Week 1:

Installation and Environment set up of Python and Programs on Data types

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

• Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)

• Win 9x/NT/2000

• Macintosh (Intel, PPC, 68K)

• OS/2

• DOS (multiple versions)

• PalmOS

• Nokia mobile phones

• Windows CE

• Acorn/RISC OS

• BeOS

• Amiga

• VMS/OpenVMS

• QNX

• VxWorks

• Psion

• Python has also been ported to the Java and .NET virtual machines

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

You can download Python documentation from https://www.python.org/doc/. The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

• Open a Web browser and go to https://www.python.org/downloads/.

• Follow the link to download zipped source code available for Unix/Linux.

• Download and extract files.

• Editing the Modules/Setup file if you want to customize some options.

• run ./configure script

• make

• make install

This installs Python at standard location /usr/local/bin and its libraries at /usr/local/lib/pythonXX where XX is the version of Python.

Windows Installation

Here are the steps to install Python on Windows machine.

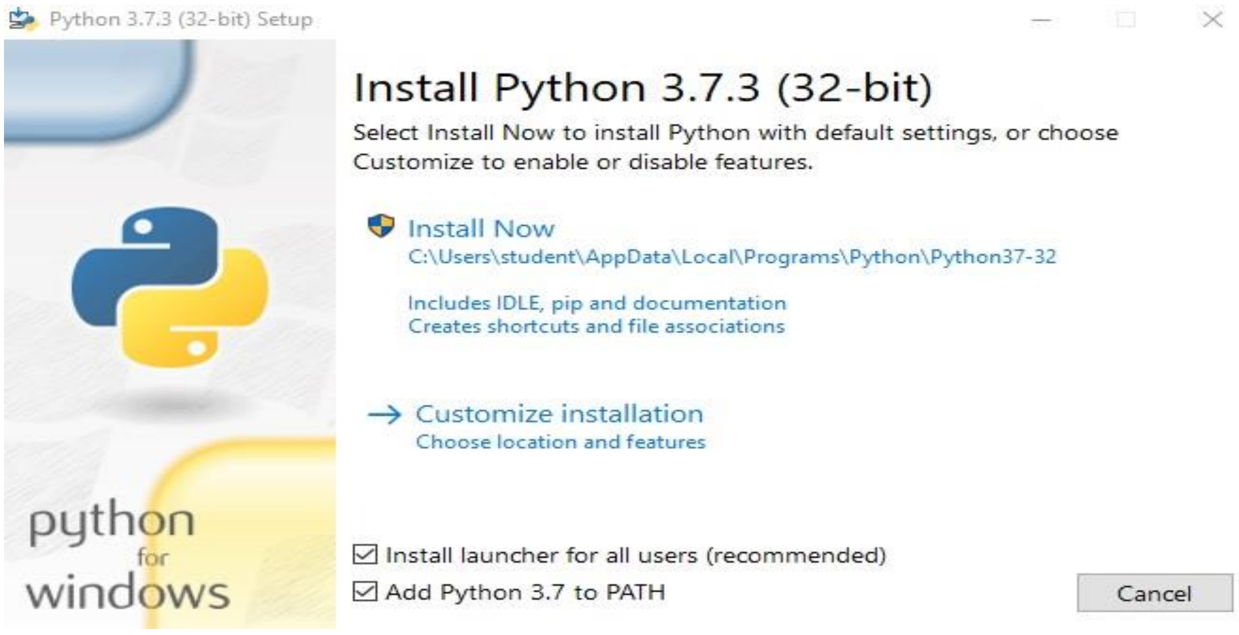
•Open a Web browser and go to https://www.python.org/downloads/.

•Follow the link for the Windows installer python-XYZ.msi file where XYZ is the version you need to install.

•To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.

•Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Installation of Python



Python 3.7.3 (32-bit) Setup

Install Python 3.7.3 (32-bit)

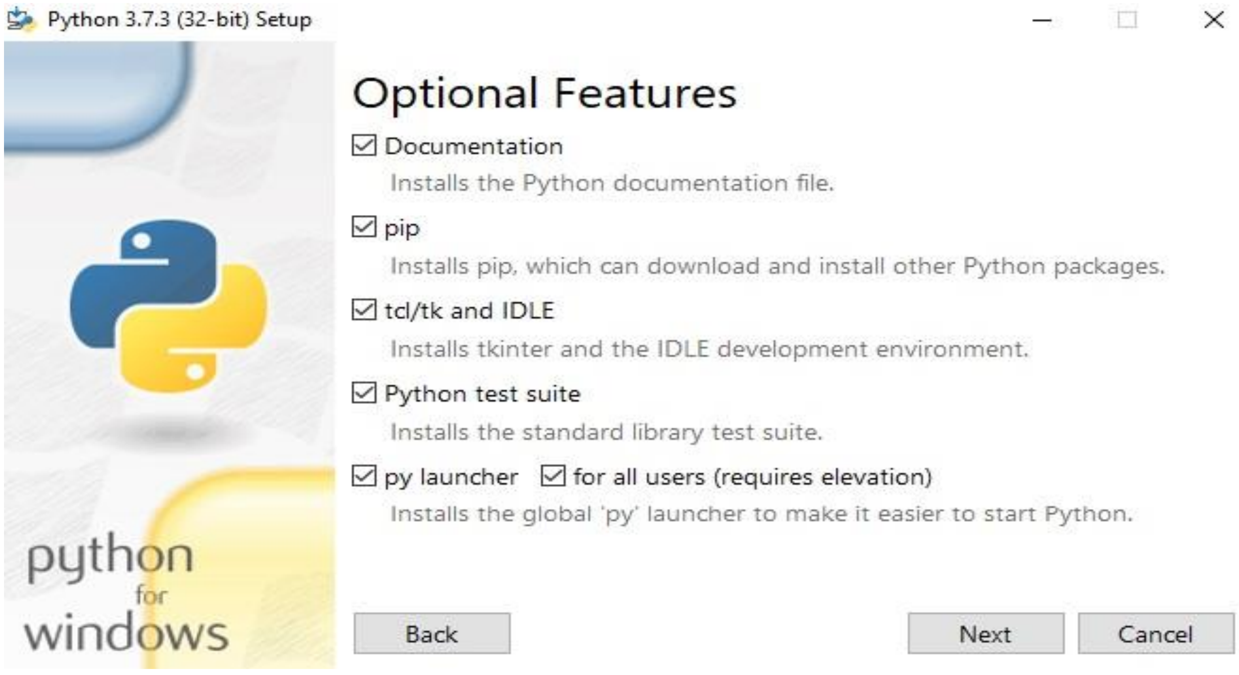
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\student\AppData\Local\Programs\Python\Python37-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH



Python 3.7.3 (32-bit) Setup

Optional Features

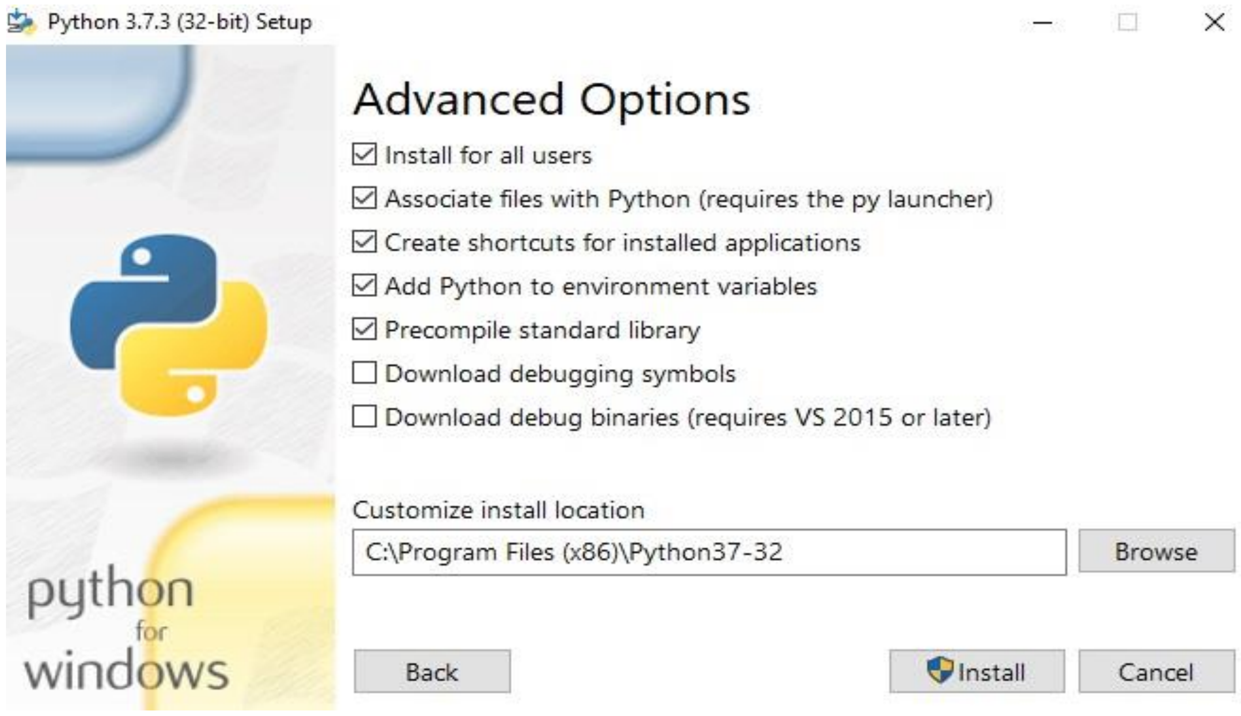
☑ Documentation
Installs the Python documentation file.

☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)
Installs the global 'py' launcher to make it easier to start Python.

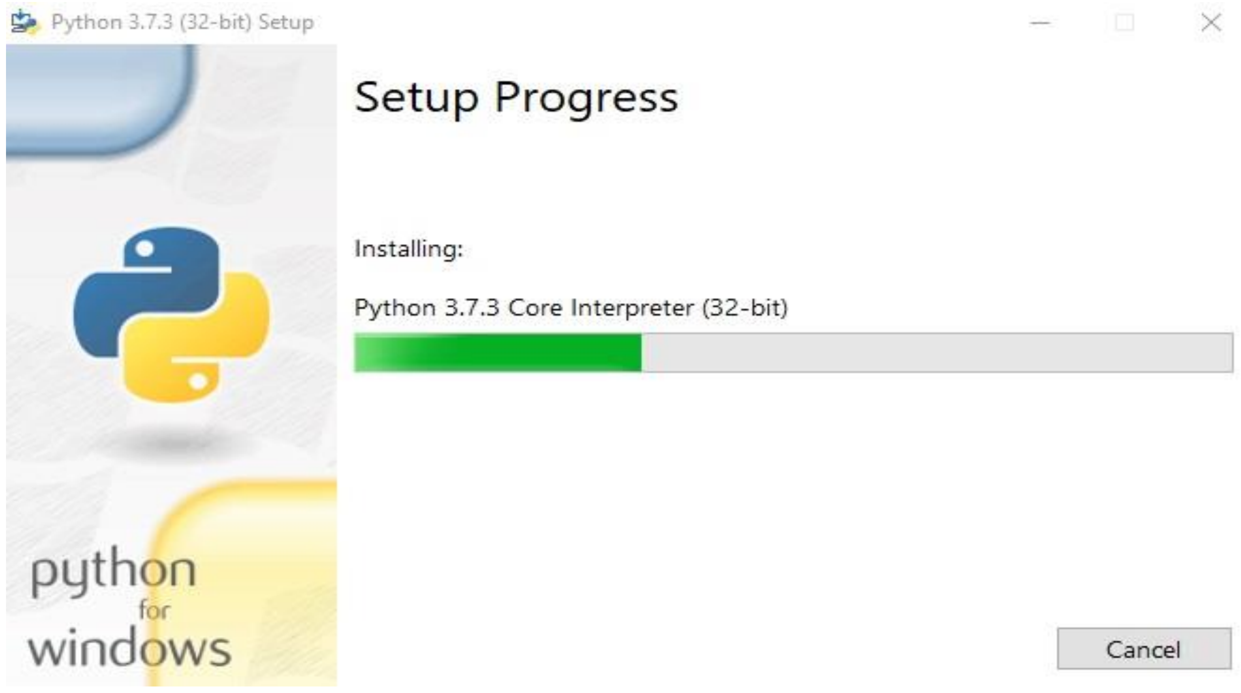


Advanced Options

- [✓] Install for all users
- [✓] Associate files with Python (requires the py launcher)
- [✓] Create shortcuts for installed applications
- [✓] Add Python to environment variables
- [✓] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2015 or later)

Customize install location

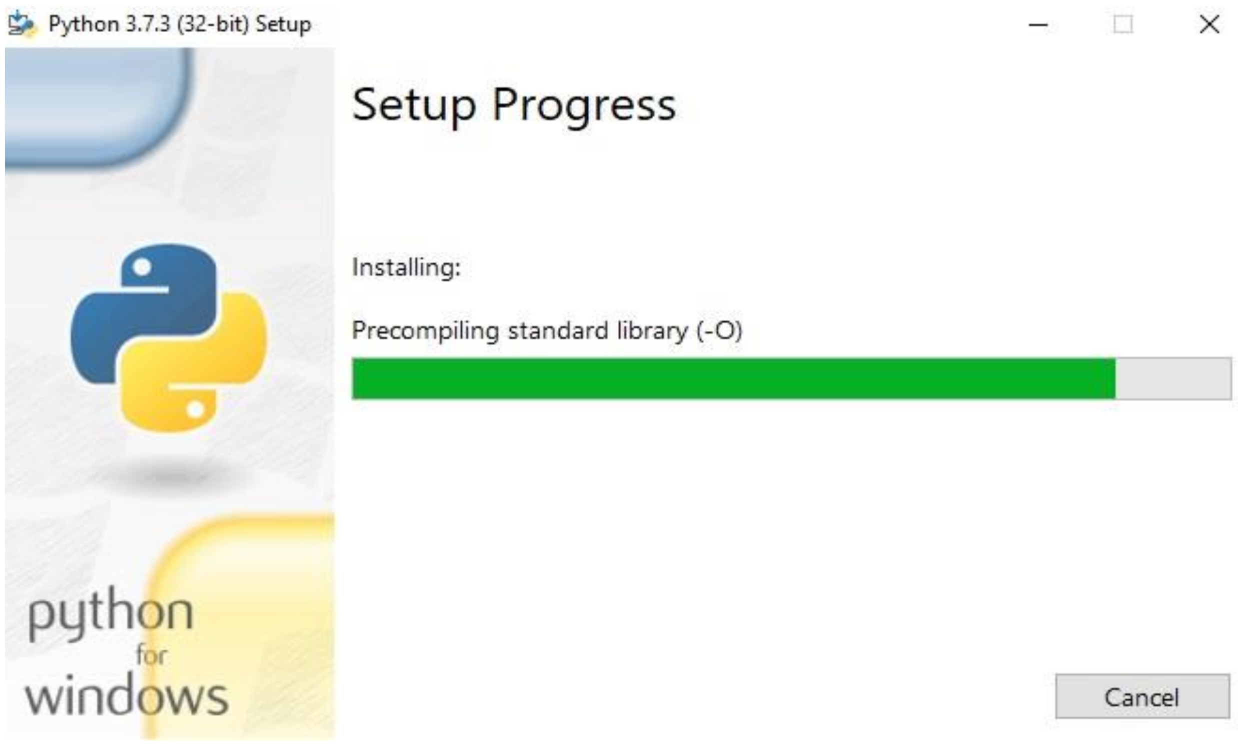
C:\Program Files (x86)\Python37-32



Setup Progress

Installing:

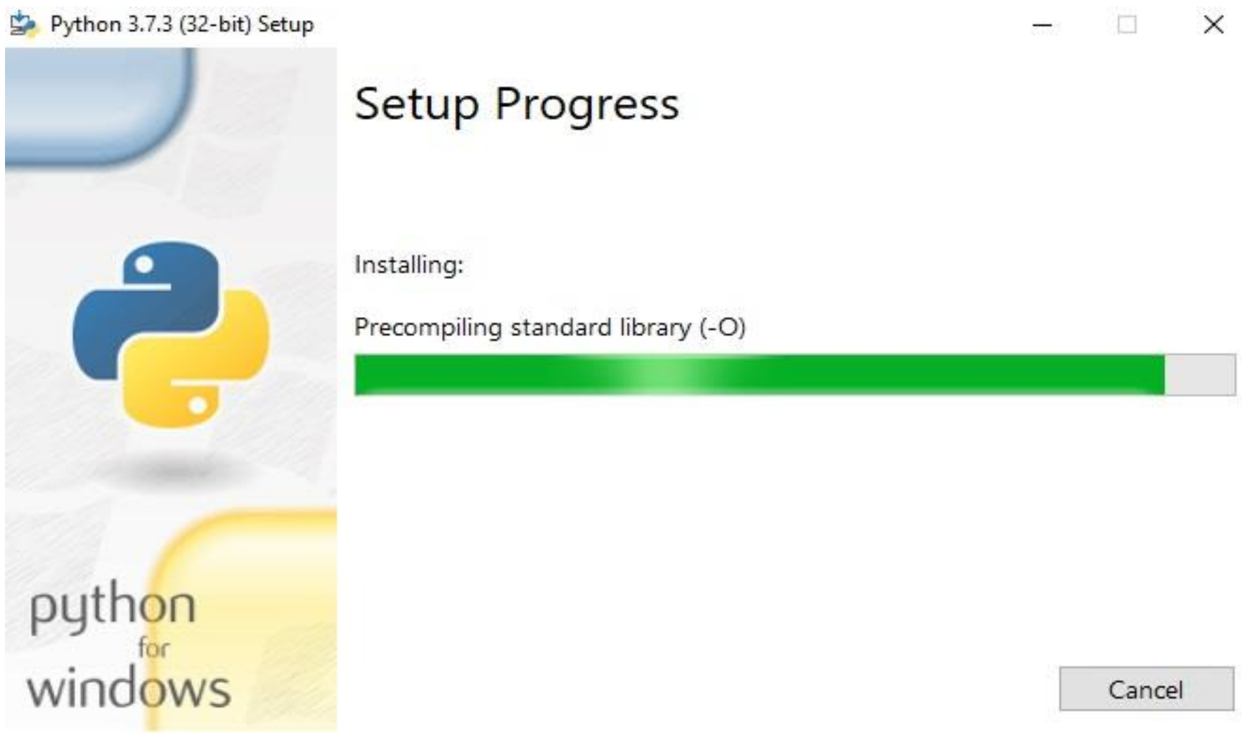
Python 3.7.3 Core Interpreter (32-bit)



