

PYTHON PRATICALS



# Experiment No. 1

## Program to demonstrate use of Expression, Variables , Qutes , range and Basic math operations in python.

PROGRAM:-

x-5

print("x = "x) y=3

print("y = ".y) print("x+y =",x+y) print("x-y ="x-y) print("x\*y ="x\*y) print("x/y =",x/y) print("x%y =",x%y) print("x//y =",x//y) print("x\*\*y =",x\*\*y) print("x>y :",x>y)

print("x<y :",x<y)

print("x>=y:",x>=y)

print("x<=y :",x<=y)

print("x==y:",x==y)

print("xl=y :",xl=y)

print("x>3 and y>4",x>3 and y>4) print("x>3 or y>4 ",x>3 or y>4) Yaprint("not x>5 : ",not x>5 )

s= ''"What's the movie about?",I asked He repied,"Comics!""

print(s)

**Output:-**

x = 5

y = 3

X+y = 8

X-y = 2

x\*y = 15

x/y 1.6666666666666667

x%y = 2

x//y 1

x\*\*y = 125 X>y : True xxy : False X>=y : True X=y : False X==y : False x!=y : True

x>3 and y>4 False x>3 or y>4 True not x>5: True

"What's the movie about?" I asked He repied,"Comics!"

range(1,20) = [1, 2, 3, 4, 5, 6, 7, 8,9, 10, 11, 12, 13, 14, 15, 16, 17, 18,

19]

range(1,20,2) = [1, 3, 5, 7,9, 11, 13, 15, 17, 19]

range(1,20,-1) = (0

range(1,20,-1) = [20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9,8, 7, 6, 5,

4,3, 2]

range(1,20,-2) = [20, 18, 16, 14, 12, 10, 8, 6, 4, 2]

# Experiment No. 2

## Program to demonstrate python list, Strings and their methods.

Program:

s= [12,3.34,50,22,89,99.9]

print('s = 's) print('s[2]= 's[2]) s[2] = 4.5

print('s[2]= 's[2]) s.append(50) print('s= 's) s.insert(5,13) print('s= ',s)

print('s.count(50) : 's.count(50))

s.extend([12,45,67,76]) print('s= ',s) print('s.reverse(0: s.reverse()) print('s[:5:]: 's[:5:1)

print('s[2:]: 's[2:])

print('s[2:5:2]: 's[2:5:2])

print('s[5:1:-1]: 's[5:1:-1])

a= 'A quick brown fox jumped over a lazy dog' print('a.title(): a.title())

print('a.capitalize : ',a.capitalize()) print("a.count('e'): ",a.count('e'))

print("a.index('c'): ",a.index('c'))

print("a.swapcase() :",a.swapcase()) #a[2] = 'e' isn't allowed

**Output:**

s= [12, 3.34, 50, 22, 89, 99.9]

s[2]= 50

s[2]= 4.5

s= [12, 3.34, 4.5, 22, 89, 99.9, 50]

s= [12, 3.34, 4.5, 22, 89, 13, 99.9, 50]

s.count(50): 1

s= [12, 3.34, 4.5, 22, 89, 13, 99.9, 50, 12, 45, 67, 76]

s.reverse() : None

s[:5:]: [76, 67, 45, 12, 50]

s[2::]: [45, 12, 50, 99.9, 13, 89, 22, 4.5, 3.34, 12]

s[2:5:2]: [45, 50]

s[5:1:-1]: [99.9, 50, 12, 45]

atitle() : A Quick Brown Fox Jumped Over A Lazy Dog a.capitalize : A quick brown fox jumped over a lazy dog a.count('e'): 2

a.index('c'): 5

a,swapcase() : a QUICK BROWN FOX JUMPED OVER A LAZY DOG

# Experiment No. 3

## Program to demonstrate python Tuples,Dictionaries and Arrays.

##### Program

TUPLES: t3(2,1,6,4,8,5,9,8,14,52,5,1,2,5,4,8,78,59)

print("t=",t)

print("Element at index 4=",t[4]) #t[6]=60

... direct value assign cannot be possible

print("How many times 5 appear in tuple=",t.count(5)) print("Index of 9=",t.index(9))

