



National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2022, Lab Manual – 01



Course Code: AI-2002	Course : Artificial Intelligence Lab
Instructor(s):	Kariz Kamal, Erum Shaheen, Mafaza Mohi, Danish Waseem, Ali Fatmi

Contents:

- | | |
|-----------------------------------|-----------------------------|
| I. Objective | a. Introduction & objective |
| II. Introduction to AI | b. Features of IDE |
| III. Python Programming Language | c. Python IDE and Editors |
| a. implementation Platform for AI | d. Jupyter v/s PyCharm |
| b. History of Python | V. Matlab Vs Python |
| c. Features of Python | VI. Installation Guide |
| d. Python Packages for AI | VII. Modules in Python |
| IV. IDE for Python | VIII. Tasks |

Objective

1. Introduction to Artificial Intelligence, exposing students to different AI Problems and solutions.
2. To acquire programming skills in Python, Introduction to different Python Libraries for Artificial Intelligence.
3. Introduction to different IDE for python programming.
4. Solve some basic AI problem using the python programming language.

Introduction to AI

The term Artificial Intelligence was first coined decades ago in the year 1956 by John McCarthy at the Dartmouth conference. He defined AI as: "The science and engineering of making intelligent machines." In other words, Artificial Intelligence is the science of getting machines to think and make decisions like humans. In the recent past, AI has been able to accomplish this by creating machines and robots that have been used in a wide range of fields including healthcare, robotics, marketing, business analytics and many more.

Implementation Platform for AI

Artificial Intelligence has been around for over half a century now and its advancements are growing at an exponential rate. The demand for AI is at its peak and this lab course on Artificial Intelligence with Python will help you understand all the concepts of AI with practical implementations in Python. Python has made its way into the most complex technologies such as Artificial Intelligence, Machine Learning, Deep Learning, and so on. According to Guido van Rossum

"Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressive-ness is endangered."

Python is the top programming language in TIOBE and PYPL Index. According to PYPL, which publishes separate ranking for five countries, Python is the top language in all five countries (US, India, Germany, United Kingdom, and France). Python has taken a huge lead in these five countries.

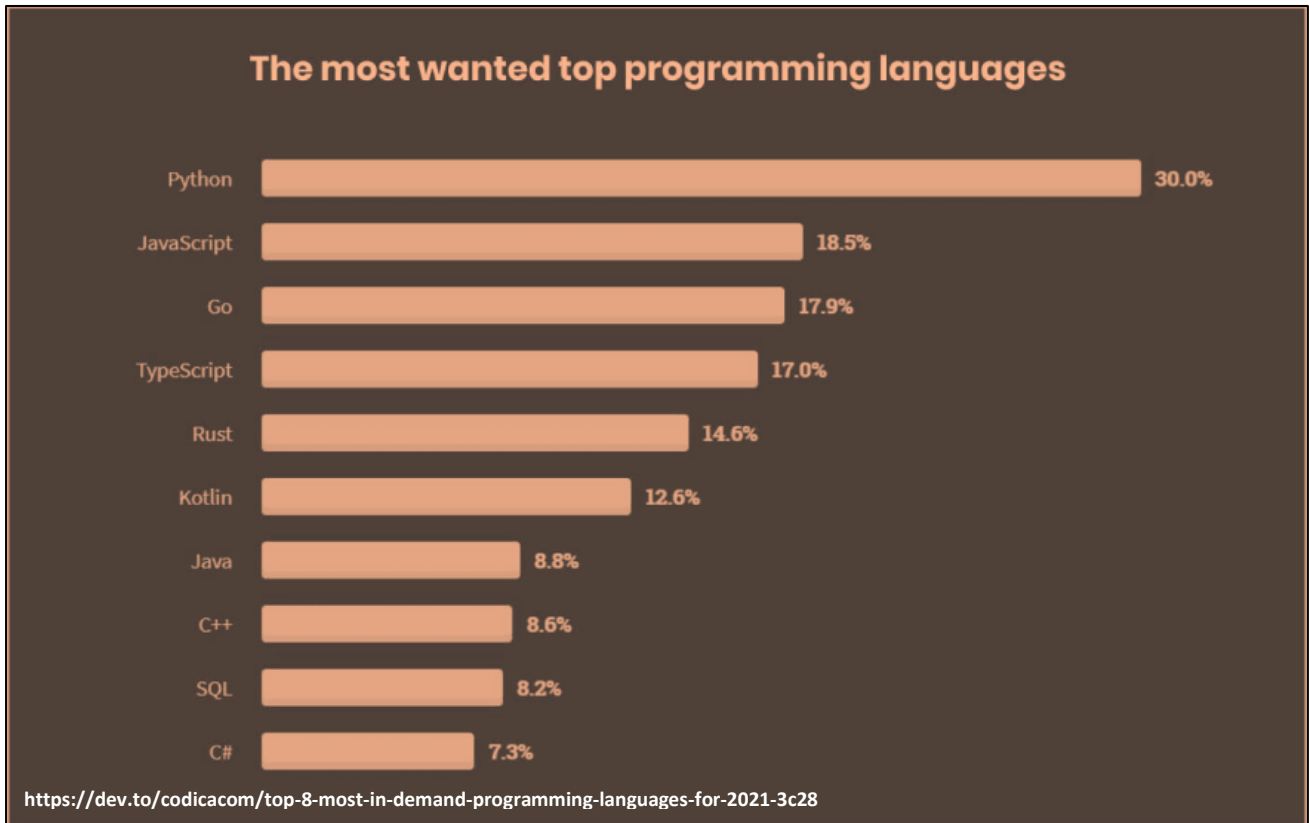


Figure 1. The most wanted programming Languages: Python overtakes JavaScript and GO and now is the most popular language, according to one popularity ranking.