Setup Progress

Installing:

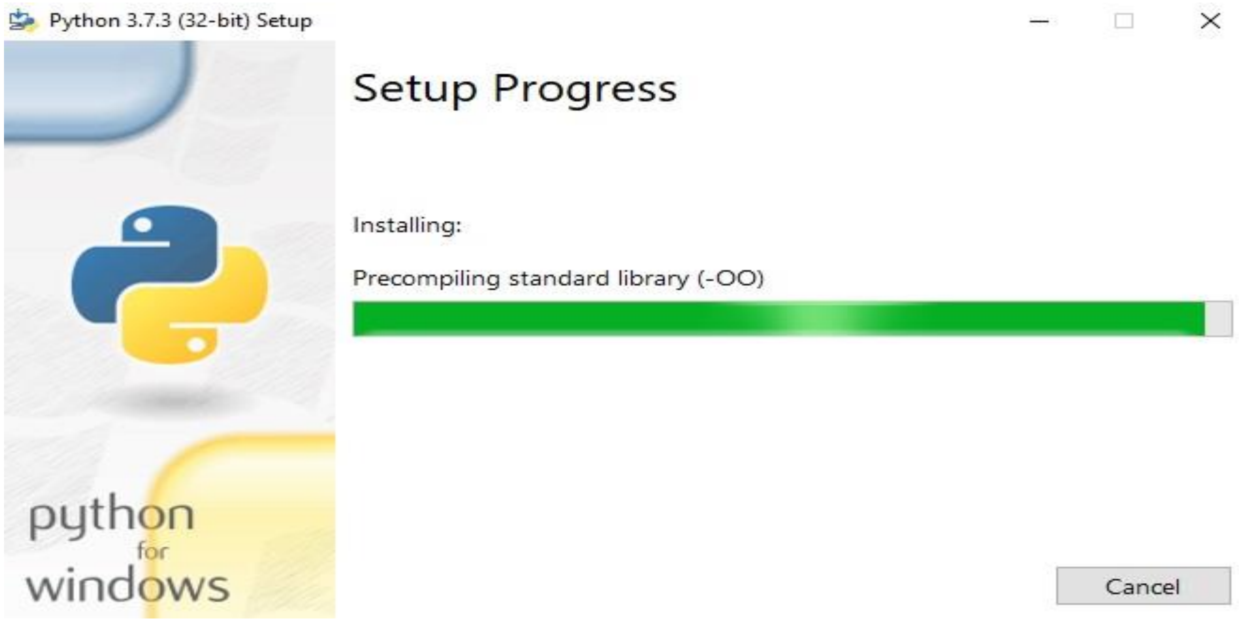
Precompiling standard library (-O)



Setup Progress

Installing:

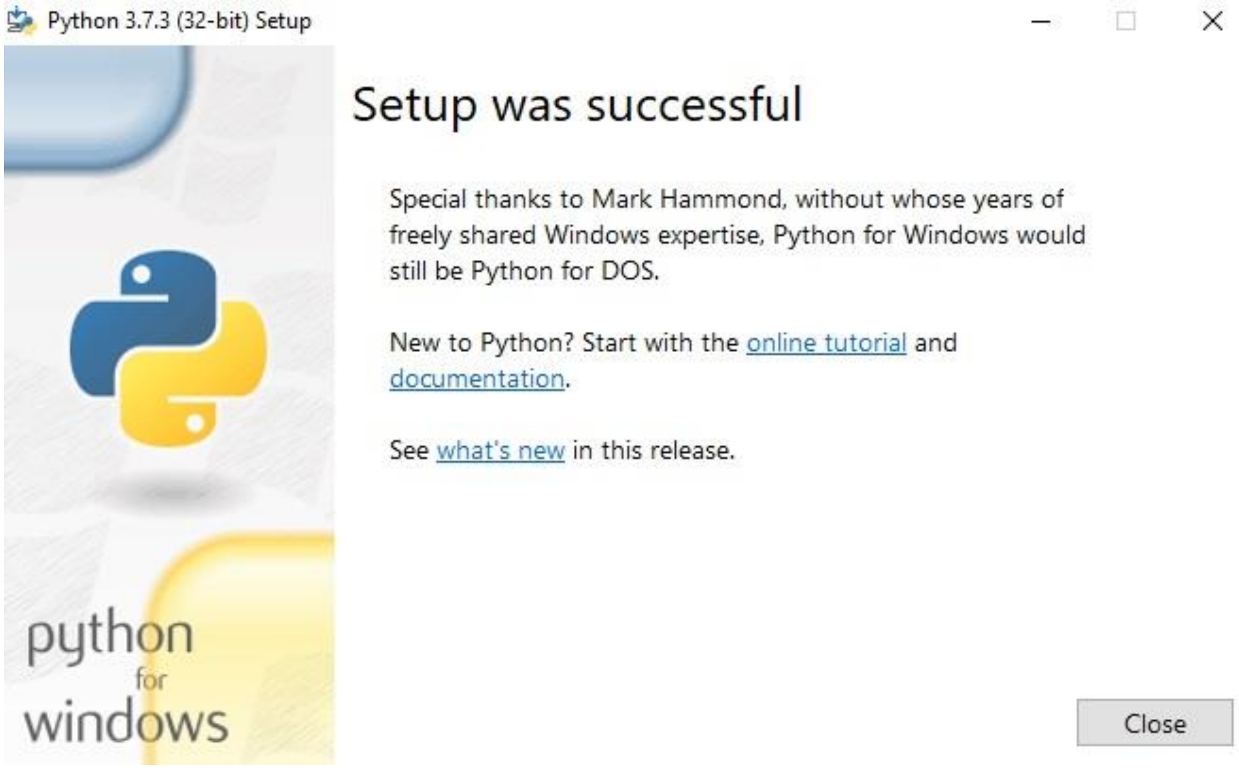
Precompiling standard library (-O)



Setup Progress

Installing:

Precompiling standard library (-OO)



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

Programs on Data types

1.Numbers

a) Integers

```
>>> a=5
>>> b=c=5
>>> a
5
>>> b
5
>>> c
5
>>> a+b
10
>>> a+b+c
15
>>> type(a)
<class 'int'>
>>> a/c
1.0
>>> a//c
1
>>> a*b
25
```

b) Floating-Point Numbers

```
>>>a= 4.2
>>>a
4.2
>>> type(a)
<class 'float'>
>>> b=4.
>>>b
4.0
>>> a+b
8.2
```

c) Complex Numbers

```
>>> d=2+4j
>>> type(d)
<class 'complex'>
>>> a=2
>>> b=3j
>>> c=a+b
>>> c
```

```
(2+3j)
 d)
```

2. Strings

```
>>> str="cvsr"
>>> str
'cvsr'
>>> type(str)
<class 'str'>
>>> str1="agi"
>>> str+str1
'cvsragi'
>>> print('a\
... b\
... c')
a... b... c
>>> print('foo\\bar')
foo\bar
>>> print('foo\tbar')
foo     bar
>>> print("a\tb")
a       b
>>> print('\u2192 \N{rightwards arrow}')
→ →
>>> print('foo\nbar')
foo
bar
>>> print(r'foo\nbar')
foo\nbar
>>> print(R'foo\\bar')
foo\\bar
>>> print('''This string has a single (') and a double (") quote.''')
This string has a single (') and a double (") quote.
>>> print("This string contains a double quote (\") character.")
This string contains a double quote (") character.
>>> print('This string contains a single quote (\') character.')
This string contains a single quote (') character.
```

3.LIST

```
>>> a=[]
>>> a
[]
>>> a=[1,'b',5.5,'c']
>>> a
[1, 'b', 5.5, 'c']
>>> type(a)
<class 'list'>
>>> a[1]
'b'
>>> a[-1]
'c'
>>> a[1]=10
>>> a
[1, 10, 5.5, 'c']
```

4. Tuple

```
>>> a=( )
>>> a
( )
>>> a=(5,'b',10,5.8)
>>> a
(5, 'b', 10, 5.8)
>>> type(a)
<class 'tuple'>
>>> a[1]
'b'
>>> a[2]
10
>>> a[-1]
5.8
```

5.Dictionary

```
>>> d={}
>>> d
{}
>>> type(d)
<class 'dict'>
>>> dict={"name":"cvsr","number":10,"loc":"hyderabad"}
>>> dict
{'name': 'cvsr', 'number': 10, 'loc': 'hyderabad'}
```

```
>>> type(dict)
<class 'dict'>
>>> dict["name"]
'cvsr'
>>> dict["number"]=88
>>> dict
{'name': 'cvsr', 'number': 88, 'loc': 'hyderabad'}
```

Week 2:

Programs on Standard I/O, Operators and Expressions

1.Standard I/O Programs:

1.Program to find maximum of two numbers

```
a= int(input('Enter a value'))

b= int(input('Enter b value'))

if a>b:

    print('a is max')

else:

    print('b is max')
```

Output:

```
Enter a value2

Enter b value3

b is max
```

2. Program to find whether given number is even or not

```
a=int(input("Enter a Number : "));

if(a%2==0):

    print("Number is Even")

else:

    print("Number is Odd")
```

