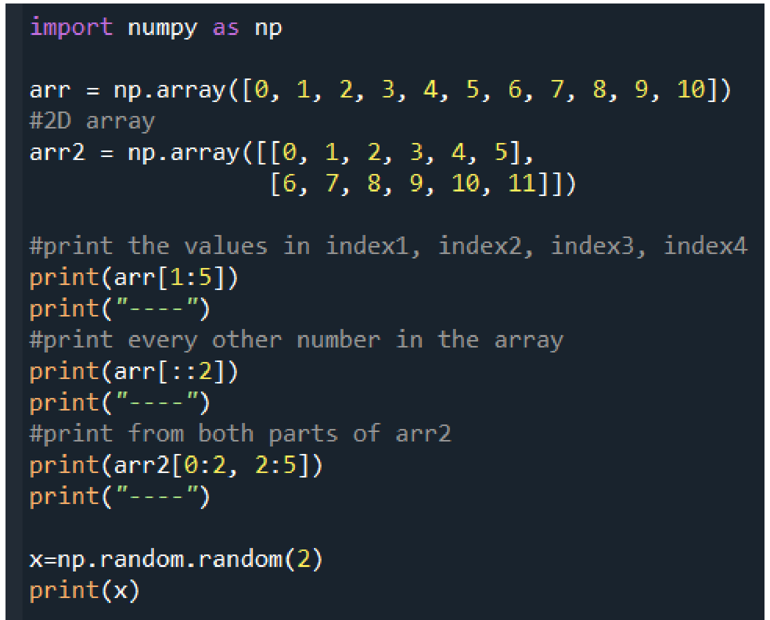
Formal Languages and Compilers

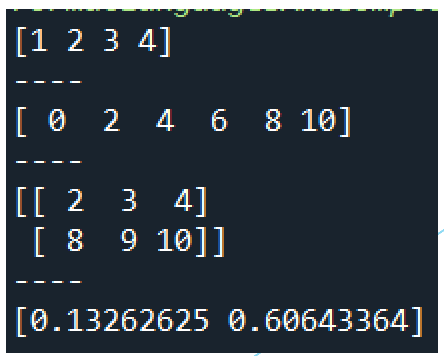
Lab7

numpy

- Open-source library, used for scientific computing and working with arrays (*pip install numpy*)
- Arrays need to be declared, can store data compactly and are great for numerical operations
- You can consider 2-D arrays as a table with rows and columns, with the dimension standing in for the row and the index for the column.
- You can iterate through arrays, split them, sort them and many more
- random() can be used to generate random populated arrays of different types
- Simple example

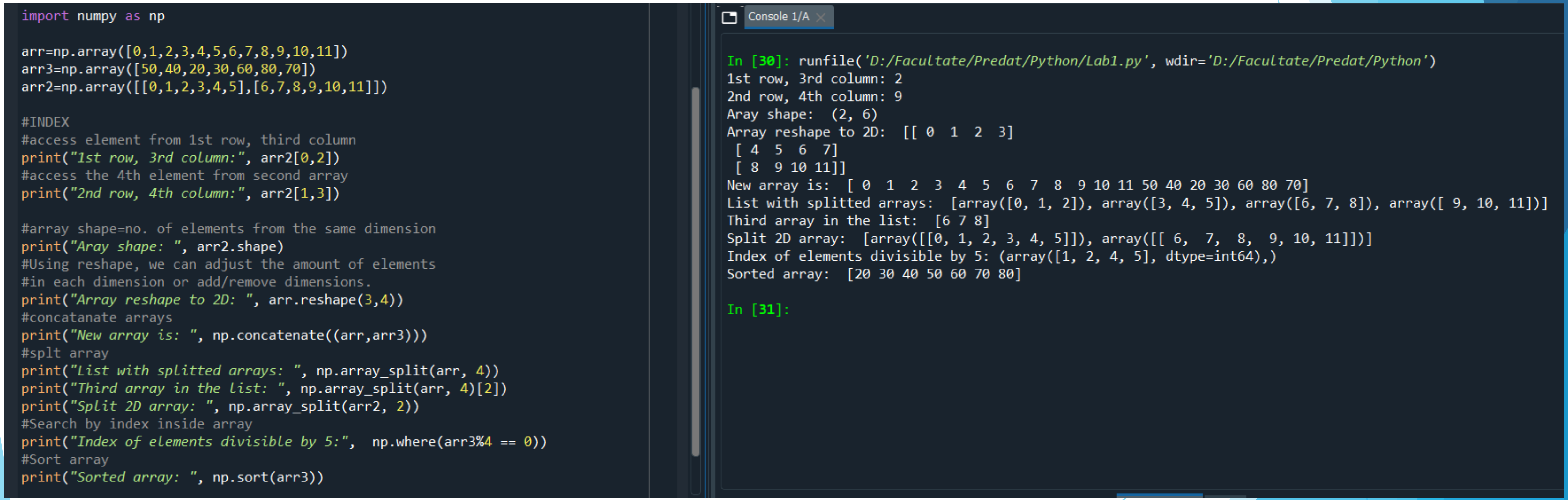


```python
import numpy as np

arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
#2D array
arr2 = np.array([[0, 1, 2, 3, 4, 5],
                 [6, 7, 8, 9, 10, 11]])

#print the values in index1, index2, index3, index4
print(arr[1:5])
print("----")
#print every other number in the array
print(arr[::2])
print("----")
#print from both parts of arr2
print(arr2[0:2, 2:5])
print("----")

x=np.random.random(2)
print(x)
```



```
[1 2 3 4]
----
[ 0  2  4  6  8 10]
----
[[ 2  3  4]
 [ 8  9 10]]
----
[0.13262625 0.60643364]
```

numpy
-examples-



```python
import numpy as np

arr=np.array([0,1,2,3,4,5,6,7,8,9,10,11])
arr3=np.array([50,40,20,30,60,80,70])
arr2=np.array([[0,1,2,3,4,5],[6,7,8,9,10,11]])

#INDEX
#access element from 1st row, third column
print("1st row, 3rd column:", arr2[0,2])
#access the 4th element from second array
print("2nd row, 4th column:", arr2[1,3])

#array shape=no. of elements from the same dimension
print("Aray shape: ", arr2.shape)
#Using reshape, we can adjust the amount of elements
#in each dimension or add/remove dimensions.
print("Array reshape to 2D: ", arr.reshape(3,4))
#concatanate arrays
print("New array is: ", np.concatenate((arr,arr3)))
#splt array
print("List with splitted arrays: ", np.array_split(arr, 4))
print("Third array in the list: ", np.array_split(arr, 4)[2])
print("Split 2D array: ", np.array_split(arr2, 2))
#Search by index inside array
print("Index of elements divisible by 5:",  np.where(arr3%4 == 0))
#Sort array
print("Sorted array: ", np.sort(arr3))
```

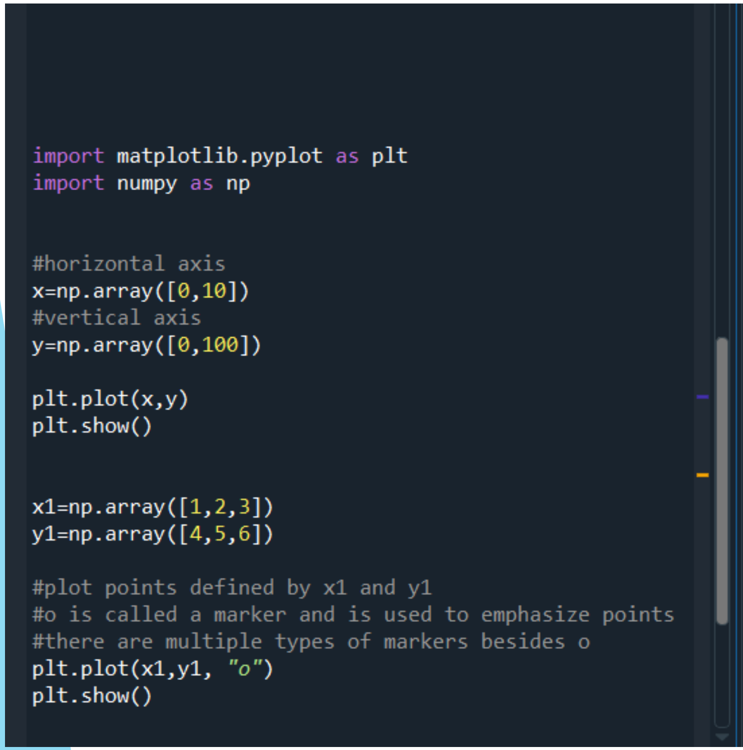
```
In [30]: runfile('D:/Facultate/Predat/Python/Lab1.py', wdir='D:/Facultate/Predat/Python')
1st row, 3rd column: 2
2nd row, 4th column: 9
Aray shape:  (2, 6)
Array reshape to 2D:  [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
New array is:  [ 0  1  2  3  4  5  6  7  8  9 10 11 50 40 20 30 60 80 70]
List with splitted arrays:  [array([0, 1, 2]), array([3, 4, 5]), array([6, 7, 8]), array([ 9, 10, 11])]
Third array in the list:  [6 7 8]
Split 2D array:  [array([[0, 1, 2, 3, 4, 5]]), array([[ 6,  7,  8,  9, 10, 11]])]
Index of elements divisible by 5: (array([1, 2, 4, 5], dtype=int64),)
Sorted array:  [20 30 40 50 60 70 80]

In [31]:
```

