**Python Lab Manual**

**1.Write Python code to demonstrate Built–in data types in python also display type of data.**

* Built-in data types in Python are fundamental data structures provided by the Python programming language.
* They are pre-defined and available for use without requiring any additional libraries or modules.
* Python offers several built-in data types, including:
* **Numeric Data Types:**Numeric data types in Python are used to represent numerical values. Python provides three primary numeric data types:
  + **Int –** It stores the integers values that can be positive or negative and do not contain any decimal point. Example: num1=10, num2 = 15
  + **Float**– These are floating-point real numbers that stores the decimal values. It consists of integer and fraction parts. Example: fnum = 25.4, fnum1=67.8
  + **Complex**– These are complex numbers specified as a real part and an imaginary part. They are stored in the form of a + bj where a is the real part and j represents the imaginary part. Example: num3= 2 + 3j, numcom = 5 – 7j
* **String Data Type(str):** Represents a sequence of characters enclosed in single quotes (‘ ‘) or double quotes (” “), such as “Hello, World!”, ‘Python’.
* **Boolean Data Type(bool)**: Represents either True or False, used for logical operations and conditions.
* **Collection Data Types:**
  + **list:**Represents an ordered and mutable collection of items, enclosed in square brackets ([]).
  + **tuple:** Represents an ordered and immutable collection of items, enclosed in parentheses ().
  + **dict**: Represents a collection of key-value pairs enclosed in curly braces ({}) with unique keys.
  + **set:** Represents an unordered and mutable collection of unique elements, enclosed in curly braces ({}) or using the set() function.

**CODE:**

#Demonstrate Python code on data types.  
#Number  
a=10  
print("a = ",a," Data Type of a is =",type(a))  
b=20.30  
print("b = ",b," Data Type of b is =",type(b))  
c=3+2j  
print("c = ",c," Data Type of c is =",type(c))  
#Sequence  
str="RLS BCA"  
print("String = ",str," Data Type of str is =",type(str))  
ls=["Amita",52,80.90,True]  
print("List = ",ls," Data Type of ls is =",type(ls))  
tp=("Deepa","Mohan","Sunita")  
print("Tuple = ",tp," Data Type of tp is =",type(tp))  
#Boolean  
d=True  
print("D = ",d," Data Type of d is =",type(d))  
#Dictionary  
dic={1:"Apple",2:"Banana",3:"Orange"}  
print("Dic = ",d," Data Type of dic is =",type(dic))  
#Set  
set={1,2,3,4,5}  
print("set = ",set," Data Type of seq is =",type(set))

**OUTPUT:**

a = 10 Data Type of a is = <class 'int'>

b = 20.3 Data Type of b is = <class 'float'>

c = (3+2j) Data Type of c is = <class 'complex'>

String = RLS BCA Data Type of str is = <class 'str'>

List = ['Amita', 52, 80.9, True] Data Type of ls is = <class 'list'>

Tuple = ('Deepa', 'Mohan', 'Sunita') Data Type of tp is = <class 'tuple'>

D = True Data Type of d is = <class 'bool'>

Dic = True Data Type of dic is = <class 'dict'>

set = {1, 2, 3, 4, 5} Data Type of seq is = <class 'set'>

**2.Demonstrate Python code on String slicing.**

str="Amit Kumar is student of RLSBCA College. He is in BCA IIIA Division."

1)Access name from above string using + indexing.

2)Access College name from above string using - indexing.

3)Display Student division.

4)Display student name and from which college.

5)Display Full string through slicing.

6)Display Reverse String using -ve indexing.

str="Amit Kumar is student of RLSBCA College. He is in BCA IIIA Division."  
#Access name from above string using + indexing  
print("Name = ",str[0:10])  
#Access College name from above string using - indexing.  
print("College= ",str[-43:-29])  
#Display Student division.  
print("Division= ",str[-14:-9])  
#Display student name and from which college.  
print("Student ",str[0:10],"from",str[-43:-29])  
#Full string  
print("String = ",str[:])  
#Display Reverse String using -ve indexing.  
print("Rverse string = ",str[::-1])

OUTPUT:

Name = Amit Kumar

College= RLSBCA College

Division= IIIA

Student Amit Kumar from RLSBCA College

String = Amit Kumar is student of RLSBCA College. He is in BCA IIIA Division.

Rverse string = .noisiviD AIII ACB ni si eH .egelloC ACBSLR fo tneduts si ramuK timA

**3.Demonstrate python code on Basic List Operations.**

**List Operations in Python**

A few of the basic list operations used in Python programming are

1. append() -The append() method adds elements at the end of the list.

#### 2. extend()-The extend() method adds more than one element at the end of the list.

#### 3. insert()-The insert() method can add an element at a given position in the list.

#### 4. remove()-The remove() method removes an element from the list.

#### 5. pop()-The method pop() can remove an element from any position in the list.

#### 6. slice-The slice operation is used to print a section of the list.

#### 7. reverse()-You can use the reverse() operation to reverse the elements of a list.

#### 8. sort()-The sort method sorts the list in ascending order. You can only perform this operation on homogeneous lists, which means lists with similar elements.

#Demo on List operation  
fruits=["Orange","Apple","Pinapple","Gava"]  
print("List = ",fruits)  
fruits.append("Chikku")  
print("After adding new Item at end= ",fruits)  
fruits.extend(["Banana","Strawberry"])  
print("After adding new Items at end= ",fruits)  
fruits.insert(2,"Jackfruit")  
print("After adding new Items at index 2= ",fruits)  
print("Access elements from fruits using slice = ",fruits[1:4])  
fruits.reverse()  
print("Reverse list = ",fruits)  
fruits.sort()  
print("Sorted list = ",fruits)  
print("Reverse list = ",fruits)  
fruits.remove("Chikku")  
print("After Removing Element= ",fruits)  
fruits.pop(2)  
print("After Removing Element using pop= ",fruits)

#### OUTPUT:

#### List = ['Orange', 'Apple', 'Pinapple', 'Gava']

#### After adding new Item at end= ['Orange', 'Apple', 'Pinapple', 'Gava', 'Chikku']

#### After adding new Items at end= ['Orange', 'Apple', 'Pinapple', 'Gava', 'Chikku', 'Banana', 'Strawberry']

#### After adding new Items at index 2= ['Orange', 'Apple', 'Jackfruit', 'Pinapple', 'Gava', 'Chikku', 'Banana', 'Strawberry']

#### Access elements from fruits using slice = ['Apple', 'Jackfruit', 'Pinapple']

#### Reverse list = ['Strawberry', 'Banana', 'Chikku', 'Gava', 'Pinapple', 'Jackfruit', 'Apple', 'Orange']

#### Sorted list = ['Apple', 'Banana', 'Chikku', 'Gava', 'Jackfruit', 'Orange', 'Pinapple', 'Strawberry']

#### Reverse list = ['Apple', 'Banana', 'Chikku', 'Gava', 'Jackfruit', 'Orange', 'Pinapple', 'Strawberry']

#### After Removing Element= ['Apple', 'Banana', 'Gava', 'Jackfruit', 'Orange', 'Pinapple', 'Strawberry']

#### After Removing Element using pop= ['Apple', 'Banana', 'Jackfruit', 'Orange', 'Pinapple', 'Strawberry']

**4.Demonstrate python code to**

**I) Convert List to tuple and vice versa.**

**II)Add tuple to list and vice versa.  
list function**

The list function creates a list object. A list object is a collection which is ordered and changeable. We can convert tuple into list using list ().