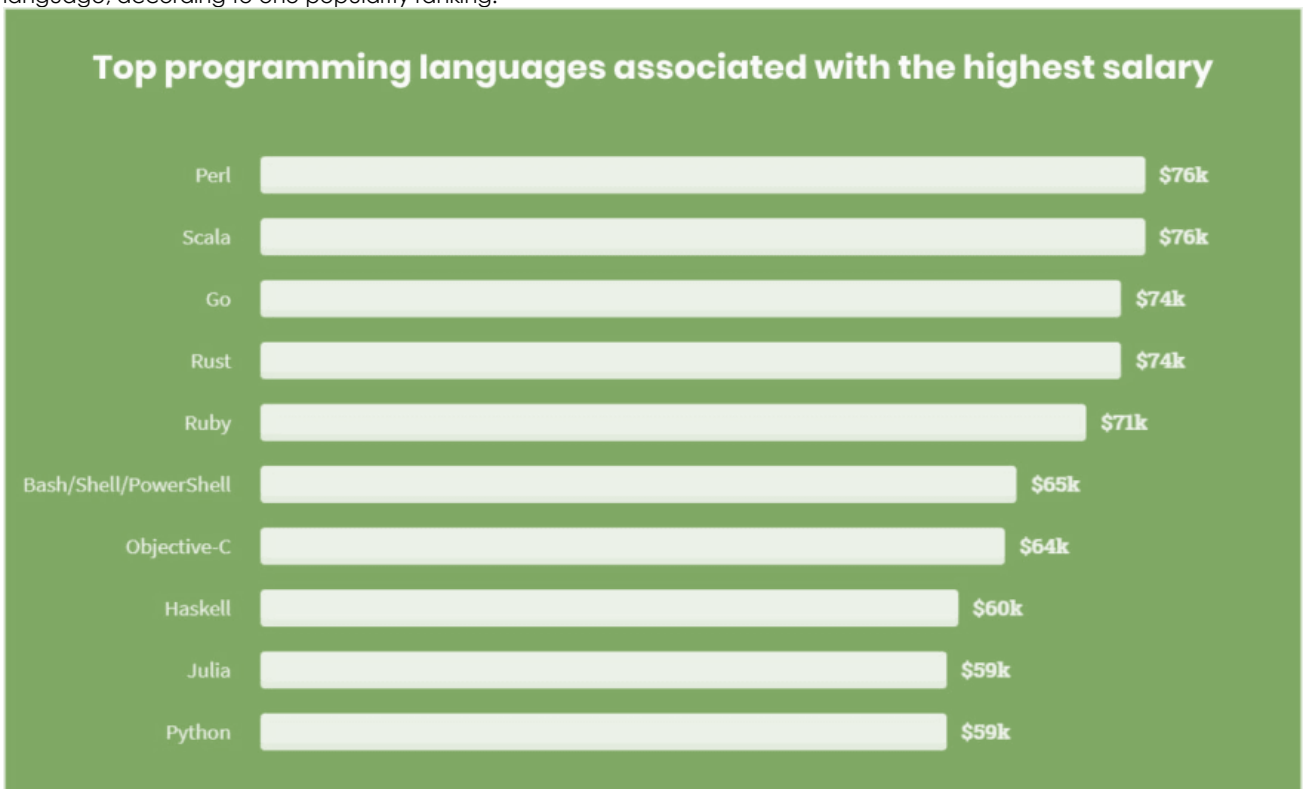


Figure 2. Top programming Languages with Highest Salary: As of Jan 23, 2022, the average annual pay for a Python Programmer in the United States is \$115,066 a year which is approximately \$55.32 an hour. This is the equivalent of \$2,213/week or \$9,589/month.

History of Python

- Invented in the Netherlands, early 90s by Guido van Rossum
- Named after Monty Python
- Open sourced from the beginning
- Considered a scripting language, but is much more
- Scalable, object oriented and functional from the beginning
- Used by Google from the beginning
- Increasingly popular

Distinct Features of Python

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, it's not necessary to declare the type of variable because it is a dynamically typed language.