Numpy
-useful functions for arrays-

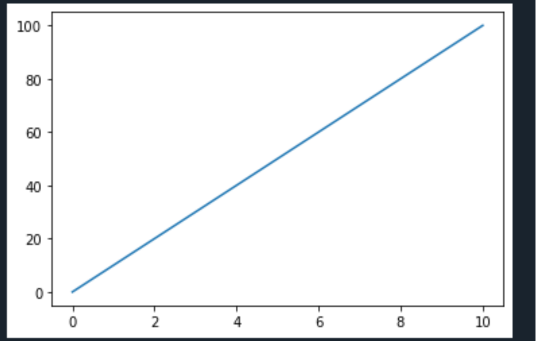
- zeros() – create an array populated with 0
- ones() – create an array populated with 1
- arrange() – return an array with values within a specified range
- identity() – create an identity matrix (m x m matrix with 1s in main diagonal)
- min() – return the minimum value within an array
- max() – return the maximum value within an array
- mean() – return the mean of the elements within an array
- put() - replace specified elements of an array with given values
- copyto() - copy the content of one array into another.
- interesctld() - return all the unique values from both arrays in a sorted manner
- setdiffld() - return all the unique elements from array 1 that are not present in array 2
- unionld() – combine 2 arrays into one (no duplicates)
- savetxt() - save the content of an array inside a text file
- loadtxt() - load the content of an array from a text file

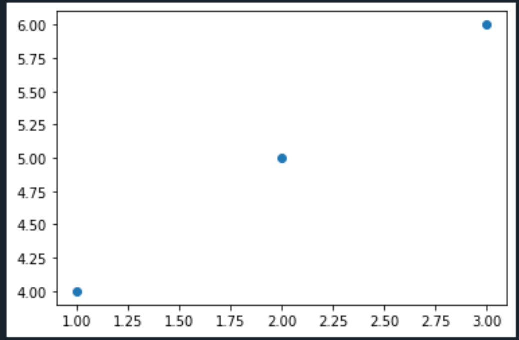
matplotlib

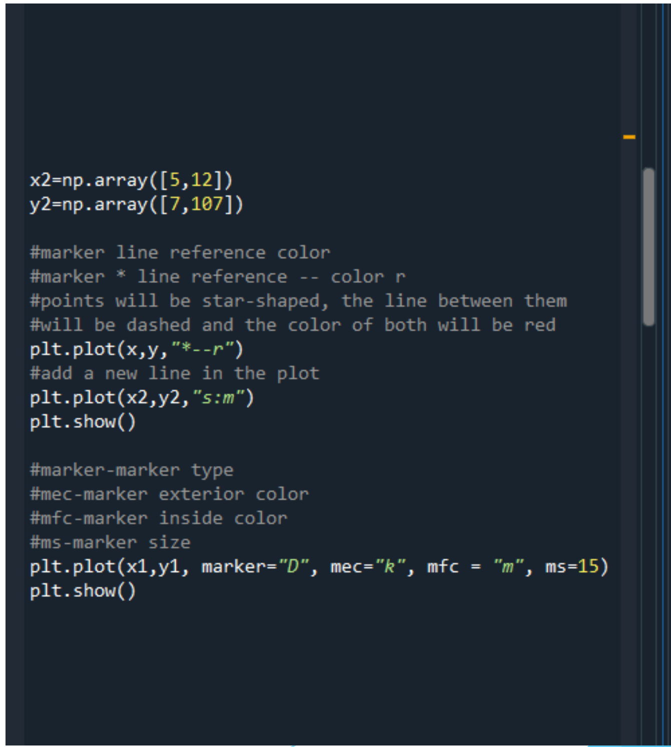
- Open-source library used for plotting graphs (pip install matplotlib)
- pyplot – module inside matplotlib with a large variety of functions



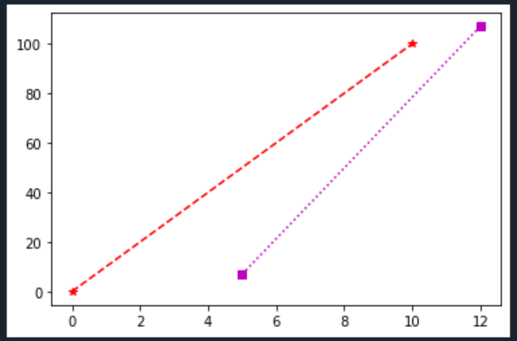
```
import matplotlib.pyplot as plt
import numpy as np

#horizontal axis
x=np.array([0,10])
#vertical axis
y=np.array([0,100])

plt.plot(x,y)
plt.show()


x1=np.array([1,2,3])
y1=np.array([4,5,6])

#plot points defined by x1 and y1
#o is called a marker and is used to emphasize points
#there are multiple types of markers besides o
plt.plot(x1,y1, "o")
plt.show()
```

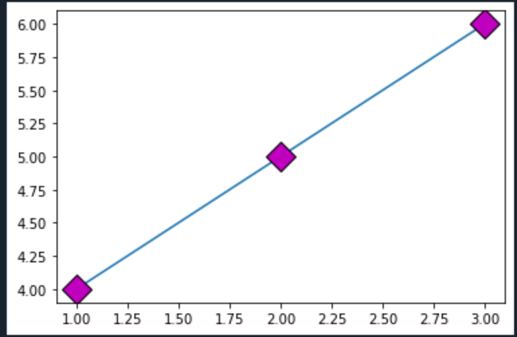






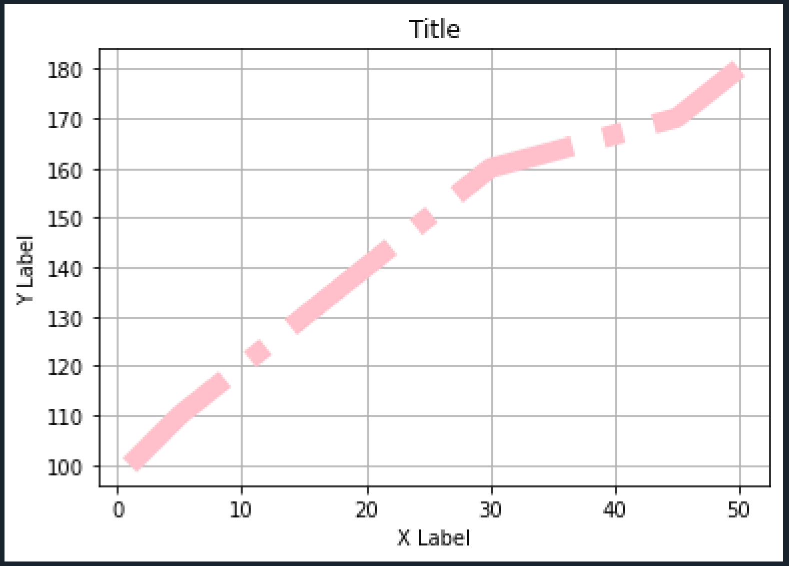
```
x2=np.array([5,12])
y2=np.array([7,107])

#marker line reference color
#marker * line reference -- color r
#points will be star-shaped, the line between them
#will be dashed and the color of both will be red
plt.plot(x,y,"*--r")
#add a new line in the plot
plt.plot(x2,y2,"s:m")
plt.show()

#marker-marker type
#mec-marker exterior color
#mfc-marker inside color
#ms-marker size
plt.plot(x1,y1, marker="D", mec="k", mfc = "m", ms=15)
plt.show()
```



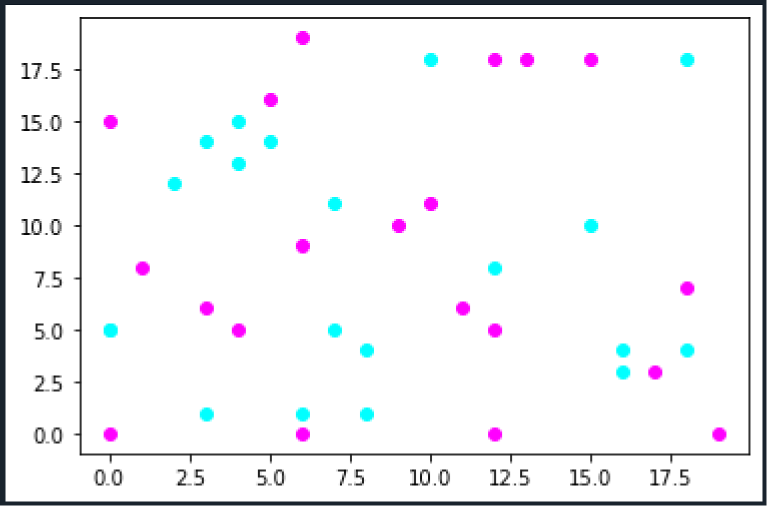


Examples

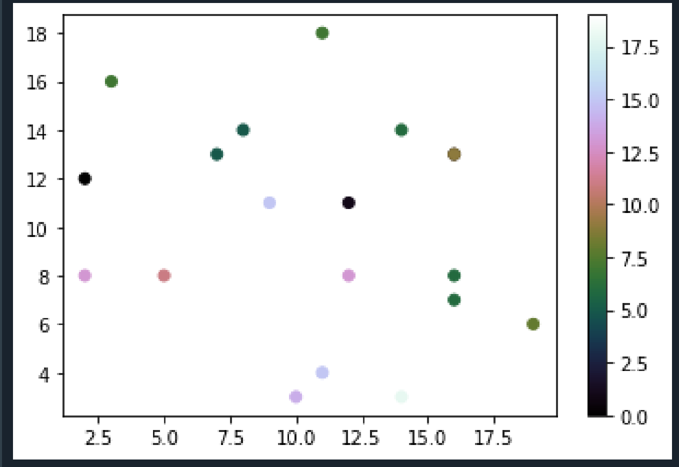
```
x2 = np.array([1,5,10,15,20,25,30,45,50])
y2 = np.array([100,110,120,130,140,150,160,170,180])

#line: linestyle-dash dot, color, width
plt.plot(x2, y2, linestyle="-.", color="pink", linewidth="10")
plt.grid()

plt.title("Title")
plt.xlabel("X Label")
plt.ylabel("Y Label")

plt.show()
```



