PYTHON NOTES:

About python:

Python is an open source, high-level programming language developed by Guido van Rossum in the late 1980s and presently administered by Python Software Foundation. It came from the ABC language that he helped create early on in his career.

Python is a powerful language that you can use to create games, write GUIs, and develop web applications. It is a high-level language. Reading and writing codes in Python is much like reading and writing regular English statements. Because they are not written in machine-readable language, Python programs need to be processed before machines can run them.

Python is an interpreted language. This means that every time a program is run, its interpreter runs through the code and translates it into machine-readable byte code.

Python is an object-oriented language that allows users to manage and control data structures or objects to create and run programs. Everything in Python is, in fact, first class. All objects, data types, functions, methods, and classes take equal position in Python.

Why python:

Here are reasons why you would prefer to learn and use Python over other high level languages:

1. Readability: Python programs use clear, simple, and concise instructions that are easy to read even by those who have no substantial programming background. Programs written in Python are, therefore, easier to maintain, debug, or enhance.
2. Higher productivity: Codes used in Python are considerably shorter, simpler, and less verbose than other high level programming languages such as Java and C++. In addition, it has well-designed built-in features and standard library as well as access to third party modules and source libraries. These features make programming in Python more efficient.
3. Less learning time: Python is relatively easy to learn. Many find Python a good first language for learning programming because it uses simple syntax and shorter codes.
4. Runs across different platforms: Python works on Windows, Linux/UNIX, Mac OS X, other operating systems and smallform devices. It also runs on microcontrollers used in appliances, toys, remote controls, embedded devices, and other similar devices.

**Installing Python in Windows**

To install Python, you must first download the installation package of your preferred

version from this link:

<https://www.python.org/downloads/>

On this page, you will be asked to choose between the two latest versions for Python 2 and

3: Python 3.7.4 and Python 2.7.11. Alternatively, if you are looking for a specific release,

you can scroll down the page to find download links for earlier versions.

**Open Python’s command line:**

>> print(“Hello, World!”)

**Exiting Python:**

>>quit()

>>exit()

>>Control-Z then press enter

**IDLE: Python’s Integrated Development Environment (IDE)**

The IDLE (Integrated Development and Learning Environment) tool is included in Python’s installation package

**Just type on search IDLE**

but you can choose to download more sophisticated third party IDEs.example : **Python Shell Window, eclipse, visual studio code (VS code) , pycharm, spyder**

**Keywords:** Python keywords are reserved words in Python that should not be used as variable, constant, function name, or identifier in your code.

and , assert , break, class ,continue ,def ,del ,elif ,else ,except, exec finally ,for ,from ,global, if ,import ,in ,is ,lambda ,not, or ,pass, print raise, return ,try ,while,with ,yield

**Python Identifiers:** A Python Identifier is a name given to a function, class, variable, module, or other objects that you’ll be using in your Python program.

* An identifier can be a combination of uppercase letters, lowercase letters, underscores,

and digits (0-9). Hence, the following are valid identifiers: myClass, my\_variable,

var\_1, and print\_hello\_world.

* Special characters such as %, @, and $ are not allowed within identifiers.
* An identifier should not begin with a number. Hence, 2variable is not valid, but

variable2 is acceptable.

* Python is a case-sensitive language and this behavior extends to identifiers. Thus,

Labor and labor are two distinct identifiers in Python.

* You cannot use Python keywords as identifiers.
* Class identifiers begin with an uppercase letter, but the rest of the identifiers begin in

lowercase. You can use underscores to separate multiple words in your identifier.

List =[2,4,5]

**Comments:**

# symbol.

For multiline comments use ‘’’amir’’’

**Variables:** A variable is like a container that stores values that you can access or change.

>>my\_variable = 10

>ltiline comments use amirhird

**To increase the value of the variable**

>>>my\_variable = my\_variable + 3

To print the value of the variable

>>>print(my\_variable)

**Data Types:**

**Numeric Data Types:**

1. ***Integer (int)*** Integers are whole numbers without decimal point. They can be positive or negative as long as they don’t contain a decimal point that would make a number a floating number, a distinct numeric type. Integers have unlimited size in Python 3.