Output:

```
Enter a Number : 10

Number is Even
```

3. Program to print first n numbers

```
a=int(input("enter a value"))
for i in range(1,a):
    print(i)
```

Output:

```
enter a value10
1
2
3
4
5
6
7
8
9
```

4. Program to print sum of n numbers

```
sum=0
a=int(input("enter a value"))
for i in range(1,a+1):
    sum=sum+i
print(sum)
```

Output:

```
enter a value10
55
```

```
i=1
sum=0
a=int(input("enter a value"))
while i<a:
    sum=sum+i
    i=i+1
print(sum)
```

Output:

```
Enter a value 10
46
```

5. Program to check number is prime or not

```
c=0
a=int(input("enter the number"))
for i in range(1,a+1):
    if(a%i==0):
        c=c+1
if(c==2):
    print(a,'is prime'
          )
else:
    print("not prime")
```

Output:

```
enter the number5
5 is prime
```

```
c=0
a=int(input("enter the first number"))
b=int(input("enter the second number"))
for i in range(a,b):
    for j in range(1,i+1):
        if(i%j==0):
            c=c+1
    if(c==2):
        print(i,'prime number')
```

Output:

```
enter the first number10
enter the second number20
11, prime number
13, prime number
17, prime number
19, prime number
```

6.Program to print sum of even numbers in given range

```
s=0
for i in range(100,200,2):
    if(i%2==0):
        s=s+i
print('total sum=',s)
```

Output:

```
total sum= 7450
```

7.Program to find factorial of given number

```
a=int(input('enter the number'))
fact=1
if(a<0):
   print("no factorial for -ve number")
elif a==0:
   print("factorial for 0 is 1")
else:
   for i in range(1,a+1):
      fact=fact*i
print("factorial of given number is",fact)
```

Output:

```
enter the number5
factorial of given number is 120
```

8. Program to evaluate 1+1/2+1/3….+1/n

```
s=0
a=int(input("enter a number"))
for i in range(1,a+1,1):
   s=s+1/i
print(s)
```

Output:

```
enter a number5
2.283333333333333
```

9.Program to print multiplication for 6, 8 and 10

```
for i in range(1,10+1):
    print(6,"*",i,"=",6*i)
for j in range(1,10+1):
    print(8,"*",j,"=",8*j, " ", 10,"*",j,"=",10*j)
```

Output:

```
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
8 * 1 = 8   10 * 1 = 10
8 * 2 = 16   10 * 2 = 20
8 * 3 = 24   10 * 3 = 30
8 * 4 = 32   10 * 4 = 40
8 * 5 = 40   10 * 5 = 50
8 * 6 = 48   10 * 6 = 60
8 * 7 = 56   10 * 7 = 70
8 * 8 = 64   10 * 8 = 80
8 * 9 = 72   10 * 9 = 90
```

```
8 * 10 = 80   10 * 10 = 100
```

10.Program to find Fibonacci series

```
n=int(input("enter the number"))
a=0
b=1
c=0
print(a,b)
for i in range(1,n+1):
    c=a+b
    a=b
    b=c
    print(c)
```

Output:

```
enter the number5
0 1
1
2
3
5
8
```

11.Program to check whether given number is palindrome or not

```
n=int(input("enter number"))
n1=n
r=0
s=0
```

```
while(n1>0):
    r=n1%10
    s=s*10+r
    n1=n1//10
if(s==n):
    print(n,"palindrome")
else:
    print(n,"not palindrome")
```

Output:

```
enter number121
121 palindrome
```

2. Programs on Operators

1. Arithmetic Operators

+, -, *, /, //, %, **

```
>>> a=20
>>> b=10
>>> a+b
30
>>> a-b
10
>>> a*b
200
>>> a/b
2.0
>>> a//b
2
>>> a%b
0
>>> a**b
10240000000000
```

2. Relational Operators

==, !=, >, <, >=, <=

```
>>> a=10
>>> b=20
>>> a==b
False
>>> a!=b
True
>>> a>b
False
>>> a<b
True
>>> a>=b
False
>>> a<=b
True
```

3. Assignment Operators

=, +=, -=, /=, %=, **=, //=, &=, /=, ^=, >>=, <<=

```
>>> a=10
>>> b=20
>>> c=5
>>> a=c
>>> a
5
>>> a+=b
>>> a
25
>>> a-=b
>>> a
5
>>> b/=c
>>> b
4.0
>>> a%=c
>>> c
5
```

```
>>> a
0
>>> c**=b
>>> c
625.0
>>> b/=c
>>> b
0.0064
>>> a=10
>>> b=20
>>> b//=a
>>> b
2
>>> a&=b
>>> a
2
>>> a^=b
>>> a
0
>>> b>>=a
>>> b
2
>>> b<<=a
>>> b
2
```

4.Logical Operators

and, or, not,

```
>>> a=10
>>> b=20
>>> a>2 and b<10
False
>>> a>2 or b<10
True
>>> not (a>2 and b>10)
False
>>> a='true'
>>> b='false'
>>> a and b
```

```
'false'
>>> a or b
'true'
>>> not a
False
>>> not b
False
```

5. Bitwise Operators
&, |, ~, ^, >>, <<

```
>>> a=2
>>> b=3
>>> a&b
2
>>> a|b
3
>>> ~a
-3
>>> a^b
1
>>> a>>2
0
>>> a<<2
8
```

6. Special Operators
is , is not

```
>>> a=10
>>> b=20
>>> a is b
False
>>> a is not b
True
```

7. Membership Operators
in, not in

```
>>> a='cvsr'
>>> 'r' in a
```

```
True
>>> 'v' not in a
False
>>> b=[1,2,'a']
>>> 1 in b
True
>>> 'a' not in b
False
```

3. Programs on Expressions

```
>>> a=5.0*(10+15/5)
>>> a
65.0
>>> a=2
>>> b=10
>>> c=5
>>> d=a+b/5-a*b+c
>>> d
-11.0
>>> e = a + (b + (10- c) / 2) *1
>>> e
14.5
>>> f = (b-a)/ 5 + a / (9- 1)
>>> f
1.85
```

Week 3:

Programs on Functions

Function Calling

```
def my_function():

  print("Hello from a function")

my_function()
```

Output:

```
Hello from a function
```

Adding two numbers using Function

```
def add(x,y):

    c=x+y

    print(c)

add(10,20)
```

Output:

```
30
```

Function to return values

```
def my_function(x):
  return x
print(my_function(10))
print(my_function(20))
print(my_function(30))
```

Output:

```
10
20
30
```

Fuction using Recursion

```
def factorial(n):
    if(n <= 1):
        return 1
    else:
        return(n*factorial(n-1))
n = int(input("Enter number:"))
print("Factorial:")
print(factorial(n))
```

Output:

```
Enter number:5
Factorial:
120
```

Lambda Fuction

```
double = lambda x: x * 2
print(double(5))
```

Output:

```
10
```

```
x = lambda a, b : a * b
print(x(5, 6))
```

Output:

```
30
```

```
def cube(y):
    return y*y*y;
  g = lambda x: x*x*x
print(g(7))
print(cube(5))
```

Output:

```
343
125
```

```
def myfunc(n):
  return lambda a : a * n
mydoubler = myfunc(2)
mytripler = myfunc(3)
print(mydoubler(11))
print(mytripler(11))
```

Output:

```
22
33
```

```
n=int(input("enter the number"))
x=lambda n:n*2
print(x(n))
```

Output:

```
enter the number 5
10
```

```
def fun(n):
    return lambda a:a*n
myfun=fun(5)
print(myfun(5))
```

Output:

```
25
```

Week 4:
Programs on different Argument types

Passing Parameter to a Function

```
def my_function(fname):
  print(fname + " Text Book")
my_function("Python")
my_function("Java")
my_function("C")
```

Output:

```
Python Text Book
Java Text Book
C Text Book
```

Default Parameter

1.
```
def my_function(country = "India"):
  print("I am from " + country)
my_function("USA")
my_function("Australia")
my_function()
my_function("Paris")
```

Output:
```
I am from USA
I am from Australia
I am from India
I am from Paris
```

2.
```
def greet(name, msg = "Good morning!"):
   print("Hello",name + ', ' + msg)
greet("Pavan")
greet("Raja","How do you do?")
```

Output:
```
Hello Pavan, Good morning!
Hello Raja, How do you do?
```

Passing List as a Parameter
```
def my_function(name):
  for x in name:
    print(x)
fruits = ["AGI", "CVSR", "ANURAG"]
my_function(fruits)
```

Output:

```
AGI
CVSR
ANURAG
```

Function with Keyword Argument

```
def my_function(name, name1, name2):
  print("Name of student " + name)
  print("Name of student1 " + name1)
  print("Name of student2 " + name2)
my_function(name = "Raja", name1 = "Pavan", name2 = "Suraj")
```

Output:

```
Name of student Raja
Name of student1 Pavan
Name of student2 Suraj
```

Arbitrary Arguments

```
def greet(a,*n):
    for x in n:
       print(x)
    print(a)
greet(1,2,3,4)
```

Output:

```
2
3
4
1
```

Week 5:
Programs on Lists and Tuples

1.Lists

Creating List
```
li=[]
print(li)
```

Output:
```
[ ]
```

```
li=[1,'a']
print(li)
```

Output:
```
[1,’a’]
```

```
li = [10,"cvsr",3.5]
print(li)
```

Output:
```
[10, 'cvsr', 3.5]
```

Nested List
```
nlist = ["cvsr", [2,0,1,5]]
print(nlist)
```

Output:
```
['cvsr', [2, 0, 1, 5]]
```

```
nlist = [[1,2,3], [2,0,1,5]]
print(nlist)
```