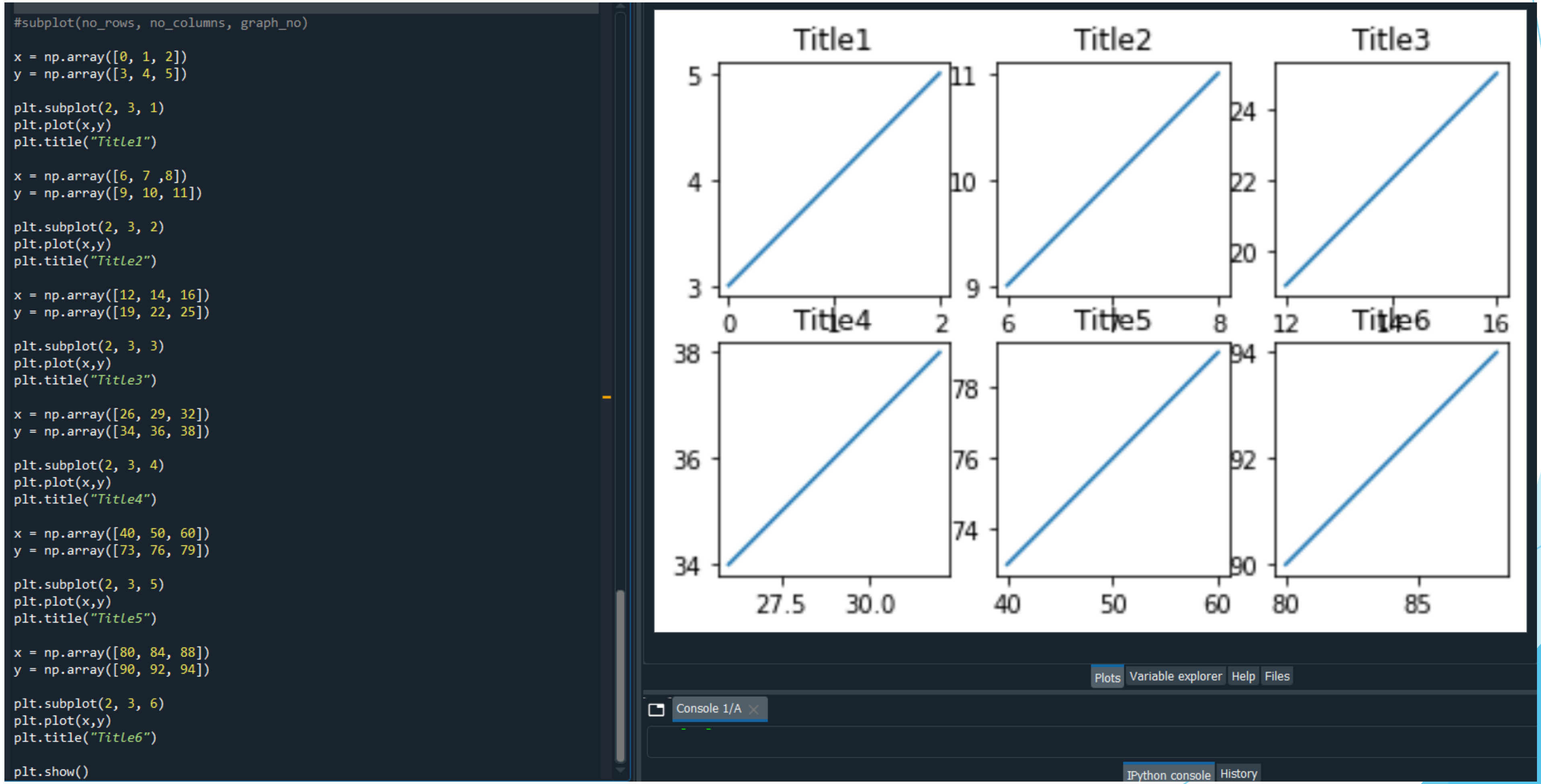
```
#scatter()-plot one dot for each point

x1 = np.random.randint(20, size=(20))
y1 = np.random.randint(20, size=(20))
plt.scatter(x1, y1, color = 'magenta')

x2 = np.random.randint(20, size=(20))
y2 = np.random.randint(20, size=(20))
plt.scatter(x2, y2, color = '#00FFFF')

plt.show()
```



```
x3 = np.random.randint(20, size=(20))
y3 = np.random.randint(20, size=(20))
colors = np.random.randint(20, size=(20))

#adding a color map (list of colors, where each color has a value)
plt.scatter(x3, y3, c=colors, cmap='cubehelix')

plt.colorbar()

plt.show()
```

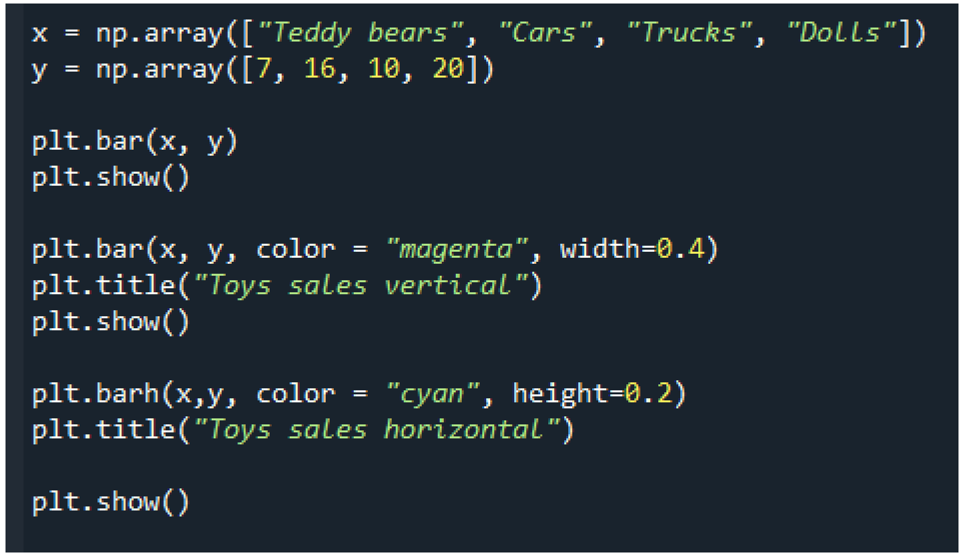


Example –subplot()

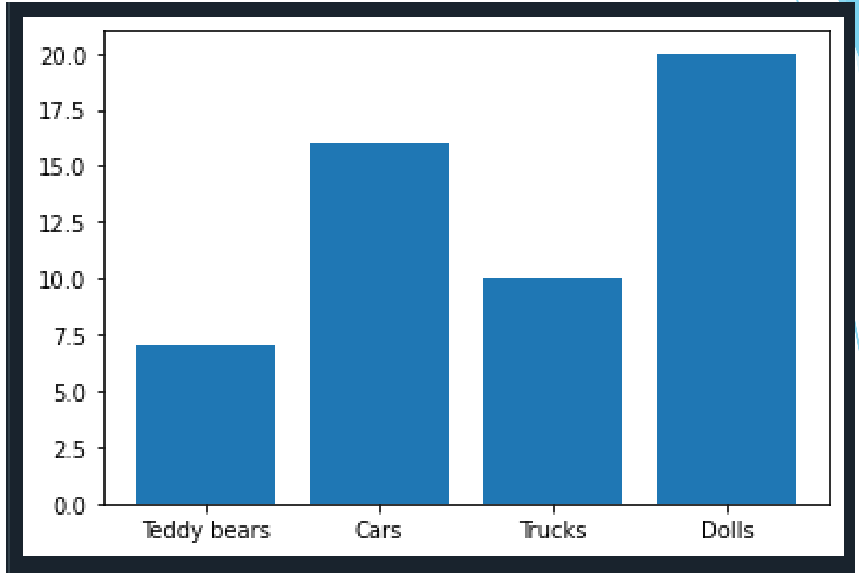


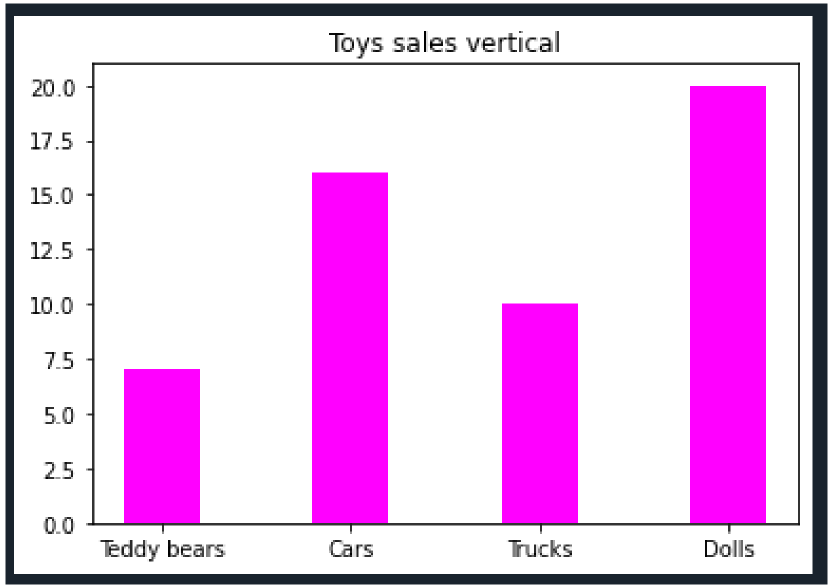
```
#subplot(no_rows, no_columns, graph_no)

x = np.array([0, 1, 2])
y = np.array([3, 4, 5])

plt.subplot(2, 3, 1)
plt.plot(x,y)
plt.title("Title1")

x = np.array([6, 7 ,8])
y = np.array([9, 10, 11])

plt.subplot(2, 3, 2)
plt.plot(x,y)
plt.title("Title2")

x = np.array([12, 14, 16])
y = np.array([19, 22, 25])

plt.subplot(2, 3, 3)
plt.plot(x,y)
plt.title("Title3")

x = np.array([26, 29, 32])
y = np.array([34, 36, 38])

plt.subplot(2, 3, 4)
plt.plot(x,y)
plt.title("Title4")

x = np.array([40, 50, 60])
y = np.array([73, 76, 79])

plt.subplot(2, 3, 5)
plt.plot(x,y)
plt.title("Title5")

x = np.array([80, 84, 88])
y = np.array([90, 92, 94])

plt.subplot(2, 3, 6)
plt.plot(x,y)
plt.title("Title6")

plt.show()
```

