

PYTHON-PROGRAMMING

YNM

What is Python?

- Python is a general purpose interpreted interactive object oriented and high level programming language.
- It was first introduced in 1991 by Guido van Rossum , a Dutch computer programmer.
- The language places strong emphasis on code reliability and simplicity so that the programmers can develop applications rapidly

contd..

- Python is multi-paradigm programming language ,which allows user to code in several different programming styles.
- Python supports cross platform development and is available through open source.
- Python is widely used for scripting in Game menu applications effectively.

contd..

- Python can be easily integrated with C/C++ CORBA, ActiveX and Java.
- CPython is a python integrated with C/C++ language.
- Similarly JPython is a purely integrated language where Java and Python code can interchangeably use inline in the program.

Advantages of Python

- Most programs in Python require considerably less number of lines of code to perform the same task compared to other languages like C .So less programming errors and reduces the development time needed also.
- Though Perl is a powerful language ,it is highly syntax oriented .Similarly C also.

Why to Learn Python

- There are large number of high-level programming languages like C ,C++,Java etc.But when compared to all these languages Python has simplest Syntax, availability of libraries and built-in modules.
- Python comes with an extensive collection of third party resources that extend the capabilities of the language.

contd..

- Python can be used for large variety of tasks like Desktop applications ,Data base applications,network programming ,game programming and even mobile development also.
- Python is also a cross platform language which means that the code written for one operating system like Windows ,will work equally well with Linux or MacOS without any changes to the python code.

INSTALLING PYTHON ON YOUR PC

- To learn this language first ,you have to download the software which is the Python Interpreter.
- Presently the version in use is Python 3.x

But you can also use Python 2.x ,like 2.7..etc .This depends on your system and your interest.

To download the Python interpreter go to the site <http://www.python.org/downloads> and click on the suitable icon.

contd..

- The website appears as shown below.



[Download Python 3.5.0](#) [Download Python 2.7.10](#)

[Here's more about the difference between Python 2 and 3.](#)

[Windows, Linux/UNIX](#) [Mac OS X](#) [Other](#)

[Pre-releases](#)

[Here's more about the difference between Python 2 and 3.](#)

[Windows, Linux/UNIX](#) [Mac OS X](#) [Other](#)

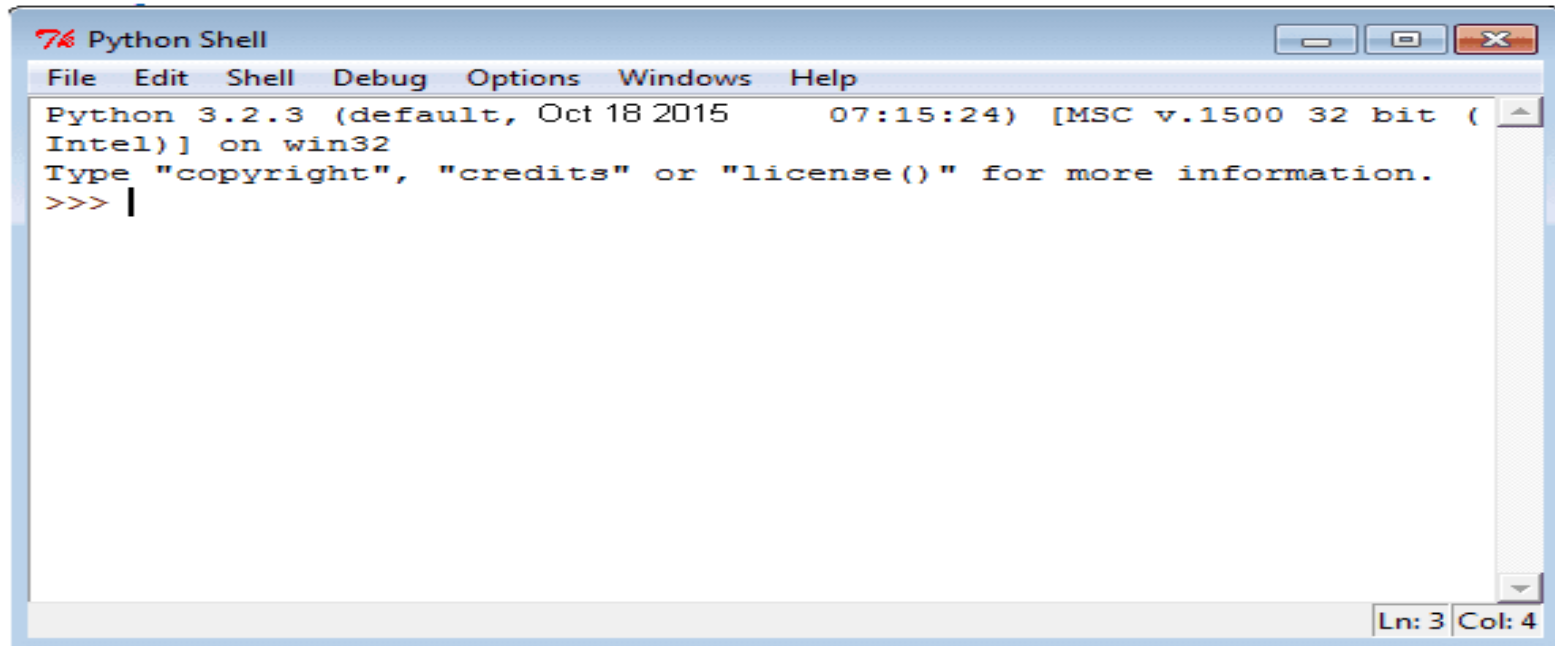
[Pre-releases](#)