Output:
```
[[1, 2, 3], [2, 0, 1, 5]]
```

```
nlist = [1,2,3,[2,0,1,5]]
print(nlist)
```

Output:
```
[1, 2, 3, [2, 0, 1, 5]]
```

Accessing List Elements with Index values

```
my_list = ['a','b','c','d','e']
print(my_list[0])
a

print(my_list[1])
b
print(my_list[2])
c

print(my_list[4])
e
```

Negitive Index

```
my_list = ['a','b','c','d','e']
print(my_list[-1])
e

print(my_list[-3])
c

print(my_list[-4])
b
```

Accessing Elements from Nested List with index values

```
nlist = ["cvsr", [2,0,1,5]]
print(nlist[0])
cvsr

print(nlist[1])
[2, 0, 1, 5]

print("Elements are from first list")
print(nlist[0][1])
print(nlist[0][2])
Elements are from first list
v
s
print("Elements are from sub list")
print(nlist[1][1])
print(nlist[1][3])
Elements are from sub list
0
5
```

```
nlist = [[1,2,3], [2,0,1,5]]
print(nlist[0])
[1, 2, 3]

print(nlist[1])
[2, 0, 1, 5]

print("Elements are from first list")
print(nlist[0][1])
print(nlist[0][2])
Elements are from first list
2
3

print("Elements are from sub list")
print(nlist[1][1])
print(nlist[1][3])
Elements are from sub list
0
5

nlist = [1,2,3,[2,0,1,5]]
print(nlist[0])
1
print(nlist[3])
[2, 0, 1, 5]
print("Elements are from first list")
print(nlist[0])
print(nlist[2])
Elements are from first list
1
3
print("Elements are from sub list")
print(nlist[3][1])
print(nlist[3][3])
Elements are from sub list
0
5
```

Negative Index

```
nlist = [1,2,3,[2,0,1,5]]
print(nlist[-1])
[2, 0, 1, 5]
print(nlist[-4])
1
```

```
print(nlist[3][-1])
5
print(nlist[-1][0])
2
```

Slice operations in List

```
values=[10, 20, 30, 40, 50, 60, 70, 80, 90]

print(values[1:])
[20, 30, 40, 50, 60, 70, 80, 90]

print(values[2:])
[30, 40, 50, 60, 70, 80, 90]

print(values[:2])
[10, 20]

print(values[::-1])
[90, 80, 70, 60, 50, 40, 30, 20, 10]

print(values[::1])
[10, 20, 30, 40, 50, 60, 70, 80, 90]

print(values[2::])
[30, 40, 50, 60, 70, 80, 90]

print(values[1::2])
[20, 40, 60, 80]

print(values[: : ])
[10, 20, 30, 40, 50, 60, 70, 80, 90]

print(values[: : 2])
[10, 30, 50, 70, 90]

print(values[1:8:2])
[20, 40, 60, 80]

print(values[-3:2:-1])
[70, 60, 50, 40]

print(values[1:-2:2])
[20, 40, 60]
print(values[-3::-1])
[70, 60, 50, 40, 30, 20, 10]
```

```
print(values[3:6])
[40, 50, 60]
print(values[0:5])
[10, 20, 30, 40, 50]

print(values[:5])

 [10, 20, 30, 40, 50]

print(values[-2:])
[80, 90]

print(values[:-2])
[10, 20, 30, 40, 50, 60, 70]

print(values[::-1])
[90, 80, 70, 60, 50, 40, 30, 20, 10]

print(values[1:-1])
[20, 30, 40, 50, 60, 70, 80]

print(values[-4:7])
[60, 70]

print(values[-4:-1])
 [60, 70, 80]
```

List Methods

```
a=[1,2,3,4]
b=[5,6,8]

a.append(5)
print(a)
[1, 2, 3, 4, 5]

a.extend([6,'c'])
print(a)
[1, 2, 3, 4, 5, 6, 'c']

a[1]=10
print(a)
[1, 10, 3, 4, 5, 6, 'c']


a.insert(0,11)
```

```
print(a)
[11, 1, 10, 3, 4, 5, 6, 'c']

a.remove(4)
print(a)
[11, 1, 10, 3, 5, 6, 'c']

a=[11, 1, 10, 3, 5, 6, 'c']
b= b=[5,6,8]
a.pop()
print(a)
[11, 1, 10, 3, 5, 6]

a=[11, 1, 10, 3,5, 6]
b= b=[5,6,8]

print(a.index(1))
1

print(a.count(a))
0

a.sort()
print(a)
[1, 3, 5, 6, 10, 11]

a.reverse()
print(a)
[11, 10, 6, 5, 3, 1]

print(len(a))
6

for i in range(len(a)):
    print(i)
0
1
2
3
4
5
print(max(a,b))
[11, 10, 6, 5, 3, 1]

print(min(a,b))
[5, 6, 8]
```

```
print(list(range(5)))
[0, 1, 2, 3, 4]

print(list(range(1,5)))
[1, 2, 3, 4]

print(list(range(1,5,2)))
[1, 3]

a.clear()
print(a)
[]

a.delete()
print(a)

Traceback (most recent call last):
  File "D:\py\list methods.py", line 36, in <module>
    a.delete()
AttributeError: 'list' object has no attribute 'delete'

b=['a','b','z','c']
b.sort()
print(b)
['a', 'b', 'c', 'z']

b.sort(key=len)
print(b)
['a', 'b', 'c', 'z']

b.sort(key=len,reverse=True)
print(b)
['a', 'b', 'c', 'z']

b.reverse()
print(b)
['z', 'c', 'b', 'a']

c=b.copy()
print(c)
['z', 'c', 'b', 'a']
```

2. Tuple

```
a = ("hi", "hello", "bye")
print(a)
('hi', 'hello', 'bye')

a = (1, 2.8, "Hello World")
print(a)
(1, 2.8, 'Hello World')

a= ("Book", [1, 2, 3])
print(a)
('Book', [1, 2, 3])

a = ((2, 3, 4), (1, 2, "hi"))
print(a)
((2, 3, 4), (1, 2, 'hi'))

b=()
print(b)
()

a = ("hi", "hello", "bye")
print(a[0])
hi

print(a[2])
bye

print(a[-1])
bye

print(a[-3])
hi
c = (1, "Steve", (11, 22, 33))
print(c[1][3])
v

print(c[2][2])
22

del c
print(c)

Traceback (most recent call last):
  File "D:/py/tuple.py", line 23, in <module>
```

```
    print(c)
NameError: name 'c' is not defined
```

Iterating a Tuple

```
tupp = ("Apple", "Orange", "Grapes", "Banana")
for fruit in tupp:
     print(fruit)
```

Output:

```
Apple
Orange
Grapes
Banana
```

Slice Operations on Tuple

```
d=(5,6,8,9,1,2,10,12,4)

print(d[1:])
(6, 8, 9, 1, 2, 10, 12, 4)

print(d[2:])
(8, 9, 1, 2, 10, 12, 4)

print(d[:2])
(5, 6)
print(d[::-1])
(4, 12, 10, 2, 1, 9, 8, 6, 5)

print(d[::1])
(5, 6, 8, 9, 1, 2, 10, 12, 4)

print(d[2::])
(8, 9, 1, 2, 10, 12, 4)

print(d[1::2])
(6, 9, 2, 12)

print(d[: : ])
(5, 6, 8, 9, 1, 2, 10, 12, 4)

print(d[: : 2])
(5, 8, 1, 10, 4)
```

```
print(d[1:8:2])
(6, 9, 2, 12)

print(d[-3:2:-1])
(10, 2, 1, 9)

print(d[1:-2:2])
(6, 9, 2)

print(d[-3::-1])
(10, 2, 1, 9, 8, 6, 5)

print(d[3:6])
(9, 1, 2)

print(d[0:5])
(5, 6, 8, 9, 1)

print(d[:5])
 (5, 6, 8, 9, 1)

print(d[-2:])
(12, 4)

print(d[:-2])
(5, 6, 8, 9, 1, 2, 10)
print(d[::-1])
(4, 12, 10, 2, 1, 9, 8, 6, 5)
print(d[1:-1])
(6, 8, 9, 1, 2, 10, 12)
print(d[-4:7])
(2, 10)
print(d[-4:-1])
(2, 10, 12)
```

Membership Test in Tuple

```
tup= (11, 22, 33, 44, 55, 66, 77, 88, 99)

print(tup)
(11, 22, 33, 44, 55, 66, 77, 88, 99)

print(22 in tup)
True
print(2 in tup)
False
```

```
print(88 not in tup)
False
print(101 not in tup)
True
```

Tuple Methods

```
tup= (11, 22,11, 33, 44, 55, 66, 77, 88, 99)
tup1=(10,20,30)

print(tup.count(11))
2

print(tup.index(11))
0
print(max(tup))
99

print(min(tup1))
10

print(sum(tup+tup1))
566

print(max(tup,tup1))
(11, 22, 11, 33, 44, 55, 66, 77, 88, 99)

print(min(tup,tup1))
(10, 20, 30)
```