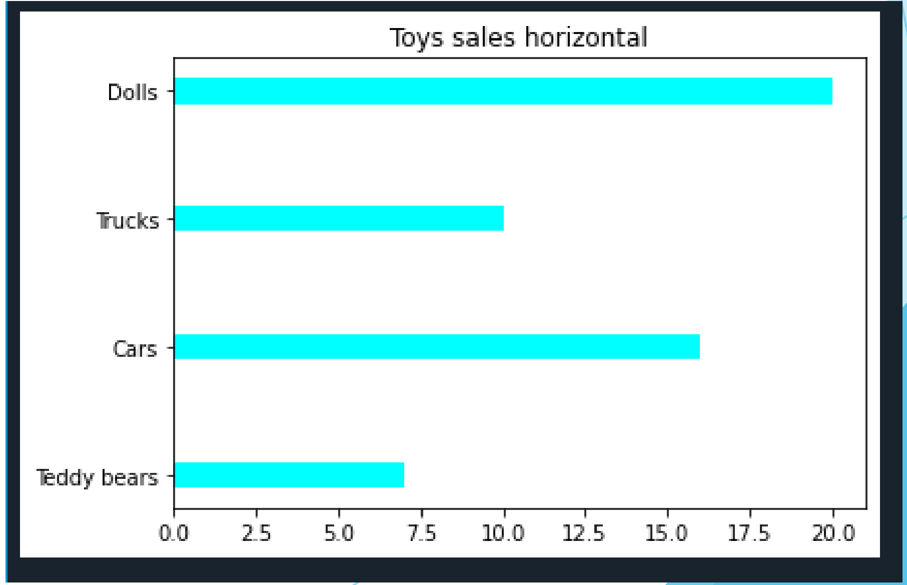
Examples - bars



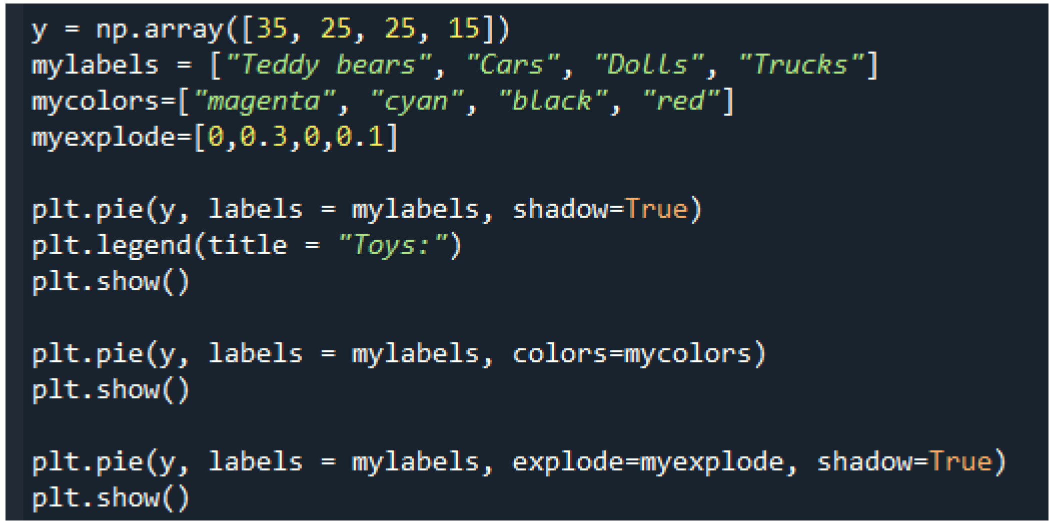
```
x = np.array(["Teddy bears", "Cars", "Trucks", "Dolls"])
y = np.array([7, 16, 10, 20])

plt.bar(x, y)
plt.show()

plt.bar(x, y, color = "magenta", width=0.4)
plt.title("Toys sales vertical")
plt.show()

plt.barh(x,y, color = "cyan", height=0.2)
plt.title("Toys sales horizontal")

plt.show()
```



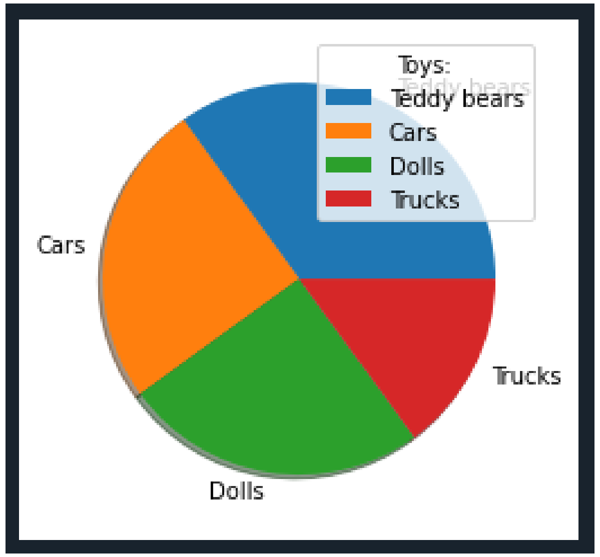


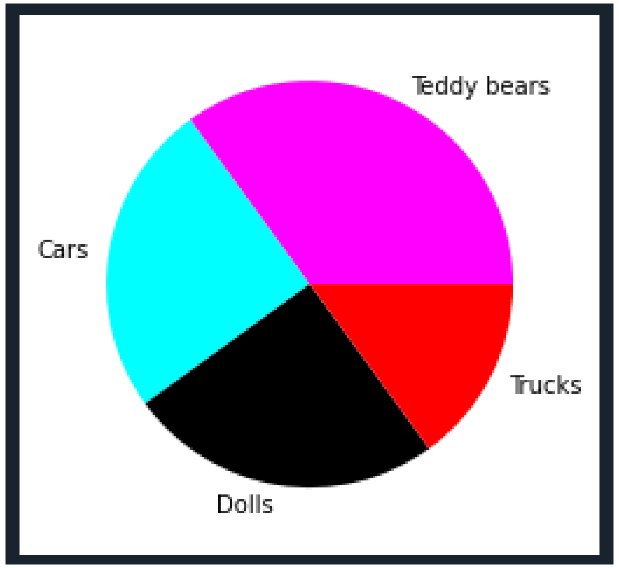


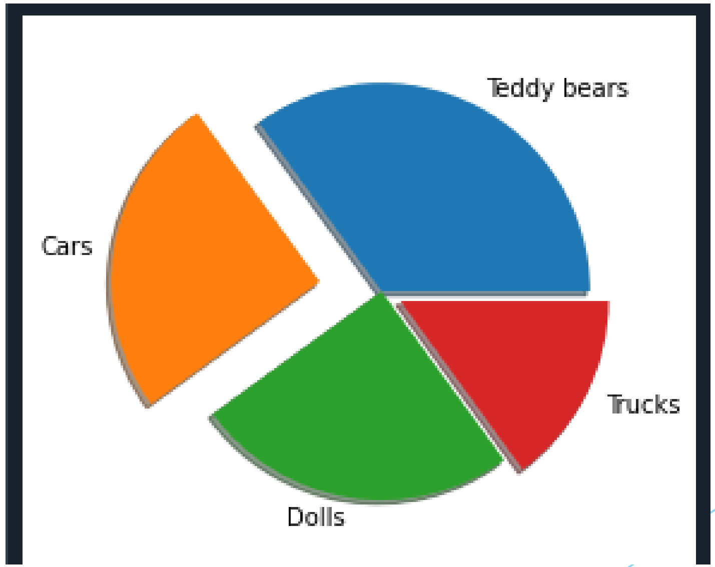
Examples - pies



```
y = np.array([35, 25, 25, 15])
mylabels = ["Teddy bears", "Cars", "Dolls", "Trucks"]
mycolors=["magenta", "cyan", "black", "red"]
myexplode=[0,0.3,0,0.1]

plt.pie(y, labels = mylabels, shadow=True)
plt.legend(title = "Toys:")
plt.show()

plt.pie(y, labels = mylabels, colors=mycolors)
plt.show()

plt.pie(y, labels = mylabels, explode=myexplode, shadow=True)
plt.show()
```