Examples: 793, -254, 4

To see the object type created and stored in the variable x, you can use and enter the

command

for e.g:

x=1

>> **type(x)**

1. ***Floating-point numbers:*** Also known as floats, floating-point numbers signify real numbers. Floats are written with a decimal point that segregates the integer from the fractional numbers

>>>6.2e3

6200.0

**3.**  ***Complex numbers:*** Complex numbers are pairs of real and imaginary numbers. They take the form ‘a + bJ’ where ‘a’ is a float and the real part of the complex number. On the other side is bJ where ‘b’ is a float and J or its lowercase indicates the square root of an imaginary number, -1. This makes ‘b’ the imaginary part of the complex number.

>>>a = 2 + 5j

>>>b = 4 – 2j

>>>c = a + b

>>>print(c)

(6 + 3j)

**Conversion of Number Type:**

a=”50”

a=int(a)

……………….

a= 4.50

a=int(a)

……………..

a=(“4.54”)

a=int(a)

…………….

**The input() Function**

print("enter a number1 ")

print("enter a number2 ")

a=input()

b=input()

a and b are always string because input() function always take string

therefor it will be type casted if used as interger

#a=int(input())

#b=int(input())

**The print() Function**

a=1

b=2

c=3

print("number", a, b, c)

print(“number = “, a+b+c)

**Conditional Statements**

if condition1:

block1\_statement

elif condition2:

block2\_statament

else:

block3\_statement

if a>b:

a=a+b

print(a)

else:

b=a-b

print(b)

**Loops in PYTHON:**

1. **While Loop:**

while expression:

statement(s)

count = 0

while (count < 3):

     count = count + 1

     print("Hello Geek")

while else

count = 0

while (count < 3):

    count = count + 1

    print("Hello Geek")

else:

    print("In Else Block")

1. **For Loop:**

***Using the range() Function with the for Loop***

for iterator\_var in sequence:

statements(s)

e.g:

i=9

for a in range(i):

print("\*")

i=9

for a in range(1,i):

print("\*")

***Using variable in list, tuple, dictionary for Loop***

for variable in list:

statements

else:

statements

pizza = [“New York Style Pizza”, “Pan Pizza”, “Thin n Crispy Pizza”, “Stuffed Crust Pizza”]

for choice in pizza:

if choice == “Pan Pizza”:

print(“Please pay $16. Thank you!”)

print(“Delicious, cheesy ” + choice)

else:

print(“Cheesy pan pizza is my all-time favorite!”)

print(“Finally, I’m full!”)

**# Iterating over a list**

print("List Iteration")

l = ["geeks", "for", "geeks"]

for i in l:

    print(i)

**# Iterating over a tuple (immutable)**

print("\nTuple Iteration")

t = ("geeks", "for", "geeks")

for i in t:

    print(i)

**# Iterating over a String**

print("\nString Iteration")

s = "Geeks"

for i in s :

    print(i)

**# Iterating over dictionary**

print("\nDictionary Iteration")

d = dict()

d['xyz'] = 123

d['abc'] = 345

for i in d :

    print("%s  %d" %(i, d[i]))

**Iterating by index of sequences**:

# Python program to illustrate

# Iterating by index

list = ["geeks", "for", "geeks"]

for index in range(len(lis)):

print list[index]

**Using else statement with for loops:**

# Python program to illustrate

# combining else with for

list = ["geeks", "for", "geeks"]

for index in range(len(list)):

    print list[index]

else:

    print "Inside Else Block"

**Nested Loops:**

for iterator\_var in sequence:

    for iterator\_var in sequence:

        statements(s)

        statements(s)

**Loop Control Statements:**

# Prints all letters except 'e' and 's'

for letter in 'geeksforgeeks':

    if letter == 'e' or letter == 's':

         continue

    print 'Current Letter :', letter

    var = 10

**Break Statement:**

for letter in 'geeksforgeeks':

   # break the loop as soon it sees 'e'

   # or 's'

   if letter == 'e' or letter == 's':

         break

print 'Current Letter :', letter

**Pass Statement:**

# An empty loop

for letter in 'geeksforgeeks':

    pass

print 'Last Letter :', letter