**Example**

tp= (1,2,3)

ls=list(tp)

**tuple function**

The tuple function creates a tuple. A tuple object is a collection which is not changeable. We can convert list into tuple using tuple ().

**Example**

ls=[1,2,3]

tp=tuple(ls)

Using simple + operator we add tuple to list and vice versa.

ls= [4, 5, 7, 1, 7]  
tp = (4, 1, 8, 3, 9)  
  
print("List is: ", ls)  
print("Tuple is: ", tp)  
  
tp1=tuple(ls)  
ls1=list(tp)  
print("After Converting List to tuple = ",tp1)  
print("After Converting tuple to list= ",list(ls1))  
  
add1=ls+list(tp)  
print("After tuple add to list = ",add1)  
add2=tp+tuple(ls)  
print("After list add to tuple = ",add2)

OUTPUT:

List is: [4, 5, 7, 1, 7]

Tuple is: (4, 1, 8, 3, 9)

After Converting List to tuple = (4, 5, 7, 1, 7)

After Converting tuple to list= [4, 1, 8, 3, 9]

After tuple add to list = [4, 5, 7, 1, 7, 4, 1, 8, 3, 9]

After list add to tuple = (4, 1, 8, 3, 9, 4, 5, 7, 1, 7)

**5.Demonstrate python code for aggregate functions apply on list also using import package statistics calculate mean, mode, median of data in list.**

**count ()** returns the number of elements with the specified value.

**min ()** returns the elements from the *list* with minimum value.

**max ()** returns the elements from the *list* with maximum value.

**sum ()** returns the sum of elements from the *list*

**mean ()** returns average value

**median ()** returns middle most value

**mode ()** returns most frequent value

import statistics as st  
score=[22,30,21,4,25,27,21,20,19,16,21,15]  
print("Scores = ",score)  
#Count no.of 21 in list  
print("Total count of 21 in list = ",score.count(21))  
#min value  
print("Minimum value in list = ",min(score))  
#max value  
print("Maximum value in list = ",max(score))  
#Sum of values in list  
print("Sum of values in list = ",sum(score))  
#Average values in list  
print("Mean = %.2f"%st.mean(score))  
#Average values in list  
print("Mode = ",st.mode(score))  
#Average values in list  
print("Median = ",st.median(score))

OUTPUT:

Scores = [22, 30, 21, 4, 25, 27, 21, 20, 19, 16, 21, 15]

Total count of 21 in list = 3

Minimum value in list = 4

Maximum value in list = 30

Sum of values in list = 241

Mean = 20.08

Mode = 21

Median = 21.0

**6.Demonstrate python code for dictionary basic operations.**

## **Dictionary**

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

## **Accessing Items**

You can access the items of a dictionary by referring to its key name, inside square brackets:

## **Change Values**

You can change the value of a specific item by referring to its key name:

## **Adding Items**

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

**Remove Item**

The pop() method removes the item with the specified key name:

## **Copy a Dictionary**

Using copy () method you can copy dictionary in another dictionary.

Accessing Keys and values

We can access keys in dictionary using dictionary keys (). similarly, we can access values of dictionary by values ()

dic={'A':"Apple",'B':"Banana",'G':"Grapes",'P':"Pineapple"}  
print("Dictionary = ",dic)  
#Accesing keys  
print("Dictionary Keys= ",dic.keys())  
#Accesing Values  
print("Dictionary Keys= ",dic.values())  
#Accessing item  
print("key P value is ",dic['P'])  
#Adding new item  
dic['O']="Orange"  
print("After adding new item in Dictionary= ",dic)  
#Modifying item  
dic['G']="Gava"  
print("After Modifying item in Dictionary= ",dic)  
#Copying Dictionary  
dic1=dic.copy()  
print("After copying dic in dic1= ",dic1)  
#Remove item from dictionary  
dic.pop('P')  
print("After Removing Pineapple from dictionary= ",dic)

OUTPUT:

Dictionary = {'A': 'Apple', 'B': 'Banana', 'G': 'Grapes', 'P': 'Pineapple'}

Dictionary Keys= dict\_keys(['A', 'B', 'G', 'P'])

Dictionary Keys= dict\_values(['Apple', 'Banana', 'Grapes', 'Pineapple'])

key P value is Pineapple

After adding new item in Dictionary= {'A': 'Apple', 'B': 'Banana', 'G': 'Grapes', 'P': 'Pineapple', 'O': 'Orange'}

After Modifying item in Dictionary= {'A': 'Apple', 'B': 'Banana', 'G': 'Gava', 'P': 'Pineapple', 'O': 'Orange'}

After copying dic in dic1= {'A': 'Apple', 'B': 'Banana', 'G': 'Gava', 'P': 'Pineapple', 'O': 'Orange'}

After Removing Pineapple from dictionary= {'A': 'Apple', 'B': 'Banana', 'G': 'Gava', 'O': 'Orange'}

**7.Demonstrate Python code on membership operators in python.**

Membership operators in Python are operators used to test whether a value exists in a sequence, such as a list, tuple, or string. The membership operators available in Python are,

**in:** The in operator returns True if the value is found in the sequence.

**not in:** The not in operator returns True if the value is not found in the sequence

#List  
ls=[20,22,33,44,66,4,9,90]  
#Check number is present in list if not add in list  
searchnum=int(input("Enter Number = "))  
if searchnum in ls:  
 print(searchnum,"Present in List")  
if searchnum not in ls:  
 ls.append(searchnum)  
 print(searchnum,"not present in list so added in list")  
print(ls)

OUTPUT:

Enter Number = 66

66 Present in List

[20, 22, 33, 44, 66, 4, 9, 90]