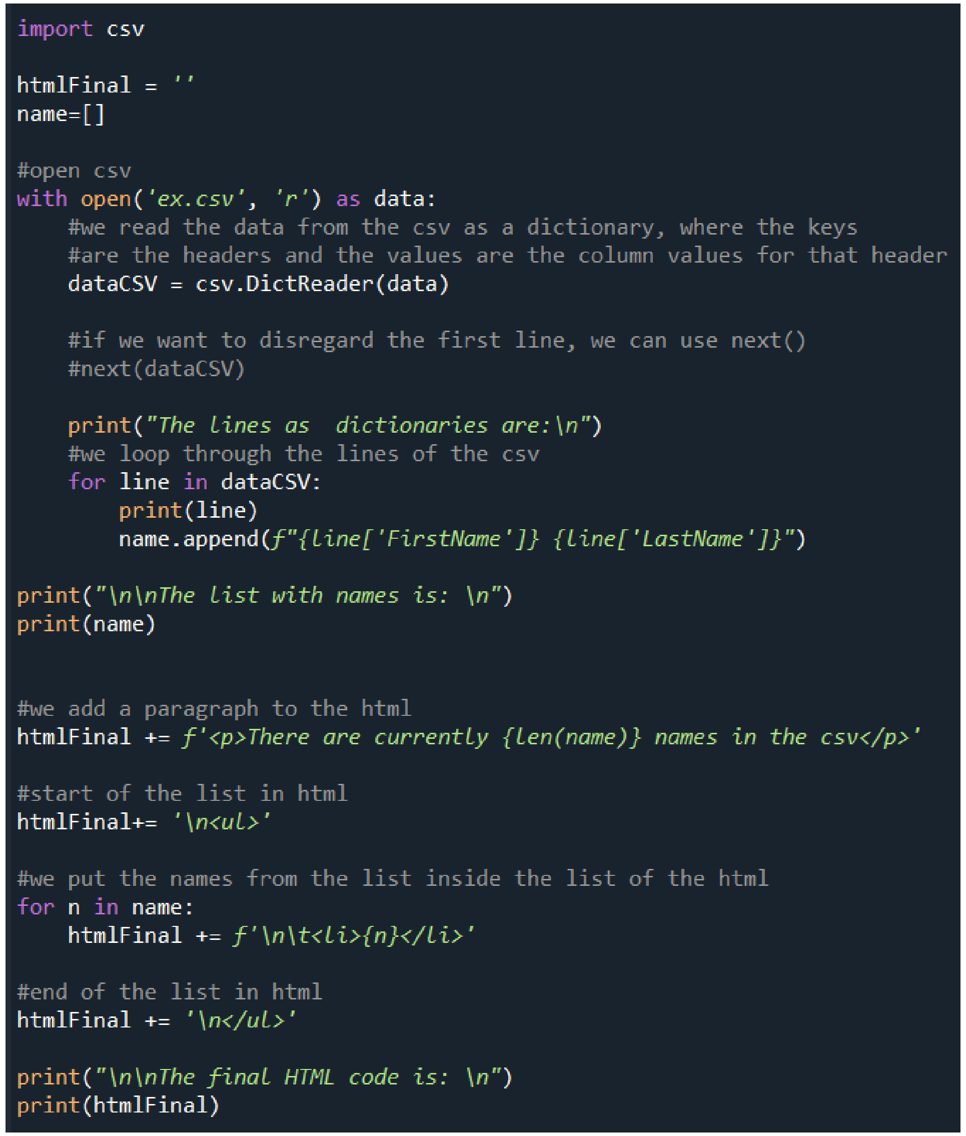


Parser

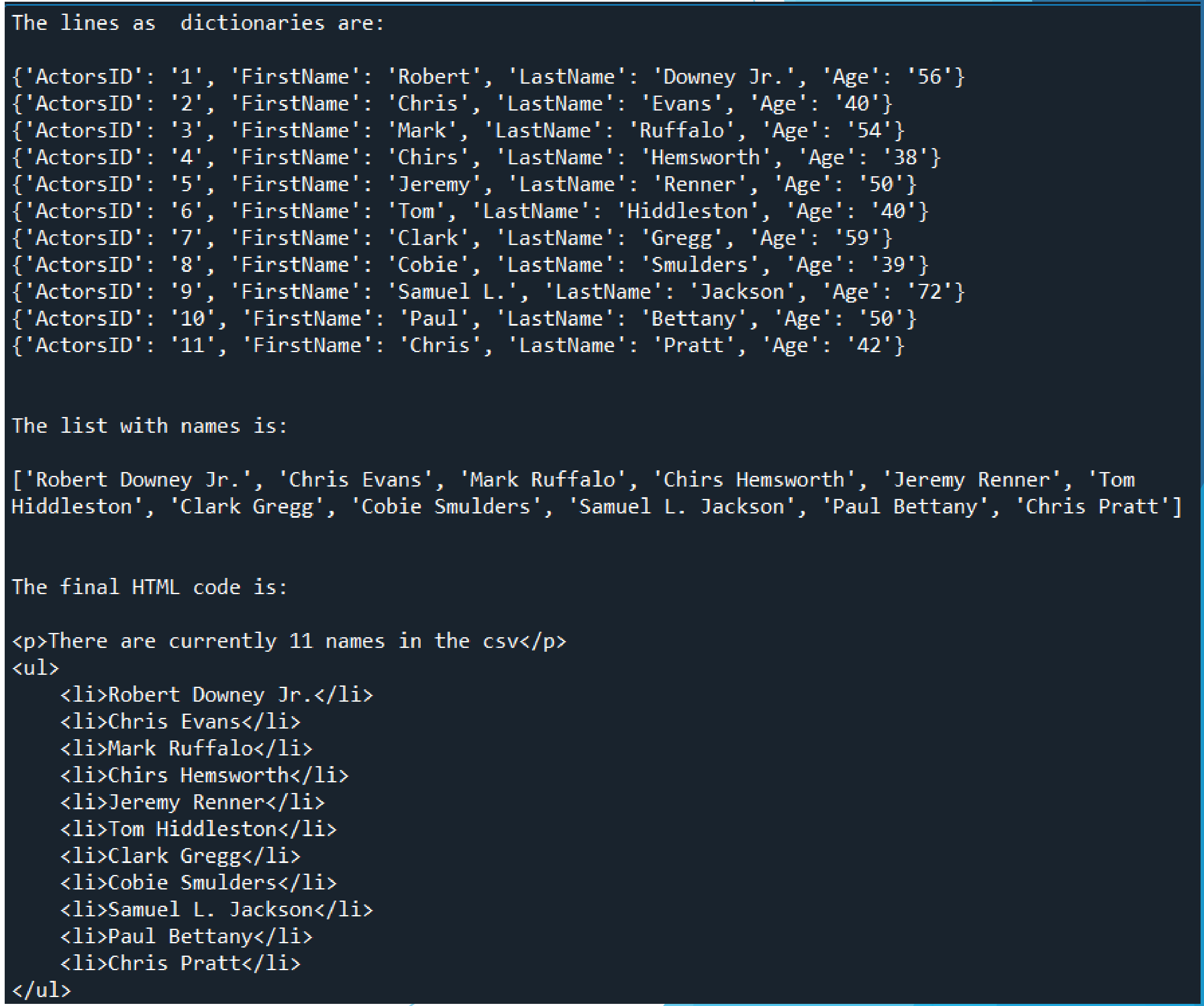
- Parser = breaks data into smaller elements for easy translation into another language
- Types of parsing:
  - Top-down (predictive parsing or recursive parsing) - starts from the top of the parse tree, move downwards, evaluates rules of grammar
  - Bottom-up (shift-reduce parsing) - starts from the lowest level of the parse tree, move upwards and evaluates rules of grammar

Example

- Parse data from a csv to a html list



```python
import csv

htmlFinal = ''
name=[]

#open csv
with open('ex.csv', 'r') as data:
    #we read the data from the csv as a dictionary, where the keys
    #are the headers and the values are the column values for that header
    dataCSV = csv.DictReader(data)

    #if we want to disregard the first line, we can use next()
    #next(dataCSV)

    print("The lines as  dictionaries are:\n")
    #we loop through the lines of the csv
    for line in dataCSV:
        print(line)
        name.append(f"{line['FirstName']} {line['LastName']}")

print("\n\nThe list with names is: \n")
print(name)


#we add a paragraph to the html
htmlFinal += f'<p>There are currently {len(name)} names in the csv</p>'

#start of the list in html
htmlFinal+= '\n<ul>'

#we put the names from the list inside the list of the html
for n in name:
    htmlFinal += f'\n\t<li>{n}</li>'

#end of the list in html
htmlFinal += '\n</ul>'

print("\n\nThe final HTML code is: \n")
print(htmlFinal)
```



```
The lines as  dictionaries are:

{'ActorsID': '1', 'FirstName': 'Robert', 'LastName': 'Downey Jr.', 'Age': '56'}
{'ActorsID': '2', 'FirstName': 'Chris', 'LastName': 'Evans', 'Age': '40'}
{'ActorsID': '3', 'FirstName': 'Mark', 'LastName': 'Ruffalo', 'Age': '54'}
{'ActorsID': '4', 'FirstName': 'Chirs', 'LastName': 'Hemsworth', 'Age': '38'}
{'ActorsID': '5', 'FirstName': 'Jeremy', 'LastName': 'Renner', 'Age': '50'}
{'ActorsID': '6', 'FirstName': 'Tom', 'LastName': 'Hiddleston', 'Age': '40'}
{'ActorsID': '7', 'FirstName': 'Clark', 'LastName': 'Gregg', 'Age': '59'}
{'ActorsID': '8', 'FirstName': 'Cobie', 'LastName': 'Smulders', 'Age': '39'}
{'ActorsID': '9', 'FirstName': 'Samuel L.', 'LastName': 'Jackson', 'Age': '72'}
{'ActorsID': '10', 'FirstName': 'Paul', 'LastName': 'Bettany', 'Age': '50'}
{'ActorsID': '11', 'FirstName': 'Chris', 'LastName': 'Pratt', 'Age': '42'}


The list with names is:

['Robert Downey Jr.', 'Chris Evans', 'Mark Ruffalo', 'Chirs Hemsworth', 'Jeremy Renner', 'Tom
Hiddleston', 'Clark Gregg', 'Cobie Smulders', 'Samuel L. Jackson', 'Paul Bettany', 'Chris Pratt']


The final HTML code is:

<p>There are currently 11 names in the csv</p>
<ul>
    <li>Robert Downey Jr.</li>
    <li>Chris Evans</li>
    <li>Mark Ruffalo</li>
    <li>Chirs Hemsworth</li>
    <li>Jeremy Renner</li>
    <li>Tom Hiddleston</li>
    <li>Clark Gregg</li>
    <li>Cobie Smulders</li>
    <li>Samuel L. Jackson</li>
    <li>Paul Bettany</li>
    <li>Chris Pratt</li>
</ul>
```