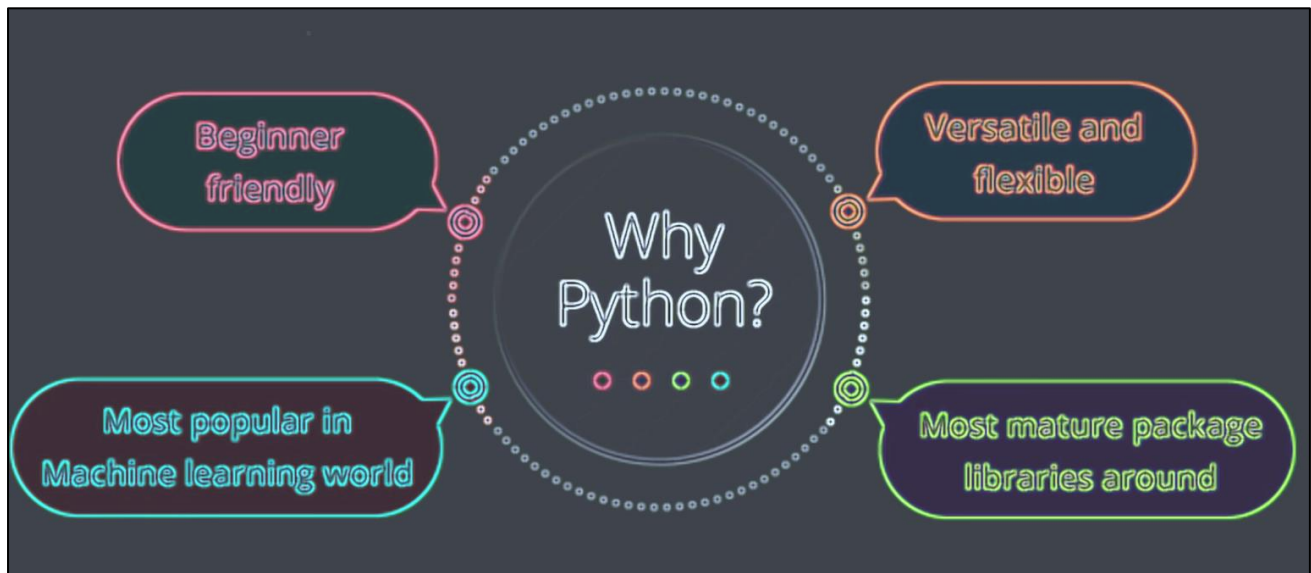


Figure 3. Features of Python: Python is one of the most dynamic and versatile programming languages available in the industry today.

- **Less Code:** Implementing AI involves tons and tons of algorithms. Thanks to Python's support for pre-defined packages, we don't have to code algorithms. And to make things easier, Python provides "check as you code" methodology that reduces the burden of testing the code.
- **Prebuilt Libraries:** Python has 100s of pre-built libraries to implement various Machine Learning and Deep Learning algorithms. So every time you want to run an algorithm on a data set, all you have to do is install and load the necessary packages with a single command. Examples of pre-built libraries include NumPy, Keras, Tensorflow, Pytorch, and so on.
- **Ease of learning:** Python uses a very simple syntax that can be used to implement simple computations like, the addition of two strings to complex processes such as building a Machine Learning model.
- **Platform Independent:** Python can run on multiple platforms including Windows, MacOS, Linux, Unix, and so on. While transferring code from one platform to the other you can make use of packages such as PyInstaller that will take care of any dependency issues.
- **Massive Community Support:** Python has a huge community of users which is always helpful when we encounter coding errors. Apart from a huge fan following, Python has

multiple communities, groups, and forums where programmers post their errors and help each other.

Most Popular Python Packages for AI

- **Tensorflow** this library was developed by Google in collaboration with brain team. Tensorflow is used in almost every Google application for machine learning. With tensorflow we can easily visualize each and every part of the graph which is not an option when you are using other packages such as numpy or scipy, another feature is that it's very flexible, one of the most important tensorflow features is that it is flexible in operability meaning that it has modularity. A good feature about tensorflow is that you can train it on both CPU and GPU and so for distributed computing you can have both these options also it supports parallel neural network training so tensorflow offers pipelining in the sense that you can train multiple neural networks and multiple GPUs which makes the models very efficient on any large scale system.
- **Cyclone** is a Python library that is associated with numpy and scipy. That's why it has a name cyclone. This is considered to be one of the best libraries for working with complex data there are a lot of changes that are being made in this library and one modification is the cross validation feature which provides the ability to use more than one metric. cross validation is one of the most important and one of the most easiest methods for checking the accuracy of a model so cross validation has been implemented in cyclone and apart from that again there are large spread of algorithms that you can implement by using cyclotron, these include unsupervised learning algorithms starting from clustering, factor analysis, principal component analysis, although unsupervised neural networks cyclone is also very essential for feature extracting in images and text so mainly cyclone is used for implementing all the standard machine learning and data mining tasks like reducing dimensionality classification regression clustering and model selection.
- **numpy** is considered as one of the most popular machine learning libraries of python. Tensorflow and other libraries they make use of numpy internally for performing multiple operations on tensors. the most important feature of numpy is the array interface it supports multi-dimensional arrays. another feature is it makes complex mathematical implementations very simple. it's mainly known for computing mathematical data so numpy is a package that you should be using for any sort of statistical analysis or data analysis that involves a lot of math, apart from that it makes coding very easy and grasping the concept is extremely easy been done by now numpy is mainly used for expressing images sound waves and other mathematical computations.
- **theano** is a computational framework which is used for computing multi-dimensional arrays it works very similar to tensorflow but the only drawback is that you can't fit theano into production environments. theano allows you to define optimize and evaluate mathematical expressions that involves multi-dimensional arrays and this is another library that lets you implement multi-dimensional arrays. theano was actually designed to handle the types of computations that are required for large neural network algorithms that it was mainly built for deep learning and neural networks it was one of the first libraries of its kind and it is considered as an industry standard for deep learning research and development theano is being used in multiple neural networks projects.