**OUTPUT:**

t= (2, 1, 6, 4, 8, 5, 9, 8, 14, 52, 5, 1, 2, 5, 4, 8, 78, 59)

t= 8

t= 3

t= 6

**DICTIONARIES:**

x={"A":12,"B":20,"C":72,"D":45,"E":34,"F":82,"G":90}

print("x=",x)

print("All items in x=",x.items()) print("Keys in x=",x.keys()) print("Values in x=",x.values()) print("Value of B=",x.get("B") print("Deleted element=",x.pop("G"))

**OUTPUT:**

x= {'A': 12, 'B': 20, 'C': 72, 'D': 45, 'E': 34, 'F': 82, 'G': 90)

Al items in x= dict\_items([('A', 12), ('B', 20), ('C', 72), ('D', 45), ('E', 34), ('F', 82), ('G', 90)1

Keys in x= dict\_keys(['A', 'B', 'C', 'D', 'E', 'F', 'G']) Values in x= dict\_values([12, 20, 72, 45, 34, 82, 90]) Value of B= 20

Deleted element= 90

##### ARRAY:

from array import array a=array('i',[3,4,7,6,8,9,1,2,5])

print("a=",a)

print("Element at index 2=",a[2]) a[4]=16

print("Inserted value at index 4 is=",a[4]) #a[5]=5.5

... float value not allowed

**OUTPUT:**

a= array('i', [3, 4, 7, 6, 8, 9, 1, 2, 5])

Element at index 2= 7

a= 16

from array import array y=array('f',[2.6,4.8,5.5,9.1,1.0])

print("y=",y)

print("Element at index 3=",y[3]) y[4]=6.0

print("Inserted value at index 4 is=",y[4]) #y[2]=5

...integer value not allowed

OUTPUT: VE array('f', [2.5999999046325684, 4.800000190734863, 5.5, 9.100000381469727, 1.0])

Element at index 3= 9.100000381469727

Inserted value at index 4 is= 6.0

# Experiment No. 4

### Program to demonstrate Python decision and loop control instructions.

**Program** 1:

g=input("Enter gender:") age=int(input("Enter age:"))

if((g'm' and age>=21) or (g'f and age>-18)): print(" eligible for marriage")

else:

print(" not eligible for marriage")

**output :**

Enter gender:m Enter age:21 eligible for marriage Enter gender:f Enter age:18 eligible for marriage **Program 2:**

day-int(input("enter day number:") if(day 1):

print("Monday") elif(day 2): print("Tuesday") elif(day-3): print("Wednesdat") elif(day-4): print("Thursday")

elif(day-5): print("Friday") elif(day-6): print("Saturday") elif(day=7): print("Sunday")

else: print("invalid") output :

enter day number:2

Tuesday

>>>

RESTART: C:\Usersluser\Downloads\p4.py enter day number:5

Friday

>>>

RESTART: CAUscrsuser\Downloads\p4.py enter day number:8

invalid **Program** 3:

n=int(input("Enter number:") for i in range(2,n):

if (n%2==0):

print("number is not prime") break

else:

print("number is prime")

##### output :

RESTART: C:\Users'user\Downloads\p4.py Enter number:13

number is prime

**Program** 4: n=int(input("Enter number:")) i=2

while(i<=n-1):

if (n%2=0):

print("number is not prime") break

i=i+1 else:

print("number is prime") **output :**

RESTART: C:\Users\user\Downloads\p4.py Enter number:13

number is prime

RESTART: C:\Users\user\Downloads\p4.py Enter number:10

number is not prime

# Experiment No. 5

### Program to demonstrate error-handling in Python.

try:

d-float(input("Enter Number d:") print("d:",d)

t-float(input("Enter Number t:")) print("t:",t)

if((d<0) or (t<0)):

raise ValueError() v=d/t

except ZeroDivisionError:

print("You Dividing by Zero") except ValueError:

print("You Enter the Char or Entered value is less than zero") else:

print("Everything is fine") print("v:",d/t)

finally:

print("Done")

**Output:**

Python 3.8.1 (tags/v3.8.1:lb293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (In tel)) on win32

Type "help", "copyright", "credits" or "license ()" for more information.