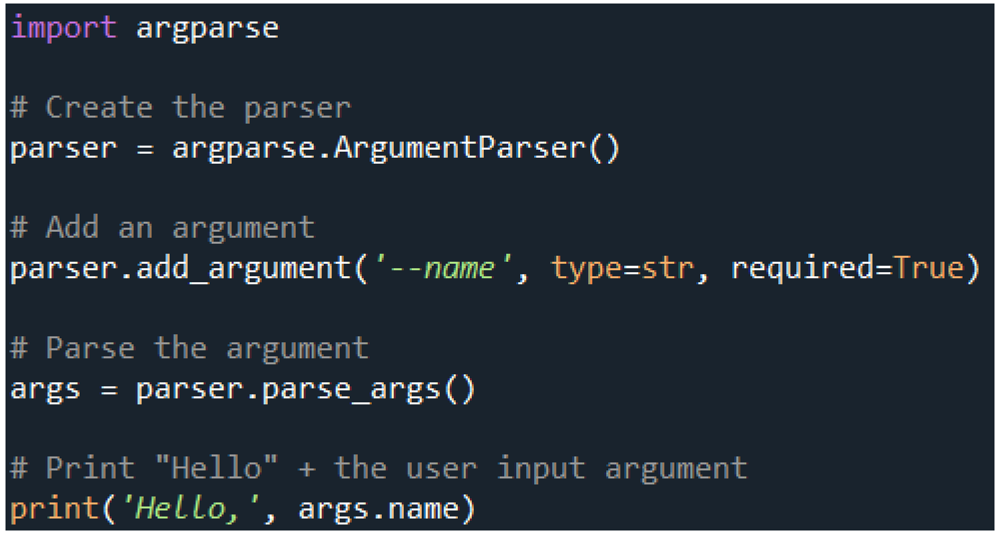
argparse

- argparse (python module)- parser for command-line options, arguments and sub-commands

- Step 1: create a parser object (argparse.ArgumentParser())
- Step 2: add arguments to the parser object (.add_argument)
- Step 3: parse arguments (checks for consistency and stores the values) (.parse_args())
- Step 4: use the returned object from the parser object to access the values supplied in the arguments

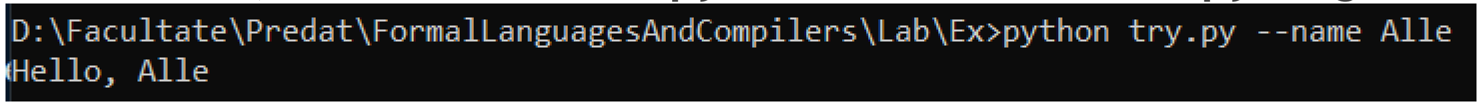
Example

- A simple Hello, user program



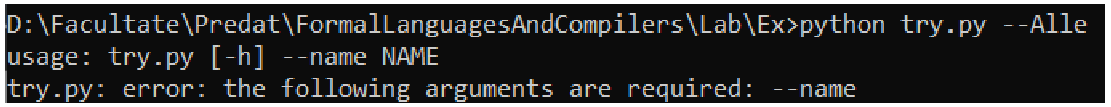
```python
import argparse

# Create the parser
parser = argparse.ArgumentParser()

# Add an argument
parser.add_argument('--name', type=str, required=True)

# Parse the argument
args = parser.parse_args()

# Print "Hello" + the user input argument
print('Hello,', args.name)
```

- In CMD we need to go to the folder where the program is.
- To run it, we need to write **python nameOfTheFile.py -argument VALUE**



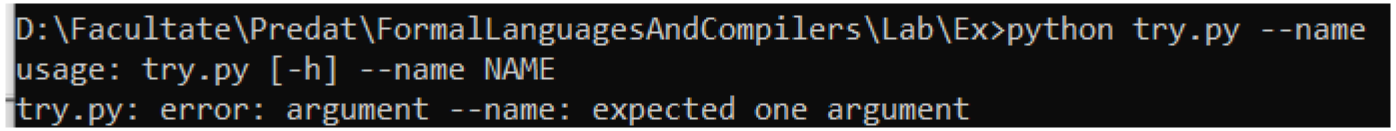
```
D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python try.py --name Alle
Hello, Alle
```

- Error for forgetting the argument:



```
D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python try.py --Alle
usage: try.py [-h] --name NAME
try.py: error: the following arguments are required: --name
```

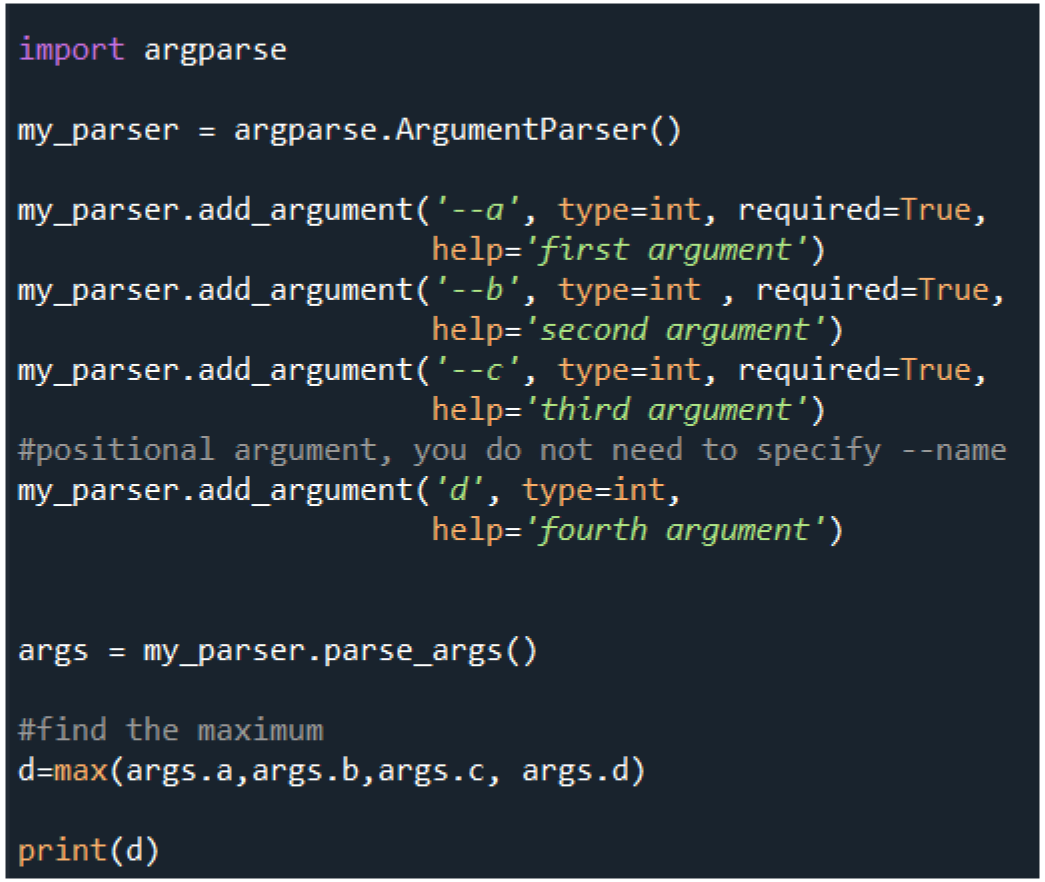
- Error for forgetting the value:



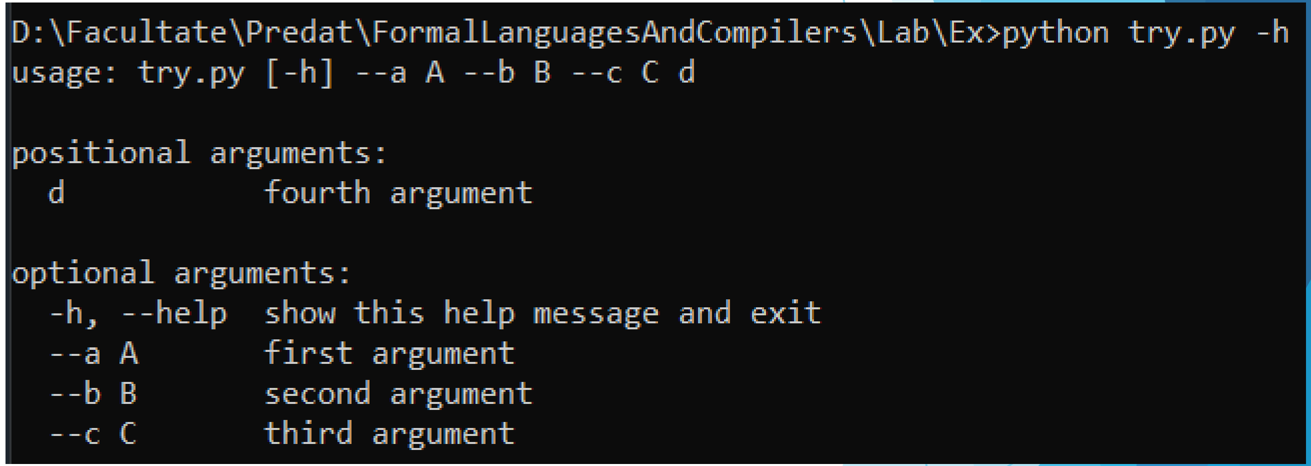
```
D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python try.py --name
usage: try.py [-h] --name NAME
try.py: error: argument --name: expected one argument
```

Example

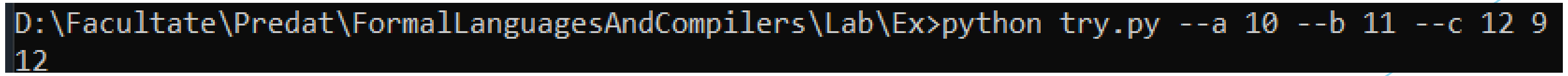
- Maximum of 4 numbers inputted by the user



```python
import argparse

my_parser = argparse.ArgumentParser()

my_parser.add_argument('--a', type=int, required=True,
                       help='first argument')
my_parser.add_argument('--b', type=int , required=True,
                       help='second argument')
my_parser.add_argument('--c', type=int, required=True,
                       help='third argument')
#positional argument, you do not need to specify --name
my_parser.add_argument('d', type=int,
                       help='fourth argument')


args = my_parser.parse_args()

#find the maximum
d=max(args.a,args.b,args.c, args.d)

print(d)
```



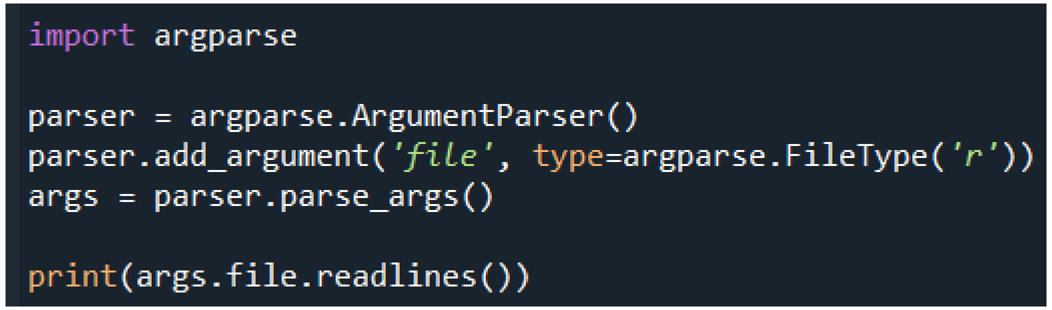
```
D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python try.py -h
usage: try.py [-h] --a A --b B --c C d

positional arguments:
  d            fourth argument

optional arguments:
  -h, --help   show this help message and exit
  --a A        first argument
  --b B        second argument
  --c C        third argument
```



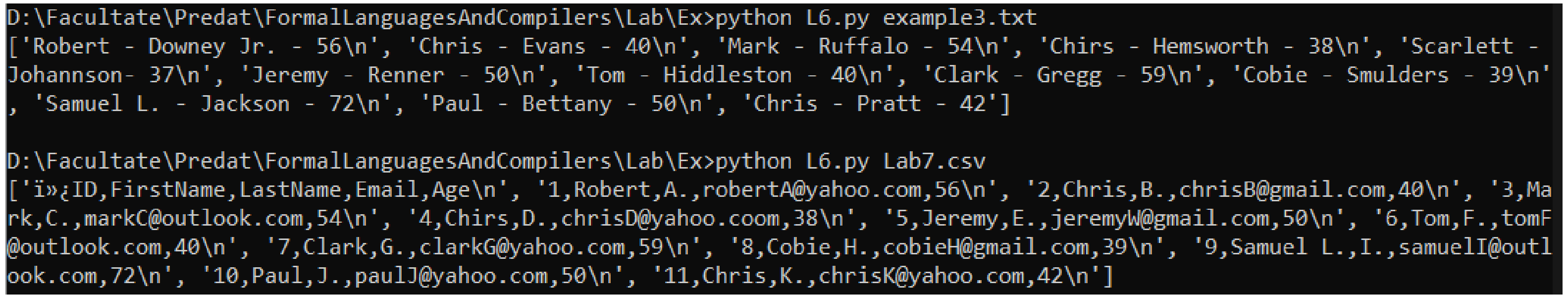
```
D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python try.py --a 10 --b 11 --c 12 9
12
```

Example

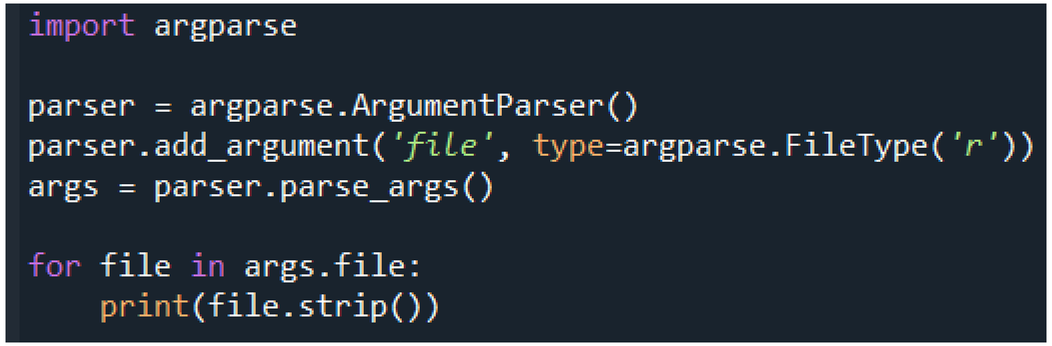
- Read content of a file



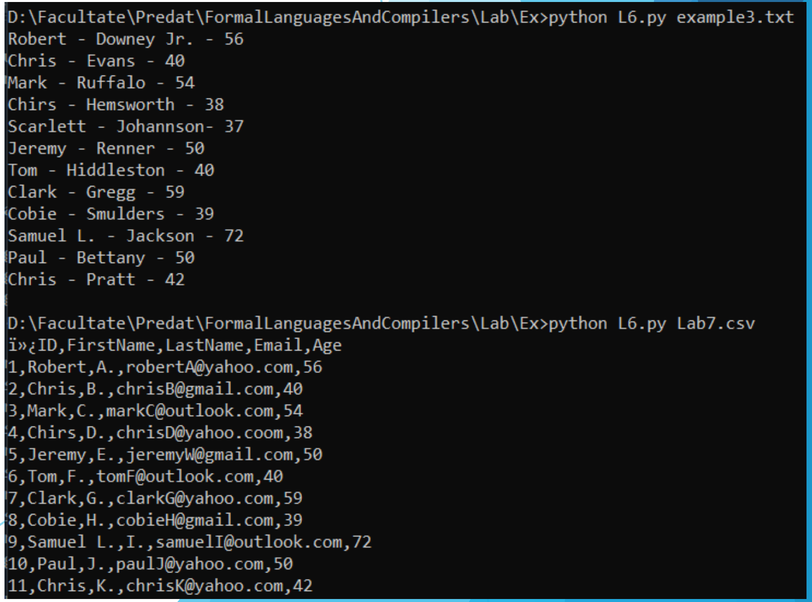
```python
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('file', type=argparse.FileType('r'))
args = parser.parse_args()

print(args.file.readlines())
```



```
D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python L6.py example3.txt
['Robert - Downey Jr. - 56\n', 'Chris - Evans - 40\n', 'Mark - Ruffalo - 54\n', 'Chirs - Hemsworth - 38\n', 'Scarlett -
Johannson- 37\n', 'Jeremy - Renner - 50\n', 'Tom - Hiddleston - 40\n', 'Clark - Gregg - 59\n', 'Cobie - Smulders - 39\n'
, 'Samuel L. - Jackson - 72\n', 'Paul - Bettany - 50\n', 'Chris - Pratt - 42']

D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python L6.py Lab7.csv
['ï»¿ID,FirstName,LastName,Email,Age\n', '1,Robert,A.,robertA@yahoo.com,56\n', '2,Chris,B.,chrisB@gmail.com,40\n', '3,Ma
rk,C.,markC@outlook.com,54\n', '4,Chirs,D.,chrisD@yahoo.coom,38\n', '5,Jeremy,E.,jeremyW@gmail.com,50\n', '6,Tom,F.,tomF
@outlook.com,40\n', '7,Clark,G.,clarkG@yahoo.com,59\n', '8,Cobie,H.,cobieH@gmail.com,39\n', '9,Samuel L.,I.,samuelI@outl
ook.com,72\n', '10,Paul,J.,paulJ@yahoo.com,50\n', '11,Chris,K.,chrisK@yahoo.com,42\n']
```



```python
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('file', type=argparse.FileType('r'))
args = parser.parse_args()

for file in args.file:
    print(file.strip())
```



```
D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python L6.py example3.txt
Robert - Downey Jr. - 56
Chris - Evans - 40
Mark - Ruffalo - 54
Chirs - Hemsworth - 38
Scarlett - Johannson- 37
Jeremy - Renner - 50
Tom - Hiddleston - 40
Clark - Gregg - 59
Cobie - Smulders - 39
Samuel L. - Jackson - 72
Paul - Bettany - 50
Chris - Pratt - 42

D:\Facultate\Predat\FormalLanguagesAndCompilers\Lab\Ex>python L6.py Lab7.csv
ï»¿ID,FirstName,LastName,Email,Age
1,Robert,A.,robertA@yahoo.com,56
2,Chris,B.,chrisB@gmail.com,40
3,Mark,C.,markC@outlook.com,54
4,Chirs,D.,chrisD@yahoo.coom,38
5,Jeremy,E.,jeremyW@gmail.com,50
6,Tom,F.,tomF@outlook.com,40
7,Clark,G.,clarkG@yahoo.com,59
8,Cobie,H.,cobieH@gmail.com,39
9,Samuel L.,I.,samuelI@outlook.com,72
10,Paul,J.,paulJ@yahoo.com,50
11,Chris,K.,chrisK@yahoo.com,42
```