- **Karas** is considered to be the most popular Python package. it provides some of the best functionalities for compiling models processing datasets and visualizing graphs. It is also popular in the implementation of neural networks. It is considered to be the simplest package with which you can implement neural networks. it runs very smoothly on both CPU and GPU it supports almost all the models of a neural network, from fully connected convolutional pooling, recurrent embedding, all of these models are supported by Karas and not only that you can combine these models to build more complex models. Karas is completely python based which makes it very easy to debug and explore since Python has a huge community of followers it's very simple in order to debug any sort of error that you find while implementing Karas.
- **Natural language toolkit** NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Example:

```
import numpy as np

# Original array
array = np.arange(10)
print(array)

r1 = np.mean(array)
print("\nMean: ", r1)

r2 = np.std(array)
print("\nstd: ", r2)

r3 = np.var(array)
print("\nvariance: ", r3)
```

```
# importing networkx
import networkx as nx
# importing matplotlib.pyplot
import matplotlib.pyplot as plt
g = nx.Graph()
g.add_edge(1, 2)
g.add_edge(2, 3)
g.add_edge(3, 4)
g.add_edge(1, 4)
g.add_edge(1, 5)
g.add_edge(5, 6)
g.add_edge(5, 7)
g.add_edge(4, 8)
g.add_edge(3, 8)
# drawing in circular layout
nx.draw_circular(g, with_labels = True)
plt.show(g)
```

IDE for Python

An integrated development environment (IDE) is a software suite that consolidates basic tools required to write and test software

Objectives of an IDE

- Faster Setup
- Faster development tasks
- Continual learning
- Standardization

Features of an IDE

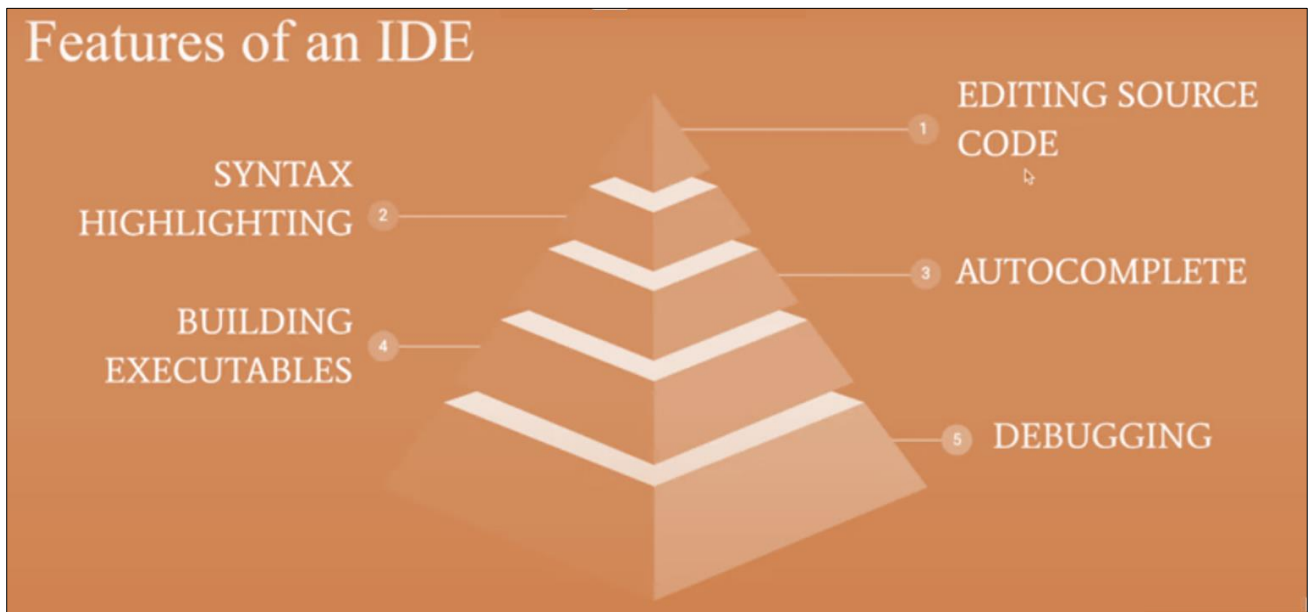


Figure 4. Features of IDE: IDE provides an editor, compiler, and debugger and usually performs tasks such as code completion and generic code management.

Most Popular IDEs for Python



Figure 5. IDE for Python: Python IDEs and code editors for beginners and professionals.