Week 6:
Programs on Dictionaries

Creating Dictionary

```
dict1 = {}
print(dict1)
{}

dic = {1: 'apple', 2: 'ball'}
print(dic)
{1: 'apple', 2: 'ball'}

dic = {'name': 'John', 1: [2, 4, 3]}
print(dic)
{'name': 'John', 1: [2, 4, 3]}

dic = dict({1:'apple', 2:'ball'})
print(dic)
{1: 'apple', 2: 'ball'}

dic = dict([(1,'apple'), (2,'ball')])
print(dic)
{1: 'apple', 2: 'ball'}
```

Accessing Elements of Dectionary

```
dic={'name':'anurag','year':2019}
print(dic)
{'name': 'anurag', 'year': 2019}

print(dic['name'])
anurag

print(dic['year'])
2019
```

Changing Values of Dictionary

```
dic = {'name':'Jack', 'age': 26}
dic['age'] = 27
print(dic)
{'name': 'Jack', 'age': 27}
dic['address'] = 'Downtown'
print(dic)
{'name': 'Jack', 'age': 27, 'address': 'Downtown'}
```

Deleting Elements from Dictionary

```
num = {1:1, 2:4, 3:9, 4:16, 5:25}

del num[5]
print(num)
{1: 1, 2: 4, 3: 9, 4: 16}

print(num.pop(4))
print(num)
16
{1: 1, 2: 4, 3: 9}

print(num.popitem())
print(num)
(3, 9)
{1: 1, 2: 4}

num.clear()
print(num)
{}

del num
print(num)
Traceback (most recent call last):
  File "D:\py\dictionary deletion.py", line 11, in <module>
    print(num)
NameError: name 'num' is not defined
```

Dictionary Methods

```
d = {1: "one", 2: "two"}
d.clear()
print('d =', d)
d = {}

original = {1:'one', 2:'two'}
new = original.copy()
print('Orignal: ', original)
print('New: ', new)
Orignal:  {1: 'one', 2: 'two'}
New:  {1: 'one', 2: 'two'}


keys = {'a', 'e', 'i', 'o', 'u' }
vowels = dict.fromkeys(keys)
```

```
print(vowels)
{'u': None, 'i': None, 'a': None, 'e': None, 'o': None}

person = {'name': 'Phill', 'age': 22}
print('Name: ', person.get('name'))
print('Age: ', person.get('age'))
print('Salary: ', person.get('salary'))
print('Salary: ', person.get('salary', 0.0))
Name:  Phill
Age:  22
Salary:  None
Salary:  0.0

sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }
print(sales.items())
dict_items([('apple', 2), ('orange', 3), ('grapes', 4)])

person = {'name': 'Phill', 'age': 22, 'salary': 3500.0}
print(person.keys())
dict_keys(['name', 'age', 'salary'])

empty_dict = {}
print(empty_dict.keys())
dict_keys([]

person = {'name': 'Phill', 'age': 22}
age = person.setdefault('age')
print('person = ',person)
print('Age = ',age)
person =  {'name': 'Phill', 'age': 22}
Age =  22

sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }
element = sales.pop('apple')
print('The popped element is:', element)
print('The dictionary is:', sales)
The popped element is: 2
The dictionary is: {'orange': 3, 'grapes': 4}

sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }
print(sales.values())
dict_values([2, 3, 4])

d = {1: "one", 2: "three"}
d1 = {2: "two"}
d.update(d1)
```

```
print(d)
{1: 'one', 2: 'two'}

d = {1: "one", 2: "three"}
d1 = {3: "three"}
d.update(d1)
print(d)
{1: 'one', 2: 'three', 3: 'three'}
```

Week 7:
Programs on Strings and String Operations

```
my_string = 'Hello'
print(my_string)
```
Hello

```
my_string = "Hello"
print(my_string)
```
Hello

```
my_string = '''Hello'''
print(my_string)
```
Hello

```
my_string = """Hello, welcome to
          the world of Python"""
print(my_string)
```

Output:
```
Hello, welcome to
          the world of Python
```

Accessing Characters in Strings

```
str = 'programiz'

print('str = ', str)
str =  programiz

print('str[0] = ', str[0])
print('str[5] = ', str[5])
str[0] =  p
str[5] =  a

print('str[-1] = ', str[-1])
print('str[-3] = ', str[-3])
str[-1] =  z
str[-3] =  m
print('str[:] = ', str[:])
print('str[::] = ', str[::])
str[:] =  programiz
str[::] =  programiz

print('str[1:5] = ', str[1:5])
str[1:5] =  rogr
```

```
print('str[5:-2] = ', str[5:-2])
str[5:-2] =  am

print('str[1:8:2] = ', str[1:8:2])
str[1:8:2] =  rgai
```

Concatenation of Two or More Strings

```
str1 = 'Hello'
str2 ='World!'

print('str1 + str2 = ', str1 + str2)

print('str1 * 3 =', str1 * 3)
```

Output:

```
str1 + str2 =  HelloWorld!
str1 * 3 = HelloHelloHello
```

String Membership Test

```
name='helloa'
print('a' in 'name')
print('at' not in 'name')
```

Output:

```
True
True
```

```
print('''He said, "What's there?"''')
print('He said, "What\'s there?"')
print("He said, \"What's there?\"")
```

Output:

```
He said, "What's there?"
He said, "What's there?"
He said, "What's there?"
```

String Opetarions

```
txt = "Hello, and welcome to my world."
x = txt.capitalize()
print (x)
Hello, and welcome to my world.

x = txt.casefold()
print(x)
```

hello, and welcome to my world.

```
txt = "banana"
x = txt.center(20)
print(x)
       banana

txt = "I love apples, apple are my favorite fruit"
x = txt.count("apple")
print(x)
2

x = txt.count("apple", 10, 24)
print(x)
1

txt = "Hello, welcome to my world."
x = txt.endswith(".")
print(x)
True


x = txt.endswith("my world.")
print(x)
True

txt = "I love apples, apple are my favorite fruit"
x = txt.endswith("my world.", 5, 11)
print(x)
False

txt = "H\te\tl\tl\to"
print(txt)
print(txt.expandtabs())
print(txt.expandtabs(2))
print(txt.expandtabs(4))
print(txt.expandtabs(10))

H       e       l       l       o
H       e       l       l       o
H e l l o
H   e   l   l   o
H         e         l         l         o

txt = "Hello, welcome to my world."
x = txt.find("welcome")
```

```
print(x)
x = txt.find("e")
print(x)
x = txt.find("e", 5, 10)
print(x)
```

7
1
8

```
txt = "Hello, welcome to my world."
x = txt.index("welcome")
print(x)
x = txt.index("e")
print(x)
x = txt.index("e", 5, 10)
print(x)
```

7
1
8

```
txt4 = "For only {price:.2f} dollars!"
print(txt4.format(price = 49))
```

For only 49.00 dollars!

```
txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'm {1}".format("John",36)
txt3 = "My name is {}, I'm {}".format("John",36)
print(txt1)
print(txt2)
print(txt3)
```

```
My name is John, I'm 36
My name is John, I'm 36
My name is John, I'm 36
```

```
txt = "Company12"
x = txt.isalnum()
print(x)
txt = "Company 12"
x = txt.isalnum()
print(x)
```

True
False

```
txt = "CompanyX"
x = txt.isalpha()
```

```
print(x)
txt = "Company10"
x = txt.isalpha()
print(x)
```

True
False

```
txt = "50800"
x = txt.isdigit()
print(x)
a = "\u0030"
b = "\u00B2"
print(a.isdigit())
print(b.isdigit())
```
True
True
True
```
txt = "\u0033"
x = txt.isdecimal()
print(x)
a = "\u0030"
b = "\u0047"
print(a.isdecimal())
print(b.isdecimal())
```
True
True
False

```
a = "MyFolder"
b = "Demo002"
c = "2bring"
d = "my demo"
print(a.isidentifier())
print(b.isidentifier())
print(c.isidentifier())
print(d.isidentifier())
```

True
True
False
False

```
a = "Hello world!"
b = "hello 123"
c = "mynameisPeter"
```

```
print(a.islower())
print(b.islower())
print(c.islower())
```
False
True
False

```
txt = "565543"
x = txt.isnumeric()
print(x)
a = "\u0030"
b = "\u00B2"
c = "10km2"
print(a.isnumeric())
print(b.isnumeric())
print(c.isnumeric())
```
True
True
True
False

```
txt = "Hello! Are you #1?"
x = txt.isprintable()
print(x)
txt = "Hello!\nAre you #1?"
x = txt.isprintable()
print(x)
```

True
False

```
a = "HELLO, AND WELCOME TO MY WORLD"
b = "Hello"
c = "22 Names"
d = "This Is %'!?"
print(a.istitle())
print(b.istitle())
print(c.istitle())
print(d.istitle())
```