Or

Enter Number = 89

89 not present in list so added in list

[20, 22, 33, 44, 66, 4, 9, 90, 89]

**8.Demonstrate python code on nested loop.**

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop"

**Syntax of using a nested for loop in Python**

# outer for loop

**for** element **in** sequence

# inner for loop

**for** element **in** sequence:

body of inner **for** loop

body of outer **for** loop

#Example of Nested for loop  
#Table of number  
num=int(input("Enter number = "))  
for i in range(1,(num+1)):  
 for j in range(1,11):  
 print(i ,"X" ,j,"=",i\*j)

OUTPUT:

Enter number = 3

1 X 1 = 1

1 X 2 = 2

1 X 3 = 3

1 X 4 = 4

1 X 5 = 5

1 X 6 = 6

1 X 7 = 7

1 X 8 = 8

1 X 9 = 9

1 X 10 = 10

2 X 1 = 2

2 X 2 = 4

2 X 3 = 6

2 X 4 = 8

2 X 5 = 10

2 X 6 = 12

2 X 7 = 14

2 X 8 = 16

2 X 9 = 18

2 X 10 = 20

3 X 1 = 3

3 X 2 = 6

3 X 3 = 9

3 X 4 = 12

3 X 5 = 15

3 X 6 = 18

3 X 7 = 21

3 X 8 = 24

3 X 9 = 27

3 X 10 = 30

**9. Write a python program to demonstrate the concept of user defined functions.**

**User Defined Functions**

A function that you define yourself in a program is known as user defined function. You can give any name to a user defined function, however you cannot use the python keywords as function name.

In python, we define the user-defined function using def keyword, followed by the function name.

Function name is followed by the parameters in parenthesis, followed by the colon

Syntax

def function\_name(parameter\_1, parameter\_2, ...) :

statements

....

## **Calling a user-defined function**

You can call a user defined function by using function name followed by the arguments in the parenthesis.

**For example:**

function\_name(argument\_1, argument\_2)

def fact(num):  
 fact=1  
 for i in range(1,(num+1)):  
 fact=fact\*i  
 return fact  
print("Factorial of 5 is ",fact(5))  
print("Factorial of 6 is ",fact(6))

**OUTPUT:**

**Factorial of 5 is 120**

**Factorial of 6 is 720**

**10. Write a python program to demonstrate map () function .**

**map ()** function returns a map object (which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

## Python map () Function Syntax

**Syntax**: map (fun, iter)

**Parameters:**

* **fun:** It is a function to which map passes each element of given iterable.
* **iter:** It is iterable which is to be mapped.

def square(n):  
 return n \* n  
  
# We square all numbers using map()  
numbers = [1, 2, 3, 4]  
print("Numbers = ",numbers)  
result = map(square, numbers)  
print("Square of Numbers = ",list(result))

**OUTPUT:**

Numbers = [1, 2, 3, 4]

Square of Numbers = [1, 4, 9, 16]

**We can also use map function with Lambda**

# Python Map Lambda

A lambda expression is a way of creating a little function inline, without all the syntax of a def. Here is a lambda with a single n parameter, returning the parameter value doubled.

lambda n: n \* 2

numbers = [1, 2, 3, 4]  
sq\_num=list(map(lambda x:x\*x,numbers))  
print("Numbers= ",numbers)  
print("Square of Numbers= ",sq\_num)

**OUTPUT:**

Numbers= [1, 2, 3, 4]

Square of Numbers= [2, 4, 6, 8]

**11. Write a python program to demonstrate string built-in functions.**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **capitalize()** | **Converts the first character to upper case** |
| **isalpha()** | **Returns True if all characters in string are in alphabet** |
| **islower()** | **Returns True if all characters in the string are lower case** |
| **isupper()** | **Returns True if all characters in the string are upper case** |
| **lower()** | **Converts a string into lower case** |
| **replace()** | **Returns a string where a specified value is replaced with a specified value** |
| **split()** | **Splits the string at the specified separator and returns a list** |
| **Method** | **Description** |
| **capitalize()** | **Converts the first character to upper case** |
| **isalpha()** | **Returns True if all characters in string are in alphabet** |
| **islower()** | **Returns True if all characters in the string are lower case** |
| **isupper()** | **Returns True if all characters in the string are upper case** |
| **lower()** | **Converts a string into lower case** |
| **replace()** | **Returns a string where a specified value is replaced with a specified value** |
| **split()** | **Splits the string at the specified separator and returns a list** |

str="rls bca college"  
print("String in Upper case = ",str.upper())  
print("String in Lower case = ",str.lower())  
print("String First letter in capital = ",str.capitalize())  
print("String all words first letter in capital = ",str.title())  
print("Is all letters string is alphabets = ",str.isalpha())  
print("Is all letters string is in uppercase = ",str.isupper())  
print("Is all letters string is in lowercase = ",str.islower())  
print("Total Length of string = ",len(str))  
print("Count letter l in string = ",str.count('l'))  
print("Join string with \_ = ","\_".join(str))  
print("Index of b in string = ",str.index('b'))  
print("Replace bca with BCA= ",str.replace('bca','BCA'))  
print("After spliting string= ",str.split())  
print("After string partition = ",str.partition('bca'))

**OUTPUT:**

String in Upper case = RLS BCA COLLEGE

String in Lower case = rls bca college

String First letter in capital = Rls bca college

String all words first letter in capital = Rls Bca College

Is all letters string is alphabets = False

Is all letters string is in uppercase = False

Is all letters string is in lowercase = True

Total Length of string = 15

Count letter l in string = 3

Join string with \_ = r\_l\_s\_ \_b\_c\_a\_ \_c\_o\_l\_l\_e\_g\_e

Index of b in string = 4

Replace bca with BCA= rls BCA college

After spliting string= ['rls', 'bca', 'college']

After string partition = ('rls ', 'bca', ' college')

**12. Write a python program to demonstrate dictionary lookup and reverse lookup.**

Dictionary Lookup

Searching value by key in dictionary

Given a dictionary d and a key k, it is easy to find the corresponding value v = d[k]. This operation is called a **lookup**.