contd..

- From the website you can download the suitable version based on
- your OS (whether the OS is Windows ,MacOS or Linux).
- And the Processor 32 bit or 64 bit.
- For ex: if you are using 64-bit Windows system ,you are likely to download Windowsx86-64MSI Installer.
- So just click on the link and download and install the Python Interpreter.

contd..

- This Python shell allows us to use Python in interactive mode.
- The shell waits for a command from the user ,executes it and returns the result.



The image shows a screenshot of a 'Python Shell' window. The title bar is blue with a red Python logo and the text 'Python Shell'. Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The main text area is white and contains the following text: 'Python 3.2.3 (default, Oct 18 2015 07:15:24) [MSC v.1500 32 bit (Intel)] on win32', 'Type "copyright", "credits" or "license()" for more information.', and the prompt '>>> |'. The status bar at the bottom right shows 'Ln: 3 Col: 4'.

```
Python 3.2.3 (default, Oct 18 2015 07:15:24) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

IDLE

- IDLE is a graphical user interface for doing Python development, and is a standard and free part of the Python system.
- It is usually referred to as an Integrated Development Environment (IDE).
- One can write the Python code or script using the IDLE which is like an editor. This comes along with Python bundle.

contd..

The only thing to do is that we have to launch the IDLE program .

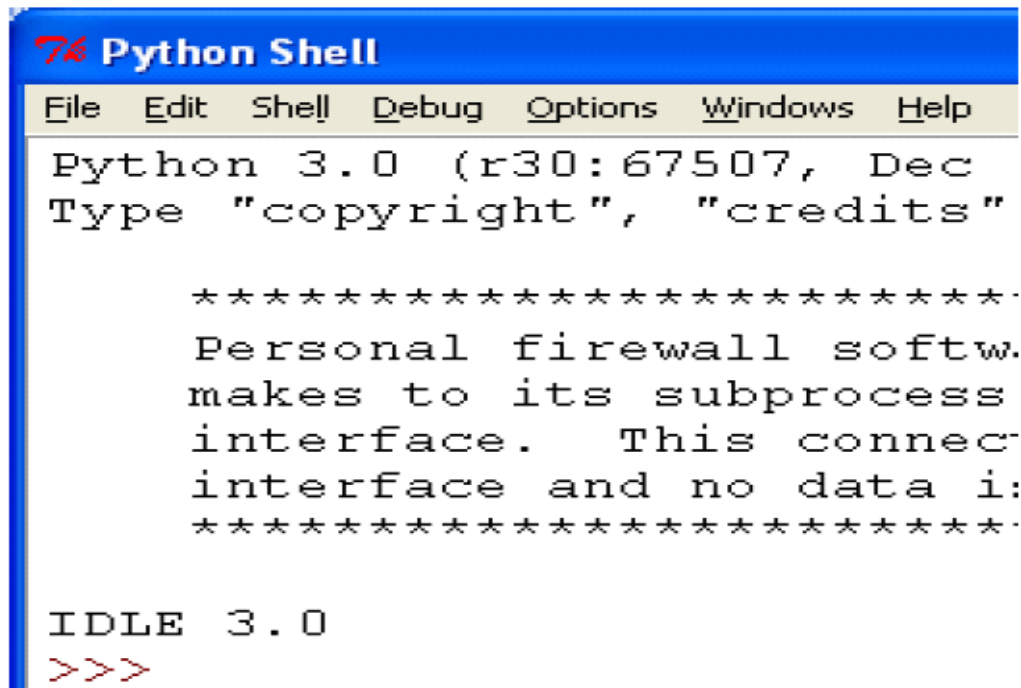
- While the Python shell allows the user to work in interactive mode, the Idle allows to write the code and save it with .py extension. For Ex: myProg_py.
- This file is known as Python script.

contd..

- Once the program is saved ,it can be executed either by clicking the Run module or F5.
- If there are no errors you observe the results.

contd..

- The Screenshot for IDLE will be as shown below. The >>> is the *prompt*, telling you Idle is waiting for you to type something.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.0 (r30:67507, Dec
Type "copyright", "credits"

*****
Personal firewall softw.
makes to its subprocess
interface. This connec-
interface and no data i:
*****

IDLE 3.0
>>>
```

