• **matplotlib** is a Python 2D plotting library

• You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc.

• Installing matplotlib: [http://matplotlib.org/users/installing.html](http://matplotlib.org/users/installing.html)


• Or use Anaconda that provides numerous built-in Python packages including matplotlib: [https://www.continuum.io/downloads](https://www.continuum.io/downloads)
Vertical Bar Chart Plotting

• Example:

```python
import matplotlib.pyplot as plot

students = ['Emre', 'Esma', 'Ahmet', 'Demet', 'Kerem']
grades = [90, 30, 45, 100, 87]
x_pos = [x for x in range(len(students))]

plot.bar(x_pos, grades, align='center', color='b', alpha=0.8)
plot.xticks(x_pos, students)
plot.ylabel('Score')
plot.title('Exam Grades')
plot.show()
```

• Output:
Horizontal Bar Chart Plotting

• Example:

```python
import matplotlib.pyplot as plot

students = ['Emre', 'Esma', 'Ahmet', 'Demet', 'Kerem']
grades = [90, 30, 45, 100, 87]
y_pos = [x for x in range(len(students))]

plot.barh(y_pos, grades, align='center', color='g', alpha=0.5)
plot.yticks(y_pos, students)
plot.xlabel('Score')
plot.ylabel('Exam Grades')
plot.show()
```

• Output:
NumPy (http://www.numpy.org) is the fundamental package for scientific computing with Python. It supports among other things:

- a powerful N-dimensional array object,
- sophisticated (broadcasting) functions,
- useful linear algebra, Fourier transform, and random number capabilities,
- efficient multi-dimensional container of generic data,
- arbitrary data-types.

- Installing Packages in PyCharm (search for numpy):

- Or use Anaconda that provides numerous built-in Python packages including NumPy:
  https://www.continuum.io/downloads
A simple plot with a custom dashed line

• Example:

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

x = np.linspace(0, 10)

line, = plt.plot(x, np.sin(x), '--', linewidth=2, color="r")

dashes = [10, 5, 100, 5]  # 10 points on, 5 off, 100 on, 5 off
line.set_dashes(dashes)

plt.show()
```

New function: `numpy.linspace(start, stop)`

Returns evenly spaced numbers over a specified interval `[start, stop]`. 
A simple plot of fill function

• Example:

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

x = np.linspace(0, 1)
y = np.sin(4 * np.pi * x) * np.exp(-5 * x)
plt.fill(x, y, 'y')
plt.grid(True)
plt.show()
```

New functions:
- `numpy.sin()` - Trigonometric sine, element-wise
- `numpy.exp()` - Calculate the exponential of all elements in the input array
- `numpy.pi()` - \( \pi \) mathematical constant

• Output:
Histogram Plotting

A histogram is a graphical representation of the distribution of numerical data.

• Example:

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

mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(10000)

num_bins = 50
# the histogram of the data
n, bins, patches = plt.hist(x, num_bins, normed=1, facecolor='green')
# add a 'best fit' line
y = mlab.normpdf(bins, mu, sigma)
plt.plot(bins, y, 'b-')
plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

# Tweak spacing to prevent clipping of ylabel
plt.subplots_adjust(left=0.15)
plt.show()
```

New function: `numpy.random.randn(dimension)`

Returns a sample (or samples) from the “standard normal” distribution
Histogram Plotting Continued (Subplots)

• Example:

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

mu = 200
sigma = 25
x = mu + sigma*np.random.randn(10000)
print(x)
fig, (ax0, ax1) = plt.subplots(nrows=2, figsize=(8, 4))

ax0.hist(x, 20, normed=1, histtype='stepfilled', facecolor='g', alpha=0.75)
ax0.set_title('Histogram type: stepfilled')

# Create a histogram by providing the bin edges (unequally spaced).
bins = [100, 150, 180, 195, 205, 220, 250, 300]
ax1.hist(x, bins, normed=1, histtype='bar', rwidth=0.7)
ax1.set_title('Histogram type: bar, unequal bins')
plt.tight_layout()
plt.show()
```

• Output:
2D Plotting and Scientific Computing in Python

• For more matplotlib examples: http://matplotlib.org/examples/index.html

• Plotting Commands Summary: http://matplotlib.org/api/pyplot_summary.html

• NumPy Manual: https://docs.scipy.org/doc/numpy/index.html
Exceptions

- Built-in Exceptions
- User-defined Exceptions
Built-in Exceptions

The simplest way to handle exceptions is with a "try-except" block:

Example 1:

```python
(x, y) = (5, 0)
try:
    z = x/y
except ZeroDivisionError:
    print("divide by zero")
```