Python DIE and Editors

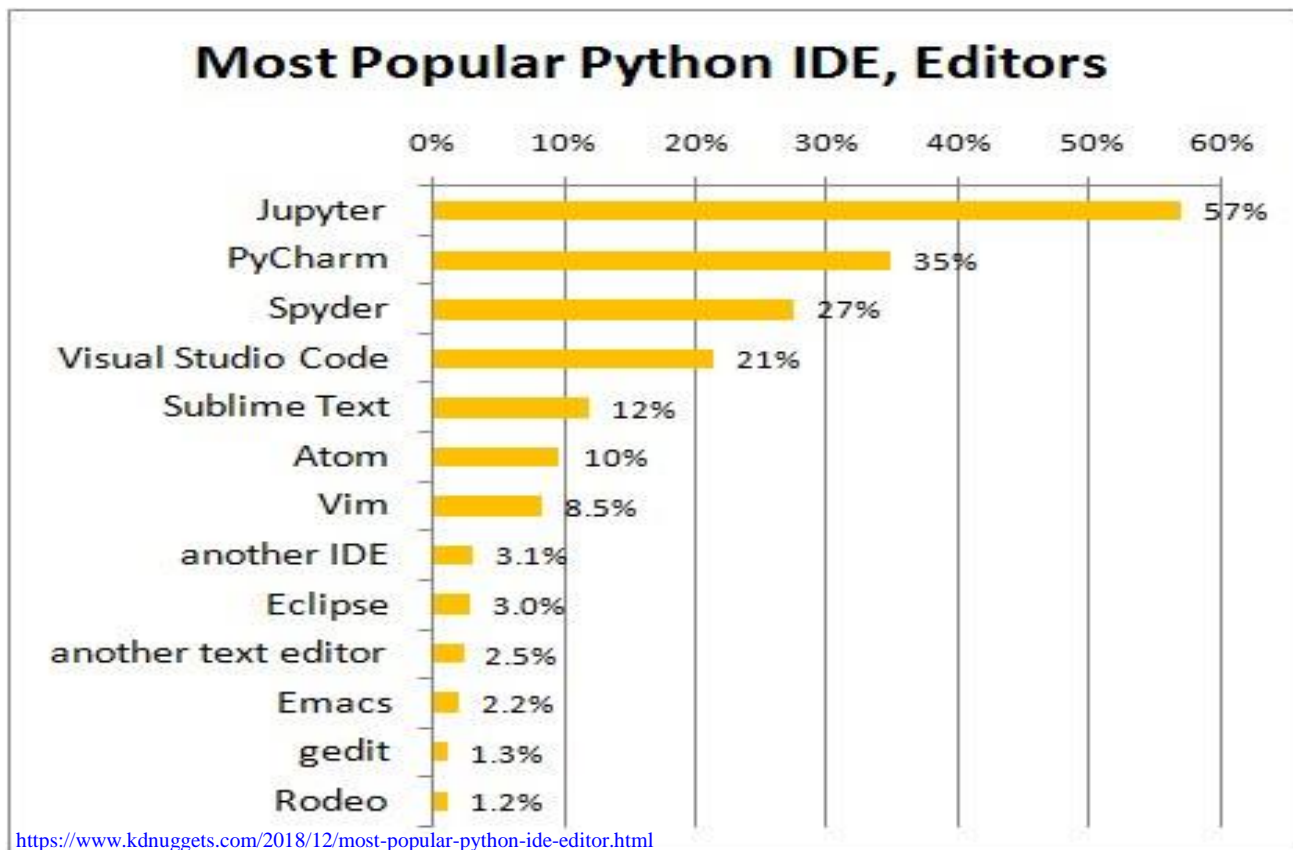


Figure 6. Most Popular Python IDE, Editors.