Reverse Lookup

Searching key by value in dictionary

 if you have v and you want to find k.

dic={'a':"Apple",'b':"Banana",'g':"Grapes",'p':"PineApple",  
 's':"strawberry"}  
#Lookup  
def lookup():  
 key=input("Enter Key = ")  
 if key in dic:  
 print("values = ",dic[key])  
 else:  
 print(key,"Key not present in dictionary")  
def reverselookup():  
 flag=0  
 value=input("Enter Value = ")  
 for key in dic:  
 if dic[key]==value:  
 flag=1  
 break  
 else:  
 flag=0  
 if(flag==1):  
 print("key = ",key)  
 else:  
 print("for value ",value,"key is not present")  
lookup()  
reverselookup()

**OUTPUT:**

Enter Key = a

values = Apple

Enter Value = Banana

key = b

**13. Write a** [**Python Program to create a Set**](https://www.tutorialgateway.org/python-program-to-create-a-set/) **and demonstrate set theory operations.**

A set is a mutable, unordered group of elements, where the elements themselves are immutable.

**Adding items to the set**

Python provides the **add()** method and **update()** method which can be used to add some particular item to the set. The add() method is used to add a single element whereas the update() method is used to add multiple elements to the set.

**Removing items from the set**

Python provides the **discard()** method and **remove()** method which can be used to remove the items from the set.

Python provides the clear() method to remove all the items from the set.

**Python Set Operations**

Set can be performed mathematical operation such as union, intersection, difference, and symmetric difference.

s1={"Monday","Tuesday","Wednesday"}  
s2={"Sunday","Monday"}  
print("Set1 = ",s1)  
print("Set2 = ",s2)  
s1.add("Thursday")  
print("After adding new day in s1 = ",s1)  
s1.update(["Friday","Saturday"])  
print("After updating new days in s1 = ",s1)  
s1.remove("Saturday")  
print("Removing day from s1 = ",s1)  
print("s1 union s2 = ",s1|s2)  
print("s1 intersection s2 = ",s1&s2)  
print("s1 minus s2 = ",s1-s2)  
print("s1 Symmetric difference s2 = ",s1^s2)

**OUTPUT:**

Set1 = {'Monday', 'Tuesday', 'Wednesday'}

Set2 = {'Monday', 'Sunday'}

After adding new day in s1 = {'Monday', 'Tuesday', 'Wednesday', 'Thursday'}

After updating new days in s1 = {'Monday', 'Saturday', 'Wednesday', 'Thursday', 'Friday', 'Tuesday'}

Removing day from s1 = {'Monday', 'Wednesday', 'Thursday', 'Friday', 'Tuesday'}

s1 union s2 = {'Monday', 'Wednesday', 'Thursday', 'Friday', 'Tuesday', 'Sunday'}

s1 intersection s2 = {'Monday'}

s1 minus s2 = {'Tuesday', 'Friday', 'Wednesday', 'Thursday'}

s1 Symmetric difference s2 = {'Wednesday', 'Thursday', 'Sunday', 'Friday', 'Tuesday'}

**14. Write a Python program to demonstrate Array and Array functions.**

**ARRAYS**

An array is defined as a collection of items that are stored at contiguous memory locations. It is a container which can hold a fixed number of items, and these items should be of the same type

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

import array as arr  
array1=arr.array('i',[1,2,3,4,5])  
print("Array =",array1)  
#find the length  
print("The length of the array is:=",len(array1))  
#insert()  
array1.insert(2,56)  
print("After insertion the array items:=",array1)  
#append()  
array1.append(80)  
print("After appending the array items are:=",array1)  
#extend()  
array1.extend([100,200,300,100,100])  
print("After Extending array items are:=",array1)  
#remove()  
array1.remove(56)  
print("After remove items are:=",array1)  
#pop()  
array1.pop()  
print("After poping array items are:=",array1)  
#index()  
print("The index of array elements :=",array1.index(100))  
#count()  
print("The number of counts:=",array1.count(100))  
#pop()  
print("After pop the array items are:=",array1.pop(2))  
print(array1)

**OUTPUT:**

Array = array('i', [1, 2, 3, 4, 5])

The length of the array is:= 5

After insertion the array items:= array('i', [1, 2, 56, 3, 4, 5])

After appending the array items are:= array('i', [1, 2, 56, 3, 4, 5, 80])

After Extending array items are:= array('i', [1, 2, 56, 3, 4, 5, 80, 100, 200, 300, 100, 100])

After remove items are:= array('i', [1, 2, 3, 4, 5, 80, 100, 200, 300, 100, 100])

After poping array items are:= array('i', [1, 2, 3, 4, 5, 80, 100, 200, 300, 100])

The index of array elements := 6

The number of counts:= 2

After pop the array items are:= 3

array('i', [1, 2, 4, 5, 80, 100, 200, 300, 100])

**15. Write a python program to perform read & write operations on a file.**

## **File Handling**

The key function for working with files in Python is the open() function.

The open() function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

r+ Opens a file for both reading and writing. The file pointer will be at the beginning of the file.

w+ Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, it creates a new file for reading and writing.

a+ Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

f=open("E:\PYTHON\demo.txt","w+")  
#Write contents in file  
f.write("JAVA \n")  
f.write("PYTHON \n")  
f.write("Big Data \n")  
f.write("R \n")  
#reading file  
f.seek(0)  
print(f.read())