Your First Program

- To develop the Python program ,click on the File and select NewFile.
- This will open a new text editor where you can write your first program.
- # Prints the words Hello Python
`print("Hello Python")`
`print("Its nice learning Python")`
`print("Python is easy to learn")`

contd..

- In the above program ,the lines appear after the # sign are treated as comments. They are ignored and not executed by Python interpreter. This will help only to improve the readability of the program.
- After typing the code save this program with an extension .py .For Ex: FirstProg.py
- Now press either F5 or click on Run module.

contd..

- After execution you find the output as
Hello Python
Its nice learning Python
Python is easy to learn.

Variables and Operators

- Variables are the names given to data that we need to store and manipulate in our program.
- For example ,to store the age of the user ,we can write a variable `userAge` and it is defined as below.

`userAge = 0` or `userAge = 50`

After we define `userAge` ,the program will allocate some memory to store this data.

contd..

- Afterwards this variable can be accessed by referring its name **userAge**.
- Every time we declare a variable ,some initial value must be given to it.
- We can also define multiple variables at a time.

For ex: userAge, userName=30,'Peter'

This is same as userAge=30
userName='Peter'

contd..

- **Some precautions:** The user name can contain any letters (lower case or upper case) numbers or underscores(-) but the first character shouldn't be a number
- For Ex: userName,user_Name,user_Name2 etc.. Are valid.
- But 2user_name ,2userName ..are not valid

contd..

- Variable names are case sensitive .username and userNAME are not same.
- Normally two conventions are used in Python to denote variables.
- Camel case notation or use underscores.
- Camel case is the practice of writing the compound words with mixed casing.For ex:”thisIsAvariableName”
- Another is to use underscores (-).

contd..

- This separates the words. For example:
“user_Name_variable”
- In the above example `userAge=30`, the '=' sign is known as **Assignment Sign**
- It means ,we are assigning the value on the right side of the = sign to the variable on the left.
- So the statements `x=y` and `y=x` have different meanings in programming.

contd..

- For example ,open your IDLE editor and type the following program.

```
x=5
```

```
y=10
```

```
x=y
```

```
print(“x=“,x)
```

```
Print(‘y=‘,y)
```

Now save this program with .py extension.

For example myprog1.py

contd..

- Run the program either by clicking .

Runmodule or F5

The output should be as below.

x=10

y=10

Here in the program you have assigned x=y in the third line. So the value of x is changed to 10 while the value of y do not change .

contd..

- Suppose you change the third line from $x=y$ to $y=x$, mathematically $x=y$ and $y=x$ may be same .But it is not same in programming.
- After changing ,you again save your program and Run the module as done above.What is the output you expect?
- Surprisingly $x=5$, $y=5$ will be the output.
are you convinced with the output?