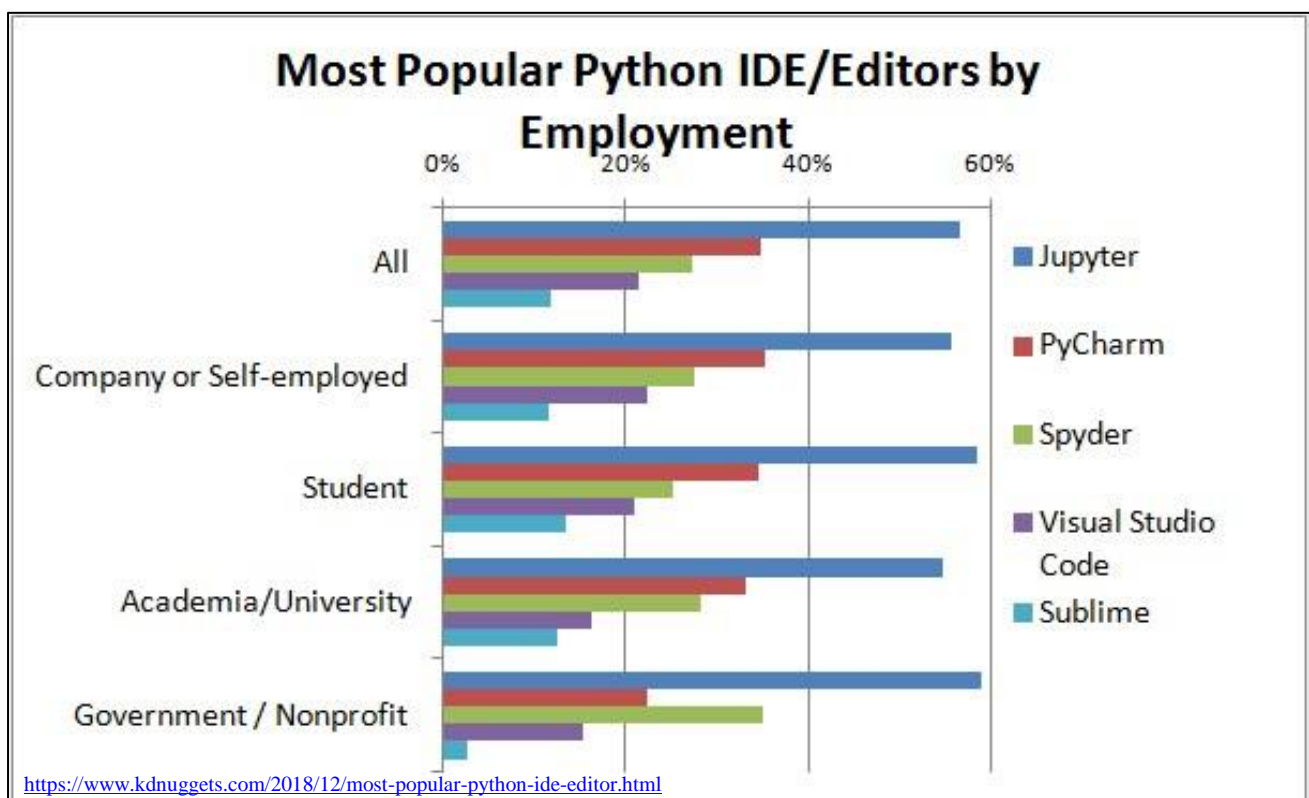
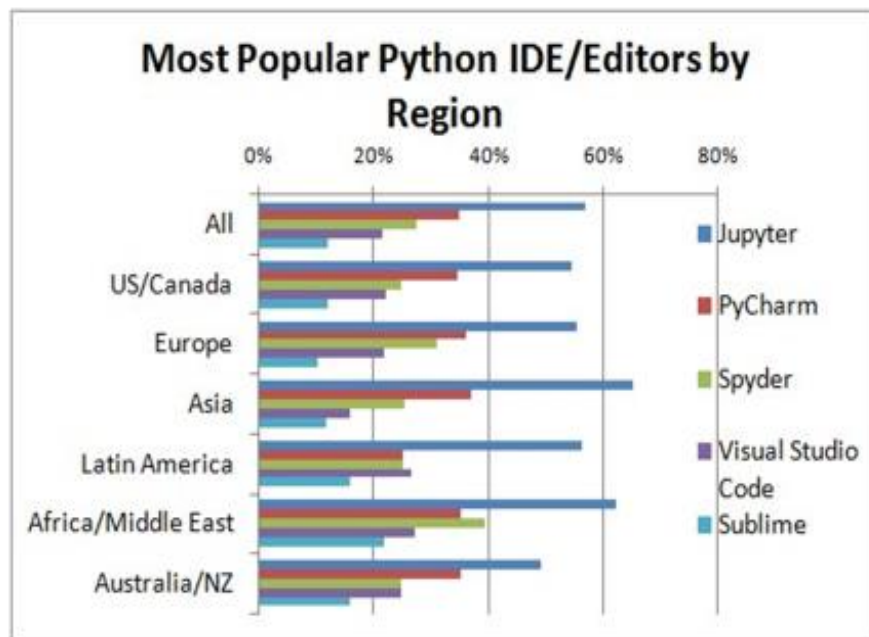


Figure 7. Most Popular Python IDE, Editors by Employment



<https://www.kdnuggets.com/2018/12/most-popular-python-ide-editor.html>

©Dr. Muhammad_Farrukh_Shahid_2021

15

Figure 8. Most Popular Python IDE, Editors by Region

Jupyter v/s PyCharm





				
	Features	Spyder	Jupyter Notebook	Google Colab
1	Installation	Yes	Yes	No
2	Launch	GUI	Web browser	Web browser
3	Internet Connection	No	No	Yes
4	Usage	Development	Data Science	Data Science
5	Visualisation	NA	Yes	Yes
6	Data	Local/Remote	Local/Remote	Google Drive
7	Power	CPU	CPU	CPU/GPU/TPU

Figure 9. Comparison Table of different Python IDE

<https://www.youtube.com/watch?v=gbvUbmvirFY>

Matlab Vs Python

Matlab	Python
It is an interactive and object-oriented programming language like PERL or Ruby. It is mainly designed to be easy to read and very simple to implement. It is open source, which means it is free to use. Python can run on all the operating systems. Python was developed by Guido van Rossum and launched in 1991. Besides its neat syntax and code readability features, Python also comes geared up with a number of ordinary libraries for conducting totally different programming and computing tasks.	It stands for Matrix Laboratory. In the late 1970s, Cleve Moler began the development of MATLAB. Developed by MathWorks, it offers a multi-paradigm numerical computing environment that combines a desktop environment tuned for iterative analysis and design processes with a programming language that directly expresses matrix and array mathematics. Its Live Editor includes creating scripts that combine code, output, and formatted text in an executable notebook.