Output: divide by zero
Example 2: except ValueError:

```python
first_number = input("First number: ")
second_number = input("Second number: ")

try:
    number1 = int(first_number)
    number2 = int(second_number)
    print(number1, "/", number2, ":", number1 / number2)

except ValueError:
    print("Error! Please enter number!")
```
Example 3: except ZeroDivisionError:

```python
first_number = input("First number: ")
second_number = input("Second number: ")

try:
    number1 = int(first_number)
    number2 = int(second_number)
    print(number1, "/", number2, ":", number1 / number2)
except ValueError:
    print("Error! Please enter number!")
except ZeroDivisionError:
    print("You can't divide a number to 0!")
```
Example 4: `except (ValueError, ZeroDivisionError)`

```python
first_number = input("First number: ")
second_number = input("Second number: ")
try:
    number1 = int(first_number)
    number2 = int(second_number)
    print(number1, "/", number2, "+", number1 / number2)
except (ValueError, ZeroDivisionError):
    print("Error!")
```
Example 5: try... except... as...

```python
first_number = input("First number: ")
second_number = input("Second number: ")

try:
    number1 = int(first_number)
    number2 = int(second_number)
    print(number1, "/", number2, "=", number1 / number2)
except (ValueError, ZeroDivisionError) as error:
    print("Error!")
    print("Original error message: ", error)
```
Example 6: try... except... else...

```python
for arg in sys.argv[1:]:
    try:
        f = open(arg, 'r')
    except IOError:
        print('cannot open', arg)
    else:
        print(arg, 'has', len(f.readlines()), 'lines')
    f.close()
```
Example 7: try... except... finally...

```python
try:
    file = open("dosyaad1", "r")
except IOError:
    print("error!")
finally:
    file.close()
```
Some Examples using Exceptions

```python
def main:
    try:
        file = open('example.txt', 'r')
        print(file.read())
    except IOError:
        print('An error occurred trying to read the file.')
    except ValueError:
        print('Non-numeric data found in the file.')
    except ImportError:
        print('NO module found')
    except EOFError:
        print('Why did you do an EOF on me?')
    except KeyboardInterrupt:
        print('You cancelled the operation.')
    except:
        print('An error occurred.')

if __name__ == '__main__':
    main()
```

Example 8:

```python
tr_character = "şçğöüî"

password = input("Enter your password: ")

for i in password:
    if i in tr_character:
        raise TypeError("You can't use Turkish characters in password!")
    else:
        pass

print("Password is excepted!")
```
Example 9:

```python
try:
    while True:
        if int(input('Guess a number: ')) == 5:
            raise ZeroDivisionError
except ZeroDivisionError:
    print ('You got it!')
```
Example 10:

```python
import sys

try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except OSError as err:
    print("OS error: {}".format(err))
except ValueError:
    print("Could not convert data to an integer.")
except:
    print("Unexpected error:", sys.exc_info()[0])
    raise
```
User-Defined Exceptions

Example 11:

class MyException(Exception):
    def __init__(self, t=0):
        self.numtries = t

try:
    for tries in range(1, 6):
        if int(input('Guess a number: ')) == 5:
            raise MyException(tries)
except MyException as e:
    print ('You got it in only %d tries!' % e.numtries)
else:
    print ('Too bad, you ran out of tries!')
Example 12 user-defined exceptions

```python
class Error(Exception):
    """Base class for other exceptions""
    pass

class ValueTooSmallError(Error):
    """Raised when the input value is too small""
    pass

class ValueTooLargeError(Error):
    """Raised when the input value is too large""
    pass

# our main program
# user guesses a number until he/she gets it right

# you need to guess this number
number = 10
```

This example continues in the next slide
Example 12 continued

```python
while True:
    try:
        i_num = int(input("Enter a number: "))
        if i_num < number:
            raise ValueTooSmallError
        elif i_num > number:
            raise ValueTooLargeError
        break
    except ValueTooSmallError:
        print("This value is too small, try again!")
        print()
    except ValueTooLargeError:
        print("This value is too large, try again!")
        print()
print("Congratulations! You guessed it correctly.")
```
Assert

assert <some_test>, <message>

Example 13:

def test_set_comparison():
    set1 = set("1308")
    set2 = set("8035")
    assert set1 == set2

test_set_comparison()

Output:

c:\users\necva\desktop> py deneme.py
traceback (most recent call last):
  file "deneme.py", line 8, in <module>
    test_set_comparison()
  file "deneme.py", line 4, in test_set_comparison
    assert set1 == set2
AssertionError
Example 14:

```python
array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

def number(input):
    assert (input in array)

number(10)
nmber(5)
```

Output:

```
C:\Users\neceva\Desktop>py deneme.py
Traceback (most recent call last):
  File "deneme.py", line 7, in <module>
    number(10)
  File "deneme.py", line 4, in number
    assert (input in array)
AssertionError
```
Example 15:

```python
def func(a, b):
    max = 0
    if a < b: max = b
    if b < a: max = a
    print(max)
    assert (max == a or max == b) and max >= a and max >= b

func(10, 15)
```

Output:

```
C:\Users\necva\Desktop>py deneme.py
15
```