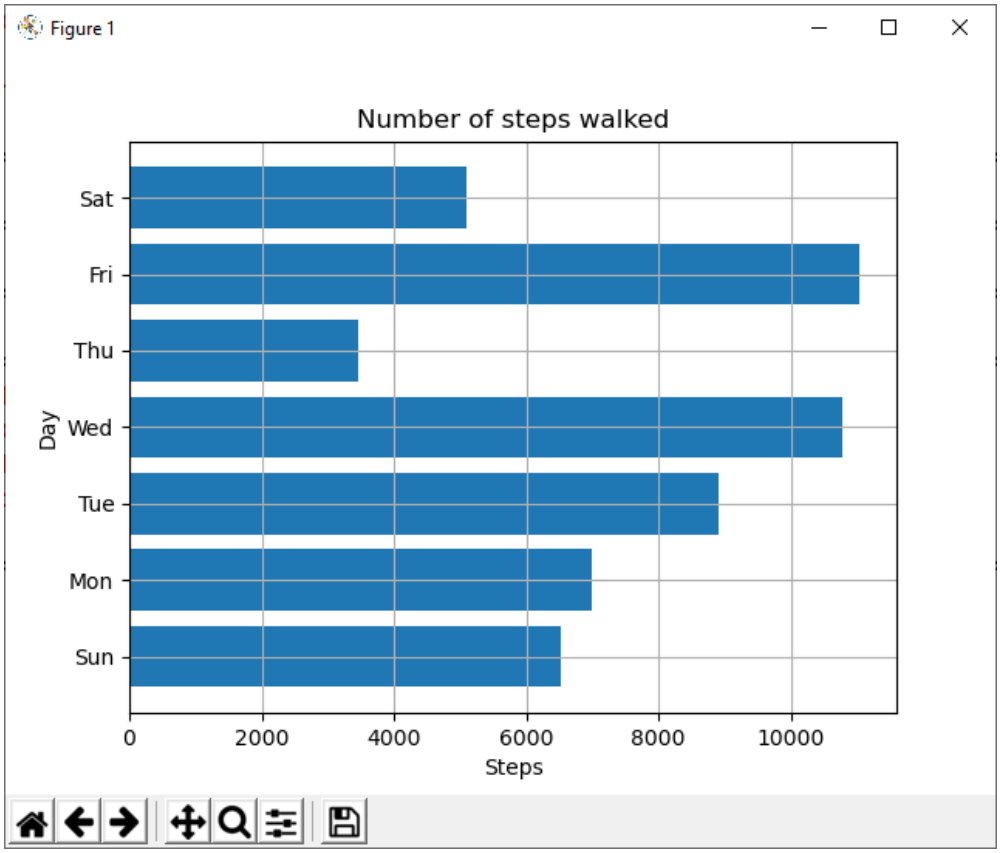
More info to chart



Code

```
temps = [53.9, 56.3, 56.4, 53.4, 54.5, 55.8, 56.8, 55.0, 55.3, 54.0, 56.7, 56.4, 57.3]
years = range(2000, 2013)
plot(years, temps, marker='X')

show()
```