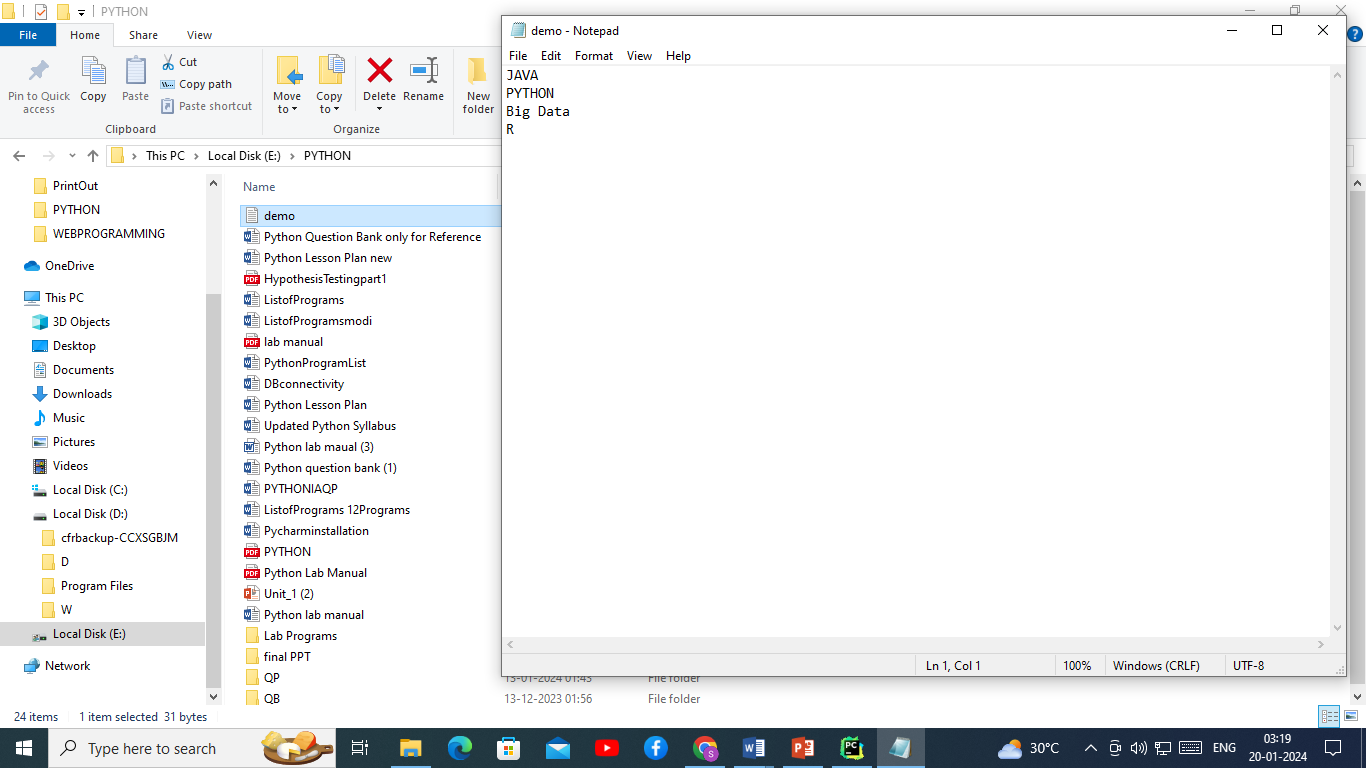
**OUTPUT:**

JAVA

PYTHON

Big Data

R



**16-Write a python program to demonstrate Class and Objects.**

## **Python Classes/Objects**

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

## **Create a Class**

To create a class, use the keyword class:

Example:

Create class named MyClass with property named X:

class MyClass:  
  x = 5

## **Create Object**

Create object m1 of MyClass with:

M1= MyClass()

class Employee:  
 ssn=0  
 name = ""  
 dept=""  
e1=Employee()  
e1.ssn=111  
e1.name="Amit P"  
e1.dept="Accounts"  
e2=Employee()  
e2.ssn=112  
e2.name="Suman L"  
e2.dept="CS"  
print("SSN", "Name"," Department")  
print(e1.ssn,e1.name,e1.dept)  
print(e2.ssn,e2.name,e2.dept)

OUTPUT:

SSN Name Department

111 Amit P Accounts

112 Suman L CS

**17-Write a Python program to demonstrate user defined functions in Class.**

## **What is Class Method in Python?**

Class methods are methods that are called on the [class](https://pynative.com/python-classes-and-objects/) itself, not on a specific object instance. Therefore, it belongs to a class level, and all class instances share a class method.

* **A class method is bound to the class** and not the object of the class. It can access only class variables.
* It can modify the class state by changing the value of a [class variable](https://pynative.com/python-class-variables/) that would apply across all the class objects.

In method implementation, if we use only class variables, we should declare such methods as class methods.

The class method can be called using ClassName.method\_name() as well as by using an object of the class.

class Employee:  
 ssn=0  
 name = ""  
 dept=""  
 def dispaly(self,ssn,name,dept):  
 self.ssn=ssn  
 self.name=name  
 self.dept=dept  
 print("SSN = ",ssn,"Name=",name,"Department=",dept)  
e1=Employee()  
e1.dispaly(111,"Amit Kumar","Accounts")  
e2=Employee()  
e2.dispaly(112,"Nilam Dev","Finanace")

**OUTPUT:**

SSN = 111 Name= Amit Kumar Department= Accounts

SSN = 112 Name= Nilam Dev Department= Finanace

**18.Write a python program to demonstrate init() constructor in class.**

Constructors are generally used for instantiating an object. The task of constructors is to initialize (assign values) to the data members of the class when an object of the class is created. In Python the \_\_init\_\_() method is called the constructor and is always called when an object is created.  
**Syntax of constructor declaration :**

def \_\_init\_\_(self):

# body of the constructor

class Addition:  
 def \_\_init\_\_(self,\*args):  
 if(len(args)==2):  
 add=args[0]+args[1]  
 print("Addtion of ",args[0],"+",args[1],"=",add)  
 elif(len(args)==3):  
 add = args[0] + args[1]+args[2]  
 print("Addtion of ", args[0], "+", args[1],"+",args[2], "=", add)  
a1=Addition(2,3)  
a2=Addition(2,3,4)

OUTPUT:

Addtion of 2 + 3 = 5

Addtion of 2 + 3 + 4 = 9

**19-Write a python program to demonstrate the concept of inheritance.**

# Inheritance in Python

One of the core concepts in [object-oriented programming](https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/) (OOP) languages is inheritance. It is a mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class. Inheritance is the capability of one class to derive or inherit the properties from another class.

## Python Inheritance Syntax

The syntax of simple inheritance in Python is as follows:

Class BaseClass:

{Body}

Class DerivedClass(BaseClass):

{Body}

class Student:  
 def \_\_init\_\_(self,Rollno,Name):  
 self.roll=Rollno  
 self.nm=Name  
 def print(self):  
 print("Rollno = ", self.roll, "Name = ", self.nm)  
class StudentInfo(Student):  
 def \_\_init\_\_(self, Rollno, Name,Age):  
 self.age=Age  
 super().\_\_init\_\_(Rollno,Name)  
 def print1(self):  
 print("Age = ",self.age)  
s1=StudentInfo(1,"Amit Kumar",19)  
s1.print()  
s1.print1()

**OUTPUT:**

Rollno = 1 Name = Amit Kumar

Age = 19

**20.Write a python program to demonstrate the concept of try, except and finally block.**

## **Exception Handling**

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

These exceptions can be handled using the try statement:

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The else block lets you execute code when there is no error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

try:  
 a=int(input("Enter Value of num1 = "))  
 b=int(input("Enter Value of num2 = "))  
 c=a/b  
except ZeroDivisionError:  
 print("You are dividing number by Zero.Division is not possible.")  
else:  
 print("C= ",c)  
finally:  
 print("End of Program")

**OUTPUT1:**

Enter Value of num1 = 10

Enter Value of num2 = 2

C= 5.0

End of Program

**OUTPUT2:**

Enter Value of num1 = 10

Enter Value of num2 = 0

You are dividing number by Zero.Division is not possible.