>>>

RESTART: D:\Python Progam\Sth.py -

Enter Number d:5 d: 5.0

Enter Number t:2 t: 2.0

Everything is fine v: 2.5

Done

>>>

# Experiment No. 6

#### Program to demonstrate user defined functions in Python.

def speed(d,t):

v=d/t return v

print("Velocity:",speed(1078,38)) print("Velocity:",speed(t-10,d%3D90)) def Num(e,b,c,d):

print("value a:",e)

print("value b:",b)

print("value c:",c)

print("value d:",d)

Num(1,2,3,4)

def Volume(pi,r):

a=pi\*r\*r return a

print("Area of circle:",Volume(3.14,38)) def Area(pi,r,h):

A-pi\*r\*r\*h return A

print("Area of Cylinder:",Area(3.14,6,5))

##### Output:

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) (MSC v.1916 32 bit (In

tel)) on win32

Type "help", "copyright", "credits" or "1icense ()" or more information.

>>>

- RESTART: D:\Python Progam\6th.py -m Velocity: 28.36842105263158

Velocity: 9.0

value a: 1

value b: 2

value c: 3

value d: 4

Area of circle: 4534.16

Area of Cylinder: 565.1999999999999

# Experiment No. 7

#### Program to introduce Python as an Object Oriented Programming (0OP)

class Rect :

def init (self,l,w):

self.length= 1

self.width= w

#self. x-66(private variable) def display(self):

print("length=",self.length,"width=",self.width) def area(self):

return self.length\*self.width def perimeter(self):

return 2\*(self.length\*self.width)

def scale(self, a,b):

self.length= a\*self.length

self.width= b\*self.width def del (self):

print("class rectangle deleted")

window-Rect(6,4)

window.display()

print("area of rectangle=",window.area()) print("parameter of rectangle=",window.perimeter(0) window.scale(2,3)

window.display() print(window.length) class Test: def\_init\_(self):

pass

def fun(self):

print(" this is fun() method") @staticmethod

def stat():

print("stat() is static method") @classmethod

def clas(cls):

print("this is class \n Class Name=",cls. t=Test()

t.fun() Test.stat() Test.clas()

**Output:**

Python 3.8.1 (tags/ v3.8.1: 1b293b6, Dec 18 2019, 22:39:24) (MSC v. 1916 32 bit (In

tel)] on win32

Type " help", " copyright", " credits" or " license ()" for more information.

>>>

\*\*\*\*\*\* RESTART: D:\exp7.py \*\*\* ==== length = 6 width = 4

area of rectangle 24 parameter of rectangle = 48 length 12 width- 12

12

this is fun () method stat () is static method this is class

Class Name = Test

>>>

# Experiment No. 8

#### class Program Child to (object demonstrate): inheritance and overriding in Python.

def init\_ (self, name): self.name = name

def getName (self):

return self.name

def isEmployee (self):

return False

def method1 (self):

print (' methodl of class Child ') class Father (Child):

def method2 (self):

print (' methodl of class Father ') def is Employee (self):

return True

class Grandfather (Father, Child): def isEmployee (self):

return False

emp = Child ("Child")

print (emp.getName (), emp.isEmployee () f = Father ("Father")

print (emp.getName (), emp.isEmployee ()) e = Father ("GarndFather")

print (emp.getName (), emp.isEmployee () emp.method10

f.method20) e.method20

f.method10

**Output:**

Python tel)) on 3.8.1 win32 (tags/ v3.8.1: 1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (In A Type " help", " copyright", " credits" or " license ()" for more information.

Child False ========== RESTART: D:\ Python Progam\ inhitance.py ======== Child False

Child False

methodl of class Child methodl of class Father methodl of class Father methodl of class Child

##### Assignment No:1

1. **List different packages and classes used to design GUI in Python. Ans) We use the tkinter package to design GUI in Python.**

**There are several predefined classes that make our work quite easy and help us to program faster such as we can create a button using the Button() class.**

**Other classes include Label, Canvas, Checkbutton, Entry, Frame, Listbox, Menubutton, Radiobutton, Scrollbar,etc. All of them are used to easily make widgets on the GUI.**

1. **Explain tkinter package.**

**Ans) Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI**

**applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.**

**Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following**

**steps −**

1. **Import the Tkinter module.**
2. **Create the GUI application main window.**
3. **Add one or more of the above-mentioned widgets to the GUI application.**
4. **Enter the main event loop to take action against each event triggered by the user.**

**3) List various classes in tkinter used for GUI components in Windows, and explain any two classes in detail.**

**Ans) The various classes used in tkinter for GUI components in Windows are Button, Label, Canvas, Checkbutton, Entry, Frame, Listbox, Menubutton, Radiobutton, Scrollbar,etc. All of them are used to easily make widgets on the GUI.**

1. **Label: This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.**

**It is also possible to underline part of the text (like to identify a keyboard shortcut) and span the text across multiple lines.**