Basic Operators

- Python accepts all the basic operators like + (addition), (subtraction), * (multiplication), / (division), //(floor division), %(modulus) and **(exponent).
- For example ,if $x=7, y=2$
addition: $x+y=9$
subtraction: $x-y=5$
Multiplication: $x*y=14$
Division: $x/y=3.5$

contd

- Floor division: $x // y = 3$ (rounds off the answer to the nearest whole number)
- Modulus: $x \% y = 1$ (Gives the remainder when 7 is divided by 2)
- Exponent : $x ** y = 49$ (7 to the power of 2)

Additional assignment Operators

- $+=$: $x+= 2$ is same as equal to $x=x+2$
- Similarly for subtraction:
 $x-=2$ is same as $x=x-2$

Data Types

- Python accepts various data types like Integer, Float and String.
- Integers are numbers with no decimal parts like 2 , -5 , 27 , 0 , 1375 etc..
- Float refers to a number that has decimal parts. Ex: 1.2467 ; -0.7865 ; 125.876
- `userHeight = 12.53`
- `userWeight = 65.872`

contd..

- String: string refers to text. Strings are defined within single quotes.
- For Ex: Variable Name = 'initialvalue'
- VariableName="initial value"
- userName='peter'
- userSpouseName="Mary"
- userAge = "35"

here "35" is a string

contd..

- Multiple substrings can be combined by using the concatenate sign(+)
- Ex: "Peter" + "Lee" is equivalent to the string "PeterLee"

Type Casting

- Type casting helps to convert from one data type to another .i.e from integer to a string or vice versa.
- There are three built in functions in Python
`int()` ; `str()` ; `float()`
- To change a float to an integer type
`int(7.3256)`.The output is 7
- To change a string into integer `int("4")`.The output is 4

contd..

- Ex: `int("RAM")` ----- returns syntax error
- `Int("3.7654")` -----returns syntax error
- The `str()` function converts an integer or a float to a string.
- Ex: `str(82.1)` The output is "82.1"
- Ex: `str(757)` The output is "757"
- Ex: `str(Ram)` The output is "Ram"

LIST

- A List in Python is a collection data which are normally related.
- Instead of storing a data as separate variables ,it can be stored as a LIST.
- For ex: to store the age of 4 users , instead of storing them as user1Age,user2Age,user3Age and user4Age we can create a List of this data.
- To some extent, lists are similar to arrays in C. Difference between them is that all the items belonging to a list can be of different data type.

contd..

- To declare a List:

ListName=[initial values]. Note ,always square brackets[] are used to declare a List.

```
userAge =[20,42,23,25]
```

- A List can also be declared without assigning any initial value to it.
- listName=[]. This is an empty list with no items in it.

contd..

- To add items to the List ,we use append().
- The individual items of the List are accessed by their indexes.The index always starts from 0.
- For ex; useAge=[20,25,35,45,65,55]
here userAge[0]= 20,userAge[2]=35.
- Similarly the index -1 denotes the last item
- userAge[-1]=55 and userAge[-2]=65

SLICE

- To access a set of items in a List we use Slice.
- The notation used is x:y.
- Whenever we use slice notation in Python, the item at the start index is always included ,but the item at the end is always excluded i.e y-1 is considered.
- For ex:userAge=[15,25,30,45,50,65,55]
the userAge[0:4] gives values from index 0 to 3 i.e 15,25,30,45

Add and Remove items

- To add items the `append()` function is useful.
- For ex : `userAge.append(100)` will add the value 100 to the List at the end. Now the new List is `userAge=[15,25,30,45,50,65,55,100]`
- To remove any value from the List we write `dellistName[index of item to be deleted]`
- For ex: `del userAge[2]` will give a new List `userAge=[15,25,45,50,65,55,100]`

Program

- # declare the list
`myList[2,5,4,5,"Ram"]`
- # to print the entire list
`print('myList')`
- # to print the third item
`print(myList[3])`
The output is "Ram"
- # to print the last but one item
`print(myList[-2])`

Tuple

- Tuples are also a type of Lists. But the difference is they can't be modified.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses().
- The main differences between lists and tuples are:
- Lists are enclosed in brackets [] and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.
- Tuples can be thought of as **read-only** lists.

contd..