False
True
True
True

```
a = "Hello World!"
```

```
b = "hello 123"
c = "MY NAME IS PETER"
print(a.isupper())
print(b.isupper())
print(c.isupper())
```

```
False
False
True
```

```
myTuple = ("John", "Peter", "Vicky")
x = "#".join(myTuple)
print(x)
```

```
John#Peter#Vicky
```

```
myDict = {"name": "John", "country": "Norway"}
mySeparator = "TEST"
x = mySeparator.join(myDict)
print(x)
```

```
nameTESTcountry
```

```
txt = "Hello my FRIENDS"
x = txt.lower()
print(x)
```

```
hello my friends
```

```
txt = "I could eat bananas all day"
x = txt.partition("bananas")
print(x)
```

```
('I could eat ', 'bananas', ' all day')
```

```
txt = "I could eat bananas all day"
x = txt.partition("apples")
print(x)
```

```
('I could eat bananas all day', '', '')
```

```
txt = "I like bananas"
x = txt.replace("bananas", "apples")
print(x)
```

```
I like apples
```

```
txt = "one one was a race horse, two two was one too."
x = txt.replace("one", "three")
print(x)
```

```
three three was a race horse, two two was three too.
```

```
txt = "Hello, welcome to my world."
```

```
x = txt.rfind("e")
print(x)
```
13

```
txt = "I could eat bananas all day, bananas are my favorite fruit"
x = txt.rpartition("bananas")
print(x)
```
('I could eat bananas all day, ', 'bananas', ' are my favorite fruit')

```
txt = "Hello, welcome to my world."
x = txt.startswith("Hello")
print(x)
```
Ture

```
x = txt.startswith("wel", 7, 20)
print(x)
```
True

```
txt = "Hello my friends"
x = txt.upper()
print(x)
```
Hello my friends

```
a = "hello"
b = "welcome to the jungle"
c = "10.000"
print(a.zfill(10))
print(b.zfill(10))
print(c.zfill(10))
```
00000hello
welcome to the jungle
000010.000

Week 8:
Programs on Regular Expressions

```
import re
pattern=r"[a-zA-Z]+ \d+"
matches=re.findall(pattern,"LXI 2013,VSI 2015,VDI 20104,maruti suzuki cars in india")
for match in matches:
    print(match,end=" ")
```

Output:

```
LXI 2013 VSI 2015 VDI 20104
```

```
import re
pattern=r"[a-zA-Z]+ \d+"
matches=re.finditer(pattern,"LXI 2013,VSI 2015,VDI 20104,maruti suzuki cars in india")
for match in matches:
    print("match found at starting index:",match.start())
    print("match found at ending index:",match.end())
    print("match found at starting and ending index:",match.span())
```

Output:

```
match found at starting index: 0
match found at ending index: 8
match found at starting and ending index: (0, 8)
match found at starting index: 9
match found at ending index: 17
match found at starting and ending index: (9, 17)
match found at starting index: 18
match found at ending index: 27
match found at starting and ending index: (18, 27)
```

```
import re
string="she sells sea shells on the seashore"
pattern1="she"
if re.match(pattern1,string):
    print("match found")
else:
    print(pattern1,"is not found")
```

Output:

```
match found
```

```
import re
string="she sells sea shells on the seashore"
pattern1="sells"
if re.match(pattern1,string):
    print("match found")
else:
    print(pattern1,"is not found")
pattern2="she"
if re.match(pattern2,string):
    print("match found")
else:
    print("not found")
```

Output:

```
sells is not found
match found
```

```
import re
string="she sells sea shells on the seashore"
pattern1="sells"
if re.search(pattern1,string):
    print("match found")
else:
    print(pattern1,"is not found")
pattern2="she"
if re.search(pattern2,string):
    print("match found")
else:
    print("not found")
```

Output:

```
match found
match found
```

```
import re
pattern=r"2{1,4}$"
if re.match(pattern,"2"):
    print("Match 2")
if re.match(pattern,"222"):
    print("Match 222")
if re.match(pattern,"22222"):
    print("match 22222")
```

Output:

```
Match 2
Match 222
```

```
import re
pattern=r"[aeiou]"
if re.search(pattern,"clue"):
    print("match clue")
else:
    print("match bcdge")
```

Output:

```
match clue
```

```
import re
pattern=r"hi(de)*"
if re.search(pattern,"hidedede"):
    print("Match hidedede")
if re.search(pattern,"hi"):
    print("Match hi")
```

Output:

```
Match hidedede
Match hi
```

```
import re
pattern=r"hi(de)+"
if re.search(pattern,"hidedede"):
    print("Match hidedede")
if re.search(pattern,"hi"):
    print("Match hi")
```

Output:

```
Match hidedede
```

```
import re
pattern=r"2{1,4}$"
if re.search(pattern,"2"):
    print("Match 2")
if re.search(pattern,"222"):
    print("Match 222")
if re.search(pattern,"22222"):
    print("match 22222")
```

Output:

```
Match 2
Match 222
match 22222
```

```
import re
string="she sells sea shells on the seashore"
pattern1="sea"
rep1="ocean"
newstring=re.sub(pattern1,rep1,string,1)
print(newstring)
```

Output:

```
she sells ocean shells on the seashore
```

Week 9:
Programs on class and object, static and instance method implementation

```
class ABC:
   var=10
obj=ABC()
print(obj.var)
```

Output:
```
10
```

```
class ABC:
   def __init__(self,val):
      print("in classmethod")
      self.val=val
      print("the value is:",self.val)
obj=ABC(10)
```

Output:
```
in classmethod
the value is: 10
```

```
class ABC:
   def __init__(self):
      print("in classmethod")
      val=10
      print("the value is:",val)
obj=ABC()
```

Output:
```
in classmethod
the value is: 10
```

```
class ABC:
   class_var=0
   def __init__(self,var):
      ABC.class_var+=1
      self.var=var
      print("the object value is",var)
      print("the value of classvariable is",ABC.class_var)
obj1=ABC(10)
obj2=ABC(20)
```

Output:
```
the object value is 10
the value of classvariable is 1
the object value is 20
the value of classvariable is 2
```

```
class ABC:
    class_var=0
    def __init__(self,var):
        ABC.class_var+=1
        self.var=var
        print("the object value is",var)
        print("the value of classvariable is",ABC.class_var)
    def
__del__(self):
         ABC.class_var-=1
         print("object with value %d is going out of scope"%self.var)
obj1=ABC(10)
obj2=ABC(20)
del obj1
```

Output:

```
the object value is 10
the value of classvariable is 1
the object value is 20
the value of classvariable is 2
object with value 10 is going out of scope
```

```
class rectangle:
    def __init__(self,length,breadth):
        self.length=length
        self.breadth=breadth
    def area(self):
        return self.length*self.breadth
    @classmethod
    def square(cls,side):
        return cls(side,side)
s=rectangle.square(10)
print("area is",s.area())
```

Output:

```
area is 100
```

```
class number:
    even=0
    def check(self,num):
        if(num%2==0):
            self.even=1
    def even_odd(self,num):
        self.check(num)
        if(self.even==1):
            print(num,"is even")
        else:
            print(num,"is odd")
n=number()
n.even_odd(21)
```

Output:

```
 21 is odd
```

```
class number:
    even=[]
    odd=[]
    def __init__(self,num):
        self.num=num
        if(num%2==0):
            number.even.append(num)
        else:
            number.odd.append(num)
n1=number(9)
n2=number(6)
n3=number(8)
n4=number(7)
n5=number(3)
print("even numbers are",number.even)
print("odd numbers are",number.odd)
```

Output:

```
even numbers are [6, 8]
odd numbers are [9, 7, 3]
```

Static method

```
class college:
    @staticmethod
    def name():
        print("*Anurag Group of Institutions*")

college.name()
```

Output:

```
*Anurag Group of Institutions*
```

Static-methods inside a class

```
class college:
    @staticmethod
    def name():
        print("*AGI*")

    def addr():
        print("Hyderabad")

college.name()
college.addr()
```

Output:

```
*AGI*
Hyderabad
```

Calling Static Methods

```
class college:
    @staticmethod
    def name():
        print("*AGI*")

    def addr(self):
        print("Hyderabad")

college.name()

obj = college()
obj.addr()
```

Output:

```
*AGI*
```

Hyderabad

Programs on Instace methods

```
class Arthmetic:
    def __init__(self, a, b):
        self.a = a;
        self.b = b;

    def get_subtraction(self):
        print("sub of %d and %d is: %d" % (self.a, self.b, (self.a-self.b)));

arth = Arthmetic(10, 3);
arth.get_subtraction();
```

Output:

```
sub of 10 and 3 is: 7
```

Week 10:
Programs on Inheritance and Polymorphism

```
class student:
    def name(self):
        print("Pavan")
class location(student):
    def area(self):
        print("Hyderabad")
d = location()
d.name()
d.area()
```

Output:
```
Pavan
Hyderabad
```

Multiple Inheritance

```
class A:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def out(self):
        print("x=",self.x)
        print("Y=",self.y)
class B:
    def __init__(self,z):
        self.z=z
    def out1(self):
        print("Z=",self.z)
class C(A,B):
    def __init__(self):
        A.__init__(self,10,30)
        B.__init__(self,40)
        print("done")
D=C()
#D.success()
D.out()
D.out1()
```

Output:
```
done
x= 10
Y= 30
Z= 40
```

```
class base1:
    def __init__(self):
        super(base1,self).__init__()
        print("Base 1 class")
class base2:
    def __init__(self):
        super(base2,self).__init__()
        print("Base 2 class")
class derived(base1,base2):
    def __init__(self):
        super(derived,self).__init__()
        print("Derived class")
d=derived()
```

Output:

```
Base 2 class
Base 1 class
Derived class
```

Multilevel Inheritance

```
class person:
    def name(self):
        print("name=bhaskar")
class teacher(person):
    def age(self):
        print("age=36")
class exp(teacher):
    def experience(self):
        print("experience=10")
n=exp()
n.name()
n.age()
n.experience()
```