**Syntax:**

**Here is the simple syntax to create this widget −**

**w = Label ( master, option, ... )**

1. **Button: The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.**

**Syntax:**

**Here is the simple syntax to create this widget −**

**w = Button ( master, option=value, ... )**

1. **What are different layouts used in GUI design?**

**Ans) Pack is the easiest to use of the three layouts of Tk and Tkinter. Instead of having to declare precisely where a widget should**

**appear on the display screen, we can declare the positions of widgets with the pack command relative to each other. The pack command takes care of the details. Though the pack command is**

**easier to use, this layout managers is limited in its possibilities compared to the grid and place mangers. For simple applications it is definitely the manager of choice. For example simple applications like placing a number of widgets side by side, or on top of each other.**

**Grid is in many cases the best choice for general use. While pack is sometimes not sufficient for changing details in the layout, place gives you complete control of positioning each element, but this makes it a lot more complex than pack and grid. Using the grid manager means that you create a widget, and use the grid method to tell the manager in which row and column to place them. The size of the grid doesn't have to be defined, because the manager automatically determines the best dimensions for the widgets used.**

**The Place geometry manager allows you explicitly set the position and size of a window, either in absolute terms, or relative to another window. The place manager can be accessed through the place method. It can be applied to all standard widgets.**

1. **Write a sample code to demonstrate any one GUI layout in python. Ans) Program implementing pack layout:**

**import tkinter as tk**

**root = tk.Tk()**

**w = tk.Label(root, text="Red Sun", bg="red", fg="white") w.pack(fill=tk.X)**

**w = tk.Label(root, text="Green Grass", bg="green", fg="black") w.pack(fill=tk.X)**

**w = tk.Label(root, text="Blue Sky", bg="blue", fg="white") w.pack(fill=tk.X)**

**tk.mainloop()**

**Assignment No:2**

* 1. **What does it mean by event handling in GUI?**

**Ans) A Tkinter application runs most of its time inside an event loop, which is entered via the mainloop method. It waits for events to happen. Events can be key presses or mouse operations by the user.**

**Tkinter provides a mechanism to let the programmer deal with events. For each widget, it's possible to bind Python functions and methods to an event. So for every event that takes place we can do various actions which is known as event handling in a GUI.**

**Here's an example:**

**Capturing clicks in a window from Tkinter import \***

**root = Tk()**

**def callback(event):**

**print "clicked at", event.x, event.y**

**frame.bind("<Button-1>", callback) frame.pack()**

**root.mainloop()**

**Each time you click, a message like “clicked at 44 63” is printed to the console window.**

* 1. **Explain bind(...) methodnto bind an event to a GUI component.**

**Ans) Tkinter provides a mechanism to let the programmer deal with events. For each widget, it's possible to bind Python functions and methods to an event.**

**The function is written like this: widget.bind(event, action)**

**For example you can bind a left mouse click on a button to call a function : Button.bind("<Button- 1>",functionName)**

**And just like that the function will be called.**

* 1. **List different event strings used to bind an event to a GUI compenent. Here's a summary of the most common event strings explained:**

**<Button-1> Button 1 is the leftmost button, button 2 is the middle button (where available), and button 3 the rightmost button.**

**<B1-Motion> The mouse is moved, with mouse button 1 being held down (use B2 for the middle button, B3 for the right button).**

**<ButtonRelease-1> Button 1 was released. This is probably a better choice in most cases than the Button event, because if the user accidentally presses the button, they can move the mouse**

**off the widget to avoid setting off the event.**

**<Double-Button-1> Button 1 was double clicked. You can use Double or Triple as prefixes.**

**<Enter> The mouse pointer entered the widget (this event doesn’t mean**

**that the user pressed the Enter key!).**

**<Leave> The mouse pointer left the widget.**

**<FocusIn> Keyboard focus was moved to this widget, or to a child of this widget.**

**<FocusOut> Keyboard focus was moved from this widget to another widget.**

**<Return> The user pressed the Enter key**

* 1. **Write a program code to demonstrate event handling. Ans)**

**Capturing clicks in a window from Tkinter import \***

**root = Tk()**

**def callback(event):**

**print "clicked at", event.x, event.y**

**frame.bind("<Button-1>", callback) frame.pack()**

**root.mainloop()**

**Each time you click, a message like “clicked at 44 63” is printed to the console window.**