- Declaring a Tuple

```
tupleName=( 'Raju',786 , 2.23, 'cdef',70.2 )
```

```
monthsOfYear("Jan","Feb","March","April")
```

- monthsOfYear[0]="Jan"
- monthsOfYear[-1]= "April"

contd..

- print tuple: # Prints complete list print
- tuple[0] : # Prints first element of the list
- print tuple[1:3]: # Prints elements starting from 2nd till 3rd print
- tuple[2:] : # Prints elements starting from 3rd element
- print tinytuple * 2 : # Prints list two times

Dictionary

- Dictionary is a collection of related data PAIRS.
- The general Syntax is
dictionaryName={dictionarykey:data}
The same dictionary key should not be used twice
- For Ex: to store the username and age of 4 users we can write
- myDictionary={"peter":25,"Ram":15,"Carl":50,"Kundan": 20}

contd...

- Dictionary can also be declared using “dict()” method.
- `userNameAndAge=dict(peter=25,Ram=20,Carl=30, Krish=“not available”)`
- Here round brackets parenthesis) are used instead of curly brackets.
- `dictionaryName={ }` denotes an empty dictionary with no items init

Add items to Dictionary

- To add items to a dictionary the syntax is `dictionaryName[dictionarykey]=data`.
To add “Sam”=40 to the dictionary ,
`userNameAndAge[“Sam”]=40`
- Now the dictionary will become
`userNameAndAge=dict(peter=25,Ram=20,Carl=30, Krish=“not available”,Sam=40)`

contd..

- To delete or remove items from a dictionary the syntax is

```
del dictionaryName[dictionarykey]
```

- Ex: To remove “Ram”:20 the syntax is

```
del usernameAndAge[“Ram”]
```

- Now the dictionary is

```
userNAmeAndAge={“peter”:25,”Carl”:30,  
“Krish”:“not available”,”Sam”:40}
```

Declare Dictionary –Alternative way

- Dictionary can also be declared by alternative method
- `myDict={"one":1.45,5.8:"TwopointFive",3:"+",4.5:2}`
`print(mydict)`

The output is

`{2.5:'TwopointFive',3:'+':4.5,4.5:2}`

Condition Statements

- To change the flow of control ,normally we use some conditional statements in a program .The flow of the program is based on whether the condition is met or not .
- The most common condition statement is Comparison statement which is denoted by $(=)$ sign.
- For ex: when you are writing $a==b$,means you are asking the program to check whether the value of a is equals to that of b or not.

contd..

- The other comparison statements are
 - “not equals” (\neq)
 - “smaller than” ($<$)
 - “greater than” ($>$)
 - “greater than or equal to” (\geq)

Examples: $8 \neq 4$ ----Not equals

$6 \geq 2$ ----Greater than or equal to

$7 > 2$ ---Greater than

$4 < 9$ – Less than or equal to

Logical Operators

- There are three logical operators in Python
- And , or ,not
- The and operator returns True if all the conditions are met ,else it will return False.
- The or operator returns true if atleast one condition is met.Else it will return false.
- The not operator returns True if the condition after the not keyword is false.Else it will return False.

If statement

- It is one of the commonly used control flow statements.
 - It allows the program to evaluate if certain condition is met. And to perform the appropriate action based on the result of the evaluation
 - For example , if condition 1 is met
do task A
elif condition 2 is met
do task B
- elif stands for else if and we can have as many elif statements as we like

Program

- Open your Idle and type the program
- ```
userInput=input("Enter 1 or 2")
if userInput == "1":
 print("Hello Python")
 print("how are you")
elif userInput == "2":
 print("python great")
 print("I Love Python")
else:
 print(" you entered invalid number")
```

# For Loop

- The For Loop executes a block of code repeatedly until the condition in the for statement is no longer valid.
- In Python ,an iterable refers to anything that can be looped over ,such as a string ,list or Tuple.
- The syntax for looping through an iterable is :

`for an iterable`  
`print(a)`

contd..