End of Program

**21. Write a python program for assertions.**

**Python Assertions** in any programming language are the debugging tools that help in the smooth flow of code. Assertions are mainly assumptions that a programmer knows or always wants to be true and hence puts them in code so that failure of these doesn’t allow the code to execute further.

## Assert Keyword in Python

In simpler terms, we can say that assertion is the boolean expression that checks if the statement is True or False. If the statement is true then it does nothing and continues the execution, but if the statement is False then it stops the execution of the program and throws an error.

***Syntax :****assert condition, error\_message(optional)*

def avg(marks):  
 assert len(marks) != 0,"List is empty."  
 return sum(marks)/len(marks)  
mark1 = []  
print("Average of mark1:",avg(mark1))  
mark2 = [55,88,78,90,79]  
print("Average of mark2:",avg(mark2))

OUTPUT:

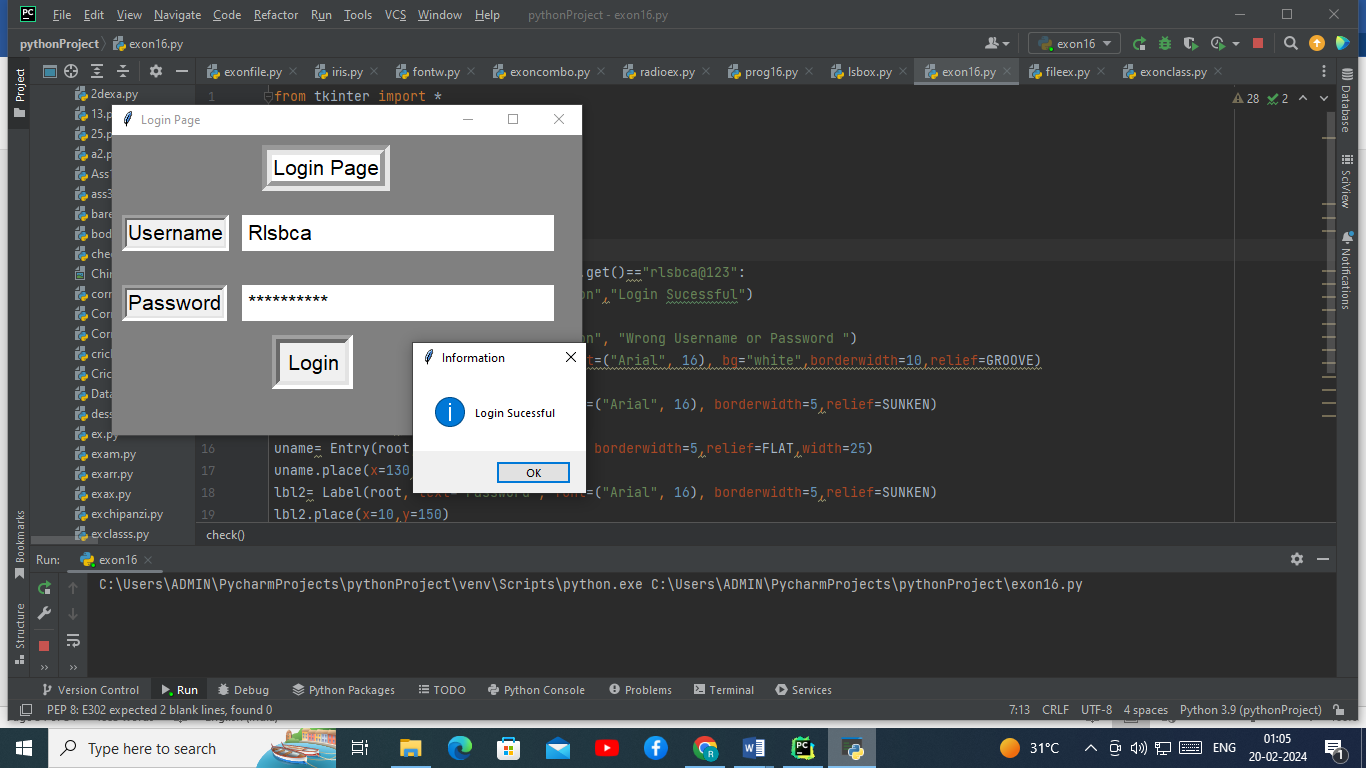
assert len(marks) != 0,"List is empty."

AssertionError: List is empty.

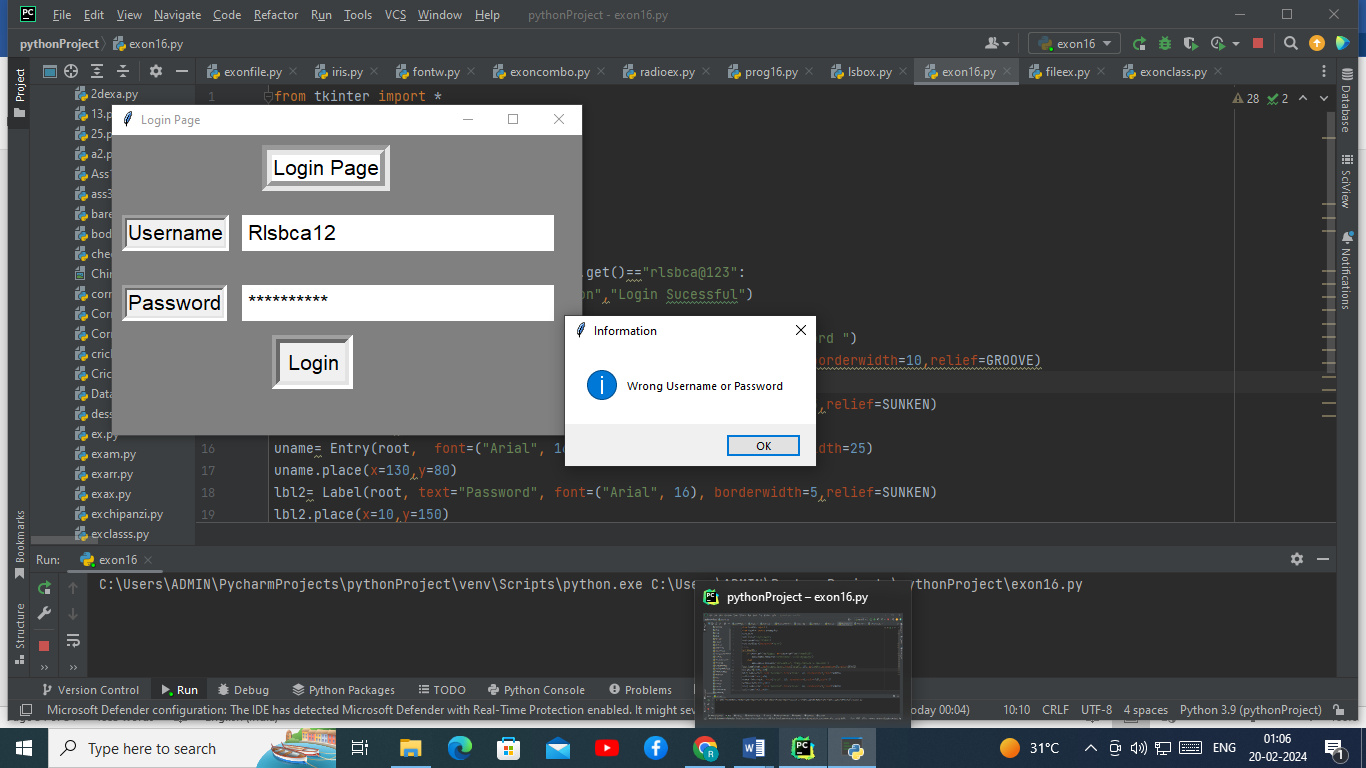
**22. Write Python code to demonstrate Login Page.**