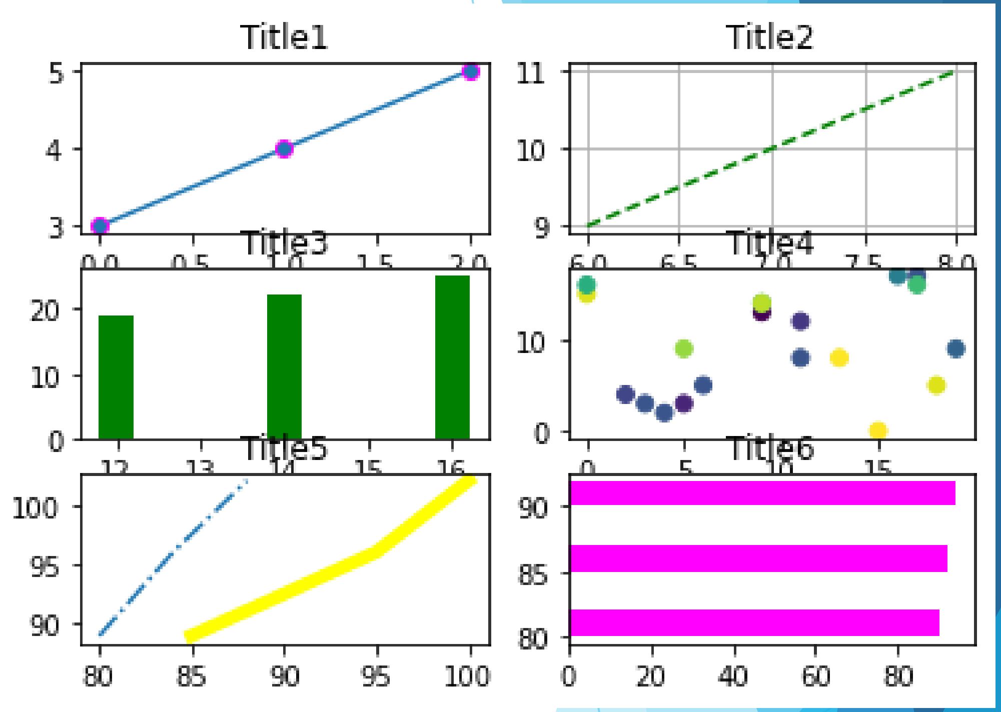
Exercises

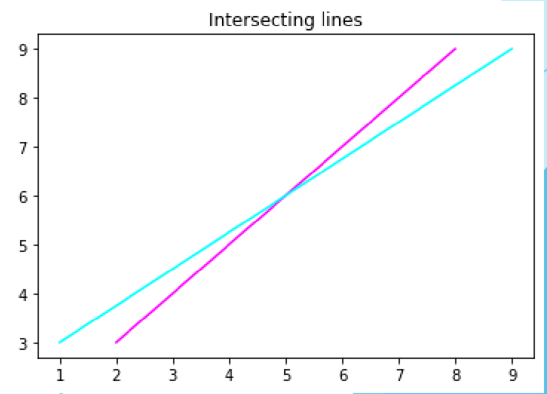
1. list1=[1.4,5.9,2.6,7.3,8.5,9.3,6.1] ; Convert it into an array using numpy.
2. Write a program that creates a 1D array of even numbers between 70 and 91. (Hint: use arrange() from numpy)
3. Write a program that creates a 3x3 matrix populated by 0s and having the numbers 3,7,11 on the main diagonal. (Hint: use diag() from numpy)
4. Write a program that creates a 2x3 array of numbers between 10 and 16. (Hint: use arrange() and reshape() from numpy).
5. ex5=np.array([[10,50,70,20,40],[5,45,95,35,65]]) ; Print all the numbers below and above 43 inside ex4 array.
6. Use ex5 array. Write a program that saves the array in a text file, then load the content of it.

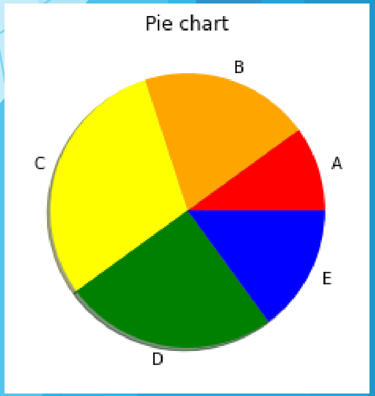
Exercises

7. Using matplotlib, create a 3x2 subplot with:
    1. Line with custom marker color for points
    2. Line with custom style and grid
    3. Vertical bars with custom color, width
    4. Scattered points with different colors
    5. 1 line with custom style, color & 1 line with custom width, color
    6. Horizontal bars with custom color, height
8. Create a plot with intersecting lines of different colors.
9. Create a pie chart with 5 items of custom colors and shadow.

!Use titles for all graphs







Homework

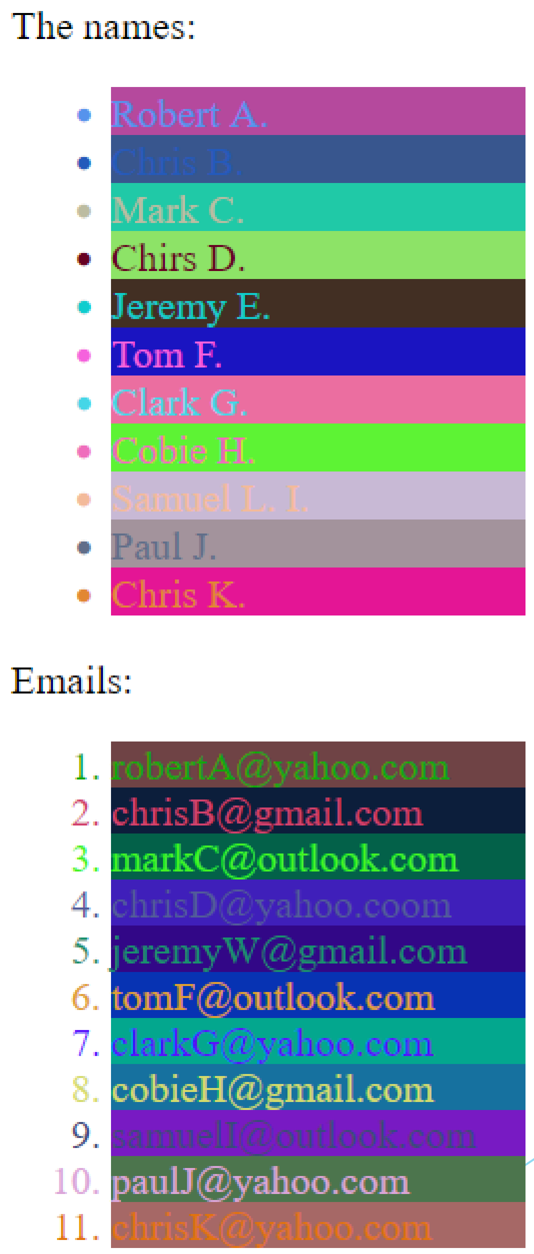
- Create a parser that takes as arguments two numbers and computes their sum, difference, multiplication, division and mean.

- Create a parser that takes as arguments two numbers and a string (mathematical symbol) and, depending on the symbol, the program computes their sum, difference, multiplication, division or mean. (e.g., 7 3 +, the output should be 10; 5 6 *, the output should be 30)

Homework

- Write a program in Python which creates an HTML file with an ordered list with and an unordered list having items of any color (besides black) on any background color (besides white- you can make the whole background a color, not each element with different ones) from the Email and FirstName columns of the Lab7.csv file. (The example is only indicative; you can make your own to look like however you wish)

- Write a program in python which creates an HTML file with a table with the columns "Name" and "Age" and populate it with the values from the following lists: name=["Alex", "Emma", "Kate", "Ryan", "Lily"], age=[21,25,36,31,27] (Hint: use open() in w/a mode and use write() to write in the file)



The names:

- Robert A.
- Chris B.
- Mark C.
- Chirs D.
- Jeremy E.
- Tom F.
- Clark G.
- Cobie H.
- Samuel L. I.
- Paul J.
- Chris K.

Emails:

1. robertA@yahoo.com
2. chrisB@gmail.com
3. markC@outlook.com
4. chrisD@yahoo.coom
5. jeremyW@gmail.com
6. tomF@outlook.com
7. clarkG@yahoo.com
8. cobieH@gmail.com
9. samuelI@outlook.com
10. paulJ@yahoo.com
11. chrisK@yahoo.com