- Ex: `pets = ['cats','dogs','rabbits','hamsters']`  
for myPets in pets:  
    `print(myPets)`

In this program, first we declared the list `pets` and give the members `'cats','dogs','rabbits'` and `'hamsters'`. Next the statement `for myPets:` loops through the `pets` list and assigns each member in the list to the variable `myPets`.

# contd..

- If we run the program ,the output will be  
cats  
dogs  
rabbits  
hamsters
- It is also possible to display the index of the members in the List.For this the function `enumerate()` is used.
- For index,myPets in `enumerate(pets):`
- `print(index,myPets)`



contd..

- The output is:

0 cats

1 dogs

2 rabbits

3 hamsters

# Looping through a sequence of Numbers

- To loop through a sequence of numbers ,the built-in range() function is used.
- The range() function generates a list of numbers and has the syntax :  
range(start,end,step).
- If start is not mentioned ,the numbers generated will start from zero.
- Similarly if step is not given , a list of consecutive numbers will be generated (i.e step=1)

contd..

- For ex:
- `Range(4)` will generate the `List[0,1,2,3,]`
- `Range(3,9)` will generate a `List[3,4,5,6,7,8]`
- `Range(4,10,2)` will generate `List[4,6,8]`

# Example

- To check the working of range function with for loop type the following script on Idle.
- ```
for i in range(5);  
    print(i)
```

The output will be:

0

1

2

3

4

While Loop

- A while loop repeatedly executes instructions inside the loop while a certain condition remains true.
- The syntax of a while loop is
while the condition is true
do the Task A

For Ex:

Counter= 5

While counter > 0:

contd..

- `print('counter=',counter)`
`counter = counter-1`

When you run the program the output will be

`counter=5`

`counter=4`

`counter=3`

`counter=2`

`counter=1`

Break

- This will help to exit from the loop when a certain condition is met. Here break is a keyword in Python.
- Ex:

```
j=0
for I in the range(5):
    j=j+2
    print('i=',i, 'j=',j)
    if j==6:
        break
```

contd..

- The output of the above program will be:

i=0,j=2

i=1,j=4

i=2,j=6

without the break keyword ,the program should loop from i=0 to i=4 because the function range(5) is used.

Due to the keyword break ,the program ends early at i=2.Because when i=2 ,j reaches the value 6 .

Continue

- Continue is another important keyword for Loops.
- When the keyword continue is used ,the rest of the loop after the keyword is skipped for that iteration.
- Ex:

```
j=0
for i in range (5):
    j=j+2
    print('\ni=','I,j=','j)
    if j==6:
        continue
```

contd..

- Print('i will be skipped over if j=6')

The output of the program is as below:

i=0,j=2

I will be skipped over if j=6

i=1,j=4

I will be skipped over if j=6

i=2,j=6

i=3,j=8

i= 4,j=10

I will be skipped over if j=6

When j=6,the line after the continue keyword is not printed.

Try,Except

- This control statement controls how the program proceeds when an error occurs The syntax is as below.
- Try:
 do a Task
except
 do something else when an error occurs

contd..

- For Ex:

try:

```
answer = 50/0
```

```
print(answer)
```

except:

```
print('An error occurred')
```

when we run the program, the message “An error occurred” will be displayed because `answer=50/0` in the try block cannot divide a number by 0. So, the remaining of the try block is ignored and the statement in the except block is executed.

Functions-Modules

- Functions are pre-written codes that perform a dedicated task.
- For ex: the function `sum()` add the numbers.
- Some function require us to pass data in for them to perform their task. These data are known as parameters and we pass them to the function by enclosing their values in parenthesis() separated by commas.
- The `print()` function is used to display the text on the screen.

contd..

- `replace()` is another function used for manipulating the text strings.
- For Ex:
“`helloworld`”.`replace`(“`world`”, “`Universe`”)
here “`world`” and “`universe`” are the parameters. The string before the Dot is the string that will be affected. So, “`hello world`” will be changed to “`hello Universe`”

Your own Functions

- Python has flexibility to define your own functions and can be used them in the program.
- The syntax is:

```
def functionName(parameters):  
    code for what the function should do  
    return[expression]
```

The two keywords here are “def” and “return”

contd..