MATLAB:

MATLAB is a closed-source software program and a proprietary commercial product. This implies one must pay a tremendous amount of money to use MATLAB. In contrast, Python is an open-source programming language, which means it is totally free. One just has to download and install Python and make alterations to the source code to obtain results. Next, MATLAB has an integrating development environment. It is a neat interface with a console located in the center where the user can type commands, while a variable explorer lies on the right and a directory listing on the left. Unfortunately, Python does not include a default development environment. Users need to choose an IDE that fits their requirement specifications. While MATLAB uses end statements as closures (i.e., to indicate the end of a block), Python determines a block's scope based on indentation. This forces Python programmers to indent code blocks. Further, Python uses square brackets for indexing and parentheses for function and method calls.

MATLAB uses parentheses for both. Python's use of square brackets for indexing is important for readability and makes life easier for programmers who must work with multiple languages. MATLAB's use of parentheses for both indexing and function calls makes it hard to understand and is a frequent bug source. Python also uses zero-based indexing, which is better than the one-based indexing of MATLAB. Moreover, Python supports augmented assignments and unrestricted use of literals in expressions.

Python:

Python is best suited for web programming, whereas MATLAB allows matrix manipulations, plotting functions and data, and creating user interfaces. In engineering, Python also helps carry simulation, vibrations, engineering modeling, and dynamic motion. Meanwhile, the IC toolbox for image processing in MATLAB makes it a better option for image data segmentation, extraction, and analysis. But in Python, image processing relies on external packages. Python's Numpy and Scipy libraries provide lots of algorithms for image processing. Results in Python are fundamentally the same as indicating that the statsmodels OLS functions are exceptionally advanced. Then again, MATLAB shows huge speed upgrades and exhibits how local direct variable based math code is favored for speed. For this model, MATLAB is around multiple times faster than Python.

Installation Guide

1. Anaconda jupyter notebook
2. Google Colab

Note: Installation guide is provided in different document.

Modules in Python

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a file consisting of Python code. A module can define functions, classes and variables. In Python, modules are accessed by using the 'import' statement. When you do this, you execute the code of the module, keeping the scopes of the definitions so that your current files can make use of these. Many build-in modules of python, some of above

- | | |
|---------------|-------------|
| I. Math | IV. Decimal |
| II. Random | V. OS |
| III. Fraction | |

1. Basic Arithmetic operation & Python Libraries

```
#CODE# 01 take three values from user
a = 12
b = 2
C = 5
d = -3
print("Addition:", a+b)
print("Subtraction:", a-b)
print("Multiplication:", a*b)
print("Division :", a/b)
print("Power : ", a**b)
print("Modulus/Remainder:", a%C)
print("Absolute:", abs(d))
```

```
#CODE# 02 take three values from user
import numpy as np
# Mathematical constants
print(np.pi)
print(np.e)
# Trigonometric Functions
angle = np.pi/4
print(np.sin(angle))
print(np.cos(angle))
print(np.tan(angle))
```

2. Input() Function

```
# CODE# 3 take three values from user
name = input("Enter Employee Name: ")
salary = input("Enter salary: ")
company = input("Enter Company name: ")

# Display all values on screen
print("\n")
print("Printing Employee Details")
print("Name", "Salary", "Company")
print(name, salary, company)
```

```
# CODE# 04 list to store multi line input
# press enter two times to exit
data = []
print("Tell me about yourself")
while True:
    line = input()
    if line:
        data.append(line)
    else:
        break
finalText = '\n'.join(data)
print("\n")
print("Final text input")
print(finalText)
```

3. STRINGS and LIST

CODE# 5 working with strings

```
var1 = 'Hello World!'
var2 = "Python Programming"
print ("var1[0]: ", var1[0])
print ( "var2[1:5]: ", var2[1:5])
print ("Complete ",var1)
```

CODE# 6 working with List

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
print (list1)
```

4. Functions**a. Built-in functions****b. User-defined functions**

CODE# 7 user Defined Function

Function definition is here

```
def printme( str ):
    "This prints a passed string into this function"
    print (str)
    return;
```

Now you can call printme function

```
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

CODE# 8 user Defined Function

Function definition is here

```
def changeme( mylist ):
    "This changes a passed list into this function"
    mylist.append([1,2,3,4]);
    print ("Values inside the function: ", mylist)
    return
```

Now you can call changeme function

```
mylist = [10,20,30];
print ("old Values outside the function: ", mylist)
changeme( mylist );
print ("new Values outside the function: ", mylist)
```

CODE# 09 Built-in Function

```
a = 12
b = -4
c = 3+4j
d = 7.90
print(abs(a))
print(abs(b))
print(abs(c))
print(abs(d))
```

CODE# 10 Built-in Function

```
stringobj = 'PALINDROME'
print(len(stringobj))
tupleobj = ('a', 'e', 'i', 'o', 'u')
print(len(tupleobj))
listobj = ['1', '2', '3', 'o', '10u']
print(len(listobj))
```