from tkinter import \*  
from tkinter import messagebox  
root=Tk()  
root.title("Login Page")  
root.geometry("470x300")  
root.configure(background="gray")  
def check():  
 if uname.get()=="Rlsbca" and passw.get()=="rlsbca@123":  
 messagebox.showinfo("Information","Login Sucessful")  
 else:  
 messagebox.showinfo("Information", "Wrong Username or Password ")  
lbl= Label(root, text="Login Page", font=("Arial", 16), bg="white",borderwidth=10,relief=GROOVE)  
lbl.place(x=150,y=10)  
lbl1= Label(root, text="Username", font=("Arial", 16), borderwidth=5,relief=SUNKEN)  
lbl1.place(x=10,y=80)  
uname= Entry(root, font=("Arial", 16), borderwidth=5,relief=FLAT,width=25)  
uname.place(x=130,y=80)  
lbl2= Label(root, text="Password", font=("Arial", 16), borderwidth=5,relief=SUNKEN)  
lbl2.place(x=10,y=150)  
passw= Entry(root, show="\*", font=("Arial", 16), borderwidth=5,relief=FLAT,width=25)  
passw.place(x=130,y=150)  
submit= Button(root, text="Login", font=("Arial", 16), borderwidth=8,relief=SUNKEN,command=check)  
submit.place(x=160,y=200)  
root.mainloop()