- def tells the program that the indented code from the next line onwards is part of the function and return is the keyword that is used to return an answer from the function.
- Once the function executes a return statement ,it will exit.
- If your function need not return any value you can write return or return None.

contd..

- Ex:

```
def checkIfPrime(numberToCheck):  
    for x in range(2,numberToCheck):  
        if(numberToCheck%x==0):  
            return False  
    return True
```

The above function will check whether the given number is a prime number or not. If the number is not a prime number it will return False otherwise return True.

Variable Scope

- An important concept to understand while defining a function is the concept of variable scope.
- Variables defined inside a function are treated differently from variables defined outside.
- Any variable defined inside a function is only accessible within the function and such functions are known as local variables.
- Any variable declared outside a function is known as global variable and it is accessible anywhere in the program.

Importing Modules

- Python provides a large number of built-in functions and these are saved in files known as modules.
- To use the modules in our program ,first we have to import them into our programs.
and this is done by using the keyword import.

Ex: To import the module random we write
`import random`

contd..

- To use the `randrange()` function in the `random` module we use
`random.randrange(1,10)`.
This can also be written as
`r.randrange(1,10)`.
- If we want to import more than one functions we separate them with a comma
- To import the `randrange()` and `randint()` functions we write `from random import randrange,randint`.

Creating own Module

- Creating a module is simple and it can be done by saving the file with a .py extension and put it in the same folder as the python file that you are going to import it from.
- Suppose you want to use the `checkIfPrime()` function defined earlier in another Python script. First save the code above as `prime.py` on your desktop.

contd..

- The prime.py should have the following code.

```
def checkIfPrime(numberToCheck):  
    for x in range(2,numberToCheck):  
        if(numberToCheck%x==0):  
            return False  
    return True
```

Next create another python file and name it
useCheckIfPrime.py

This should have the following code

contd..

- Import prime

```
answer = prime.checkIfPrime(17)
```

```
print(answer)
```

- Now run useCheckIfPrime.py.
- You get the output as True
- Suppose you want to store prime.py and useCheckIfPrime.py in different folders, you have to give the python's system path.

contd..

- Create a folder called 'MyPythonModules' in the C drive to store prime.py.
- Add the following code to the top of your useCheckIfPrime.py file.

(i.e before the import prime line)

- Import sys
if 'C:\\MyPythonModules' not in sys.path:
 sys.path.append('C:\\MyPythonModules')
here sys.path refers to python's system path.

contd..

- The code above appends the folder 'C:\MyPythonModules' to your system path.
- Now you can put prime.py in C:\MyPythonModules and checkIfPrime.py in any other folder of your choice.

Handling Files

- The basic type of a file is a text file. It consists of multiple lines of text. To create a text file type some lines of text like
“I am learning Python programming.
Python is an interesting language” .
save this text file as myfile.txt on your desktop.
Next open Idle and type the code below.

contd..

- ```
f= open('myfile.txt','r')
firstline=f.readline()
secondline=f.readline()
print(firstline)
print(secondline)
f.close()
```

Save this code as fileOperation.py on your desktop.

- To open a file we can use open() function.

contd..

- The open function require two parameters: The first one is the path to the file and the second parameter is the mode.
- The commonly used modes are
- ‘r’ mode: For reading only
- ‘w’ mode: For writing only.
- ‘a’ mode: For appending .If the specified file does not exist ,it will be created and if the file exist ,any data written to the file is automatically added at the end of the file.

contd..

- ‘r+’ mode: For both reading and writing.
- After opening the file  
    `firstline=f.readline()` reads the firstline in the file and assigns it to the variable `firstline`.

Each time the `readline()` function is called ,it reads a new line from the file.

When you run the program the output will be  
    I am learning Python programming.

    Python is an interesting language.

contd..

- In the output ,you can observe that a line break is inserted after each line.
- Because the function `readline()` adds the 'n\' characters to the end of each line.
- If you don't want the extra line between each line of text ,you can write `print(firstline,end='')`. This will remove the 'n\' characters
- `f.close()` function closes the file. One should always close the file after reading is done. It will free up any system resources.