5. TUPLES

CODE# 11 working with Tuples

```
tup1 = ('physics', 'chemistry', 1997,
        2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print ("tup1[0]: ", tup1[0]);
print ("tup2[1:5]: ", tup2[1:5]);
print(tup1)
```

CODE# 12 working with Tuples

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# Following action is not valid for t
 uples
# tup1[0] = 100;
tup3 = tup1 + tup2;
print (tup3);
```

6. DICTIONARY

CODE# 13 working with dictionary

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School";

print ("dict['Age']: ", dict['Age'])
print ("dict['School']: ", dict['School'])
```

CODE# 14 working with dictionary

```
dict = {'Name': 'Zara', 'Age': 7,
        'Class': 'First'}
print ("dict['Name']: ", dict['Name'])
print ("dict['Age']: ", dict['Age'])
```

7. Sequential STATEMENTS

CODE# 13 working with Loops

```
count = 0
while count < 5:
    print (count, " is less than 5")
    count = count + 1
else:
    print (count, " is not less than 5")
```

CODE# 14 working with Loops
for letter in 'Python': # First Example
 print ('Current Letter:', letter)

```
fruits = ['banana', 'apple', 'mango']
for fruit in fruits: # Second Example
    print ('Current fruit:', fruit)
print ("Good bye!")
```

8. CONDITIONAL STATEMENTS

CODE# 15 working with if

```
var1 = 100
if var1:
    print ("1 - Got a true expression value")
    print (var1)
var2 = 0
if var2:
    print ("2 - Got a true expression value")
    print (var2)
print ("Good bye!")
```

CODE# 15 working with if else

```
var1 = 100
if var1:
    print ("1 - Got a true expression value")
    print (var1)
else:
    print ("1 - Got a false expression value")
    print (var1)

var2 = 0
if var2:
    print ("2 - Got a true expression value")
    print (var2)
else:
    print ("2 - Got a false expression value")
    print (var2)
print ("Good bye!")
```

TASKS

Task#01

Consider an *Interactive Cognitive Environment (ICE)* in which autonomous robot is performing cleaning task in the big room that appears to be a matrix of $N * M$. Each index referred to as a cell of the matrix is valued as dirty "D" or clean "C". The cells which are occupied by the stuff in the room are blocked and valued "B". The vacuum can move in all four directions (up, down, left, right), and if the cell status is D, it will clean the cell and change the status to "C", if the cell status is either C, it will not enter the cell. The vacuum will stop working if the whole room is cleaned, i.e., the status of all the cells is either C. The vacuum may start cleaning the room from the first cell (0, 0) or any random location. You will trace the path of the vacuum and display at each step of the program. * Represent the location of the vacuum cleaner. Develop a Python code of the above describe scenario of the autonomous robot.

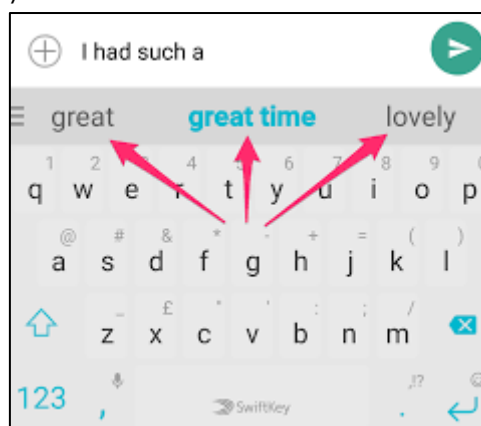
D*	D	D	D	D	D	D													
			C	D															
D	D	C	D			D	D	D											
D	D	D	D	C	C	C	C	D											
D	D	D	D	D	D	D	D	D											
D	D	D	D	D	D	D	D	D											
D	D	D				D	D	D											

If vacuum is in a location where it's all neighbors (up, down, left and right) are clean (with status C) it will move in any one of the directions and keep searching the Dirt (D). It will stop its execution if it does not sense any dirt after 10 movements.

If vacuum is in a location where it's one more neighbor (up, down, left and right) is dirty it will move in any one of the directions and will return to the location when it cleans all the dirty cell of its neighbors. e.g., cell (0, 3) where it's three neighbors are dirty.

Task#02

Next Word Prediction or what is also called Language Modeling is the task of predicting what word comes next. It is one of the fundamental tasks of NLP and has many applications. You might be using it daily when you write texts or emails without realizing it.



Your devices or applications keep the track of your writing style or text in memory and apply different algorithm and models. These models break the paragraph/text into group words e.g. group may contain 1 to N word based on the model. Your task is to build the

same program for breaking the texts/paragraph in to group of words and counts which group of words occurring together in text.

e.g.

Sentence = 'the purpose of our life is to happy'

('The', 'purpose')

('Purpose', 'of')

('Of', 'our')

('Our', 'life')

('Life', 'is')

('is', 'to')

('To', 'happy')

Sentence = 'Whoever is happy will make others happy too'

('Whoever', 'is', 'happy')

('is', 'happy', 'will')

('happy', 'will', 'make')

('will', 'make', 'others')

('make', 'others', 'happy')

('others', 'happy', 'too')