Example: bar chart



Code

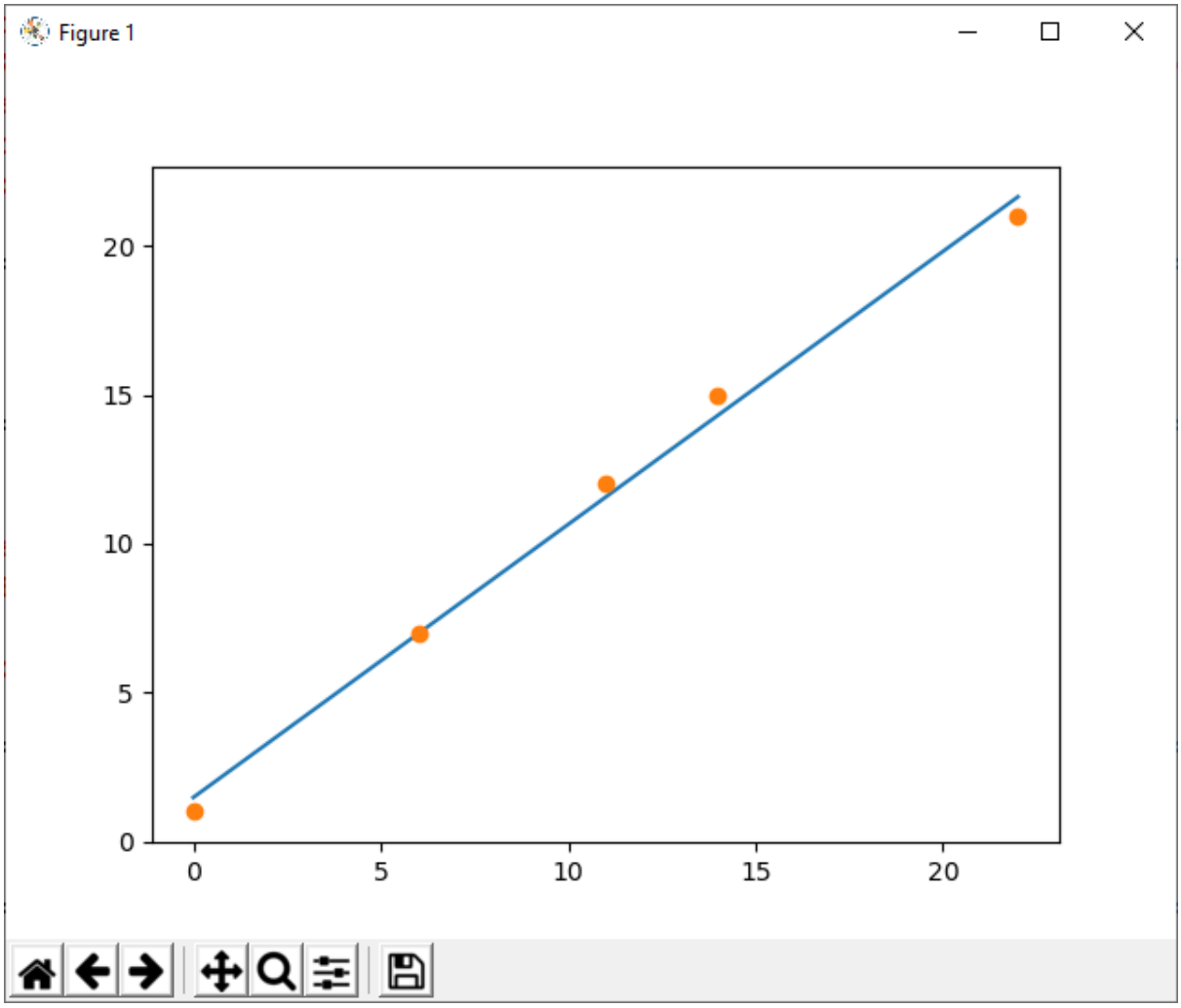
plot3.py - C:/kk/PYTHON2020/plot3.py (3.8.3)

File Edit Format Run Options Window Help

```
import matplotlib.pyplot as plt

# Number of steps I walked during the past week
steps = [6534, 7000, 8900, 10786, 3467, 11045, 5095]
 # Corresponding days
labels = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
data = steps
# Number of bars
num_bars = len(data)
 # This list is the point on the y-axis where each
 # Bar is centered. Here it will be [1, 2, 3...]
positions = range(1, num_bars+1)
plt.barh(positions, data, align='center')
 # Set the label of each bar
plt.yticks(positions, labels)
plt.xlabel('Steps')
plt.ylabel('Day')
plt.title('Number of steps walked')
# Turns on the grid which may assist in visual estimation
plt.grid()
plt.show()
```

Ln: 22 Col: 0

Example: Best fit curve



Code

```
import matplotlib.pyplot as plt

# best fit example

# sample points

X = [0, 6, 11, 14, 22]

Y = [1, 7, 12, 15, 21]

# solve for a and b

def best_fit(X, Y):

  xbar = sum(X)/len(X)

  ybar = sum(Y)/len(Y)

  n = len(X) # or len(Y)
```

```
    numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
    denum = sum([xi**2 for xi in X]) - n * xbar**2

    b = numer / denum
    a = ybar - b * xbar

    print('best fit line:\ny = {:.2f} + {:.2f}x'.format(a, b))

    return a, b

# solution
a, b = best_fit(X, Y)

# plotting in separate process
xbar = sum(X)/len(X)
ybar = sum(Y)/len(Y)
n = len(X) # or len(Y)

numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
denum = sum([xi**2 for xi in X]) - n * xbar**2

b = numer / denum
a = ybar - b * xbar

fitArray1 = [];
fitArray2 = [];
for s in range (5):
    fitArray1.append(X[s])
    fitArray2.append(a + b*X[s])

plt.plot(fitArray1, fitArray2)
```

```
plt.plot(X,Y,linestyle='none', marker='o')
plt.show()
```

Go on studying more!

Final words

This part covered basics of some features of programming with Python.

There are several other areas where Python is used.

New parts are coming and they are going discuss technologies like Data Analysis!!

Also GUI: how to create GUI using Python?

We discuss that later.

Feedback is welcome!