# For Loop to Read Text Files

- For loop can also be used to read a text file.
- For ex: 

```
f = open('myfile.txt','r')
for line in f:
 print(line,end='')
f.close()
```

The for loop helps to read the text file line by line.

# Opening and reading Text files by Buffer size

- To avoid the too much usage of memory we can use the `read()` function instead of `readline()` function.
- Ex: `msg=inputfile.read(10)`

The program reads only the next 10 bytes and keeps doing it until the entire file is read.



# Deleting and re-naming the Files

- `remove()` and `rename()` are the two important functions available as modules which have to be imported before they are used in a program.
- The `remove()` function deletes a file.the syntax is `remove(filename)`
- Ex: `remove('myfile.txt')`
- To rename a file the function `rename()` is used.
- The syntax is `rename(old name,new name)`
- `rename('oldfile.txt','newfile.txt')` will rename `oldfile.txt` to `newfile.txt`

# Simple Examples-1

- Python code to find the area of a triangle:
- First open Idle and type the code
- ```
a = float(input('Enter first side:'))  
b= float(input('Enter first side:'))  
c= float(input('Enter first side:'))  
s=(a+b+c)/2  
area=(s*(s-a)*(s-b)*(s-c))**0.5  
print('The area of the triangle is %0.2f'%area)
```

Simple Examples-2

- Python code to find Square root

```
num=float(input('Enter a number:'))
```

```
num_sqrt=num**0.5
```

```
print('The square root of %0.3f is % 0.3f%  
(num,num_sqrt))
```

This program is saved with .py extension and executed .the out will ask to enter a number and on entering a number it gives the squire root of the number.

Simple Examples-3

- Python code for factorial of a number
- ```
Num= int(input('Enter a number:'))
factorial = 1
if num < 0:
 print('sorry ,factorial does not exist for
 negative numbers')
elif num == 0;
 print("The factorial of 0 is 1")
```

contd..

- `else for i in range(1,num+1):`  
    `factorial = factorial*i`  
    `print("The factorial of “,num,”is”,factorial)`
- Save the program with .py extension and run module or press F5.
- The program will ask to enter a number
- After entering the number you get the output

## Simple Examples-4

- Print all the Prime numbers in an Interval.
- Lower=int(input('Enter lower range:'))

```
upper=int(input('Enter upper range'))
```

```
for num in range(lower,upper+1):
```

```
 if num>1:
```

```
 for i in range(2,num):
```

```
 if(num%i)==0:
```

```
 break
```

```
 else:
```

```
 print(num)
```

## Simple Examples-5

- Code to swap two numbers
- ```
x= input('Enter value of x:')  
y=input('Enter value of y:')  
temp = x  
x=y  
y= temp  
print('The value of x after swapping:  
{ }'format(x))  
print('The value of y after swapping:  
{ }'format(y)).
```

Run the code and observe the output

Simple Examples-6

- Solve a quadratic equation $ax^2+bx+c=0$
- Import cmath # add complex math module

```
a= float(input('Enter a:'))
```

```
b= float(input('Enter b:'))
```

```
c= float(input('Enter c:'))
```

```
d= (b**2)-(4*a*c) //calculate discriminant
```

```
sol1=(-b-cmath.sqrt(d))/(2*a)
```

```
sol2=(-b+cmath.sqrt(d))/(2*a)
```

```
print('The solutions are {0} and {1}'.format(sol1,sol2))
```


Simple Examples-7

- To find ASCII value of a given characters
- # Take character from user

```
c = input("Enter a character: ")
```

```
print("The ASCII value of '" + c + "' is",ord(c))
```

Note: here `ord(c)` is a builtin function available in Python to find the ASCII value of a character

ACKNOWLEDGEMENT

- I don't claim any ownership for this lecture as it is the collection from many books & web resources
- I thank the authors of the books and web resources from where I collected the information and code.
- The sole objective behind this effort is only to encourage and inspire if possible some young minds towards this wonderful programming language -Python

References

- Learn Python in One Day and Learn It Well - Jamie Chan.
- Sams Teach Yourself Python Programming for Raspberry.
- Programming Python, 5th Edition, O'Reilly Media.
- Python Cook Book O'Reilly Media.