**OUTPUT1:**



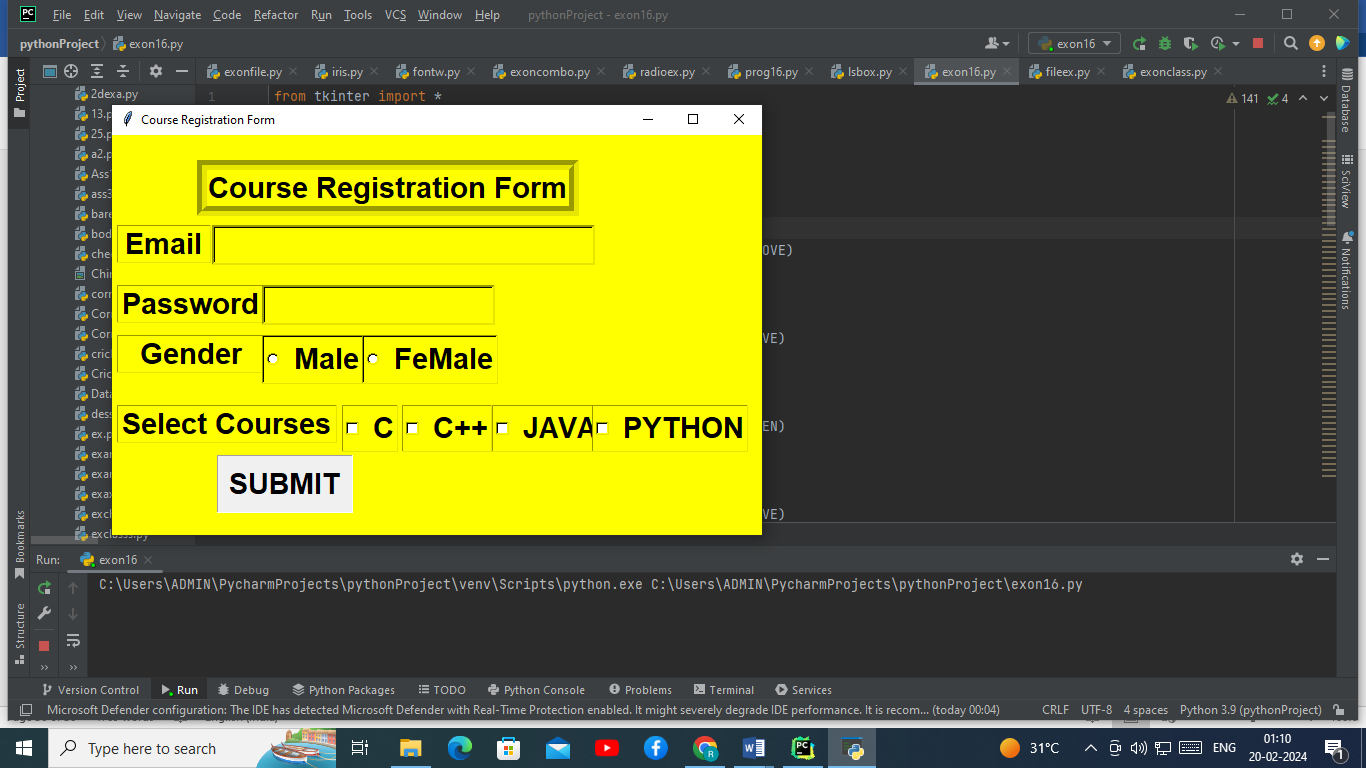
**OUTPUT2:**



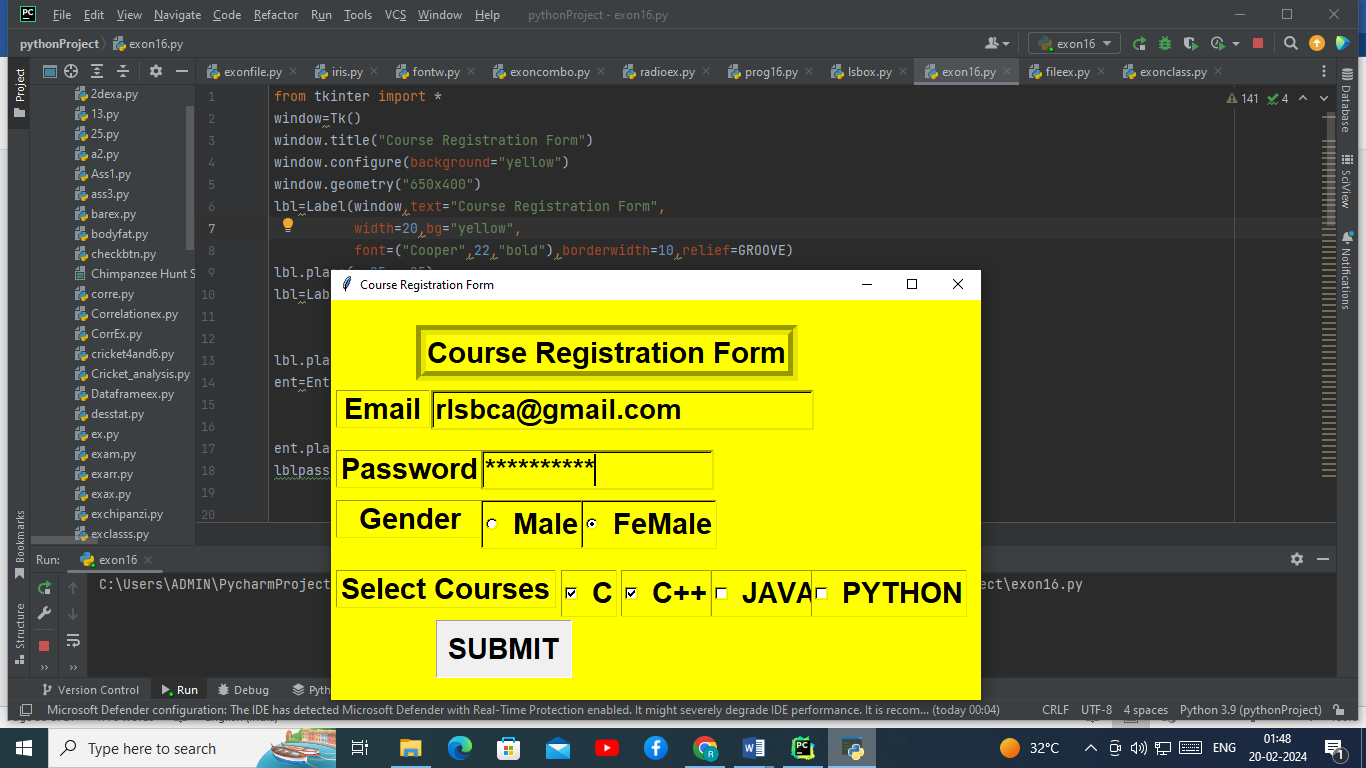
**23. Write python code to design registration form.**

from tkinter import \*  
window=Tk()  
window.title("Course Registration Form")  
window.configure(background="yellow")  
window.geometry("650x400")  
lbl=Label(window,text="Course Registration Form",  
 width=20,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=10,relief=GROOVE)  
lbl.place(x=85,y=25)  
lbl=Label(window,text="Email",  
 width=5,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
lbl.place(x=5,y=90)  
ent=Entry(window,  
 width=25,bg="yellow",  
 font=("Cooper",20,"bold"),borderwidth=3,relief=SUNKEN)  
ent.place(x=100,y=90)  
lblpass=Label(window,text="Password",  
 width=8,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
lblpass.place(x=5,y=150)  
entpass=Entry(window,show="\*",  
 width=15,bg="yellow",  
 font=("Cooper",20,"bold"),borderwidth=3,relief=SUNKEN)  
entpass.place(x=150,y=150)  
lblgen=Label(window,text="Gender",  
 width=8,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
lblgen.place(x=5,y=200)  
var=IntVar()  
R1=Radiobutton(window,text="Male",variable=var,value=1,  
bg = "yellow",font=("Cooper",22,"bold"),relief=SUNKEN)  
R1.place(x=150,y=200)  
R2=Radiobutton(window,text="FeMale",variable=var,value=2,  
bg = "yellow",font=("Cooper",22,"bold"),relief=SUNKEN)  
R2.place(x=250,y=200)  
lblcourse=Label(window,text="Select Courses",  
 width=12,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
lblcourse.place(x=5,y=270)  
chk1=IntVar()  
chk2=IntVar()  
chk3=IntVar()  
chk4=IntVar()  
c1=Checkbutton(window,text="C",variable=chk1,  
 onvalue=1,offvalue=0,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
c1.place(x=230,y=270)  
c2=Checkbutton(window,text="C++",variable=chk2,  
 onvalue=1,offvalue=0,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
c2.place(x=290,y=270)  
c3=Checkbutton(window,text="JAVA",variable=chk3,  
 onvalue=1,offvalue=0,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
c3.place(x=380,y=270)  
c4=Checkbutton(window,text="PYTHON",variable=chk4,  
 onvalue=1,offvalue=0,bg="yellow",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
c4.place(x=480,y=270)  
sub=Button(window,text="SUBMIT",  
 font=("Cooper",22,"bold"),borderwidth=1,relief=GROOVE)  
sub.place(x=105,y=320)  
window.mainloop()

**OUTPUT1:**



**OUTPUT2:**



**24.Demonstrate python code on tkinter canvas.Show different shapes.**

from tkinter import \*  
top = Tk()  
top.title("Canvas Drawing")  
Can = Canvas(top, bg="red", height=500, width=500)  
arc = Can.create\_arc(80, 10, 250, 130, start=0, extent=150, fill="#476042")  
arv2 = Can.create\_arc(80, 10, 250, 130, start=150, extent=215, fill="yellow")  
t1=Can.create\_text(270, 50, text="Arc", fill="white", font=('Helvetica 15 bold'))  
r1=Can.create\_rectangle(80, 150, 250, 250, fill="#476042")  
r2=Can.create\_rectangle(95, 165, 235, 230, fill="yellow")  
t2=Can.create\_text(300, 200, text="Rectangle", fill="white", font=('Helvetica 15 bold'))  
o1=Can.create\_oval(80,280, 260, 420, fill="#476042",outline="yellow")  
t3=Can.create\_text(290, 350, text="Oval", fill="white", font=('Helvetica 15 bold'))  
Can.pack()  
top.mainloop()

**OUTPUT:**