Output:

```
name=bhaskar
age=36
experience=10
```

```
class person:
    def name(self):
        print("Name.....")
class teacher(person):
    def quali(self):
        print("Qualification..... must be Ph.D")
class HOD(teacher):
    def experi(self):
        print("Experience.......atleast 15 years")
hod=HOD()
hod.name()
hod.quali()
hod.experi()
```

Output:

```
Name.....
Qualification..... must be Ph.D
Experience.......atleast 15 years
```

Program to demonstrate method overriding

```
class bird:
    def intro(self):
        print("there are many types of birds")
    def flight(self):
        print("most of the birds can fly but some cannot")
class sparrow(bird):
    def flight(self):
        print("sparrows can fly")
class ostrich(bird):
    def flight(self):
        print("ostriches cannot fly")
obj_bird=bird()
obj_spr=sparrow()
obj_ost=ostrich()
obj_bird.intro()
obj_bird.flight()
obj_spr.intro()
obj_spr.flight()
obj_ost.intro()
obj_ost.flight()
```

Output:

```
there are many types of birds
```

```
most of the birds can fly but some cannot
there are many types of birds
sparrows can fly
there are many types of birds
ostriches cannot fly
```

```
class Animal:
  # properties
          multicellular = True
          # Eukaryotic means Cells with Nucleus
          eukaryotic = True

          # function breath
          def breathe(self):
              print("I breathe oxygen.")

  # function feed
          def feed(self):
              print("I eat food.")

# child class
class Herbivorous(Animal):

    # function feed
          def feed(self):
              print("I eat only plants. I am vegetarian.")

herbi = Herbivorous()
herbi.feed()
# calling some other function
herbi.breathe()
```

Output:

```
I eat only plants. I am vegetarian.
I breathe oxygen.
```

Program to perform addition of two complex numbers using binary + operator overloading

```
class complex:
    def __init__(self,a,b):
        self.a=a
        self.b=b
    def __add__(self,other):
         return self.a+other.a,self.b+other.b
    def __str__(self):
        return self.a,self.b
ob1=complex(1,2)
ob2=complex(2,3)
ob3=ob1+ob2
print(ob3)
```

Output:

```
(3, 5)
```

```
class Human:
    def sayHello(self, name=None):
            if name is not None:
                print('Hello ' + name)
            else:
                print('Hello ')

# Create instance
obj = Human()
# Call the method
obj.sayHello()
# Call the method with a parameter
obj.sayHello('Guido')
```

Output:

```
Hello
Hello Guido
```

Week 11:
Programs on Abstract classes and Interfaces

Abstract Class

```
class AbstractClass:

    def do_something(self):
        pass

class B(AbstractClass):
    pass

a = AbstractClass()
b = B()



from abc import ABC, abstractmethod

class AbstractClassExample(ABC):

    @abstractmethod
    def do_something(self):
        print("Some implementation!")

class AnotherSubclass(AbstractClassExample):

    def do_something(self):
        super().do_something()
        print("The enrichment from AnotherSubclass")

x = AnotherSubclass()
x.do_something()
```

Output:

```
Some implementation!
The enrichment from AnotherSubclass
```

```
from abc import ABC, abstractmethod

class Polygon(ABC):

    # abstract method
    def noofsides(self):
        pass

class Triangle(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 3 sides")

class Pentagon(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 5 sides")

class Hexagon(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 6 sides")

class Quadrilateral(Polygon):

    # overriding abstract method
    def noofsides(self):
        print("I have 4 sides")

# Driver code
R = Triangle()
R.noofsides()

K = Quadrilateral()
K.noofsides()

R = Pentagon()
R.noofsides()

K = Hexagon()
K.noofsides()
```

Output:

```
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides
class fruit:
    def taste(self):
        raiseNotImplementedError()
    def rich_in(self):
        raiseNotImplementedError()
    def colour(self):
        raiseNotImplementedError()
class mango:
    def taste(self):
        return "Sweet"
    def rich_in(self):
        return "Vitamin A"
    def colour(self):
        return "Yellow"
class orange:
    def taste(self):
        return "Sour"
    def rich_in(self):
        return "Vitamin C"
    def colour(self):
        return "Orange"
alphanso=mango()
print(alphanso.taste(),alphanso.rich_in(),alphanso.colour())
org=orange()
print(org.taste(),org.rich_in(),org.colour())
```

Output:

```
Sweet Vitamin A Yellow
Sour Vitamin C Orange
```

Week 12:
Programs on Exception Handling

```
a = [1, 2, 3]
try:
        print ("Second element = %d" %(a[1]))
        print ("Fourth element = %d" %(a[3]))
except IndexError:
        print ("An error occurred")
```

Output:

```
Second element = 2
An error occurred
```

```
a = int(input("Enter a:"))
b = int(input("Enter b:"))
c = a/b;
print("a/b = %d"%c)
```

Output:

```
Enter a:10
Enter b:0
Traceback (most recent call last):
File "D:/py/exception.py", line 3, in <module>
c = a/b;
ZeroDivisionError: division by zero
```

```
try:
        a = int(input("Enter a:"))
        b = int(input("Enter b:"))
        c = a/b;
print("a/b = %d"%c)
except Exception:
        print("can't divide by zero")
else:
        print("Hi I am else block")
```

Output:

```
Enter a:6
Enter b:3
a/b = 2
Hi I am else block
```

```
Enter a:10
Enter b:0
can't divide by zero
```

```
def enterage(age):
   if age < 0:
      raise ValueError("Only positive integers are allowed")
   if age % 2 == 0:
      print("age is even")
   else:
      print("age is odd")
try:
   num = int(input("Enter your age: "))
   enterage(num)
except ValueError:
   print("Only positive integers are allowed")
except:
   print("something is wrong")
```

Output:

```
Enter your age: 10
age is even

Enter your age: -1
Only positive integers are allowed
```

```
try:
         file=open("foo.txt")
         str=file.readline()
         print(str)
except IOError:
         print("Error occured during input... Program terminated")
else:
         print("Program terminating successfully")
```

Output:

```
Python is a great language.
Program terminating successfully
```

```
try:
    num=int(input("Enter a number:"))
    if(num>=0):
        print(num)
    else:
        raise ValueError("Negative number not allowed")
except ValueError as e:
    print(e)
```

Output:

```
Enter a number:-33
Negative number not allowed
```

```
class invalidAge(Exception):
    def display(self):
        print("Sorry!!! Age cannot be below 18....")
class invalidName(Exception):
    def display(self):
        print("Please Enter a valid name....")
try:
    name=input("Enter a name")
    if len(name)==0:
        raise invalidName
    age=int(input("Enter the age= "))
    if age<18:
        raise invalidAge
except invalidName as n:
    n.display()
except invalidAge as a:
    a.display()
else:
    print(name,"you can vote")
```

Output:

```
Enter a namejaswanth
Enter the age= 19
jaswanth you can vote
```

Week 13:
Demonstration of Numpy package

```
import numpy as np
a=np.array([1,2,3])
print(a)
```

Output :

```
 [1 2 3]
```

```
a=np.array([(1,2,3),(4,5,6)])
print(a)
```

Output:

```
 [[ 1 2 3]
[4 5 6]]
```

```
from numpy import *
arr=array([1.3,2,3,4,5],'int')
print(arr)
arr1=ones((3,3))
print(arr1)
a=linspace(0,15,3)
print(a)
b=diag([3.1,4])
print(b)
c=logspace(1,20,5)
print(c)
print(arr1+a)
```

Output:

```
[1 2 3 4 5]
[[ 1.  1.  1.]
[ 1.  1.  1.]
[ 1.  1.  1.]]
[  0.    7.5  15. ]
[[ 3.1  0. ]
[ 0.   4. ]]
[  1.00000000e+01   5.62341325e+05   3.16227766e+10   1.77827941e+15
1.00000000e+20]
[[  1.    8.5  16. ]
[  1.    8.5  16. ]
[  1.    8.5  16. ]]
```

```
import numpy as np
a= np.array([1,3,5,7,9])
b = np.array([3,5,6,7,9])
c = a + b
print(c)
```

Output:

```
[4 8 11 14 18]
```

Week 14:
Demonstration of Pandas package

```
import pandas as pd
lst = ['Geeks', 'For', 'Geeks', 'is', 'portal', 'for', 'Geeks']
df = pd.DataFrame(lst)
print(df)
```

Output:

```
        0
0   Geeks
1     For
2   Geeks
3      is
4  portal
5     for
6   Geeks
```

```
import pandas as pd
data = {'Name':['Tom', 'nick', 'krish', 'jack'],'Age':[20, 21, 19, 18]}
df = pd.DataFrame(data)
print(df)
```

Output:

```
    Name  Age
0    Tom   20
1   nick   21
2  krish   19
3   jack   18
```

```
import pandas as pd
studentData = {
'name' : ['jack', 'Riti', 'Aadi'],
'age' : [34, 30, 16],
'city' : ['Sydney', 'Delhi', 'New york']
}
dfObj = pd.DataFrame(studentData)
print(dfObj)
```

Output :

```
   name  age      city
0  jack   34    Sydney
1  Riti   30     Delhi
2  Aadi   16  New york
```

Week 15 :
Demonstration of Matplotlib package

MATPLOTLIB

Installation :

Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package :

```
python -mpip install -U matplotlib
```

Importing matplotlib :

```
from matplotlib import pyplot as plt
```

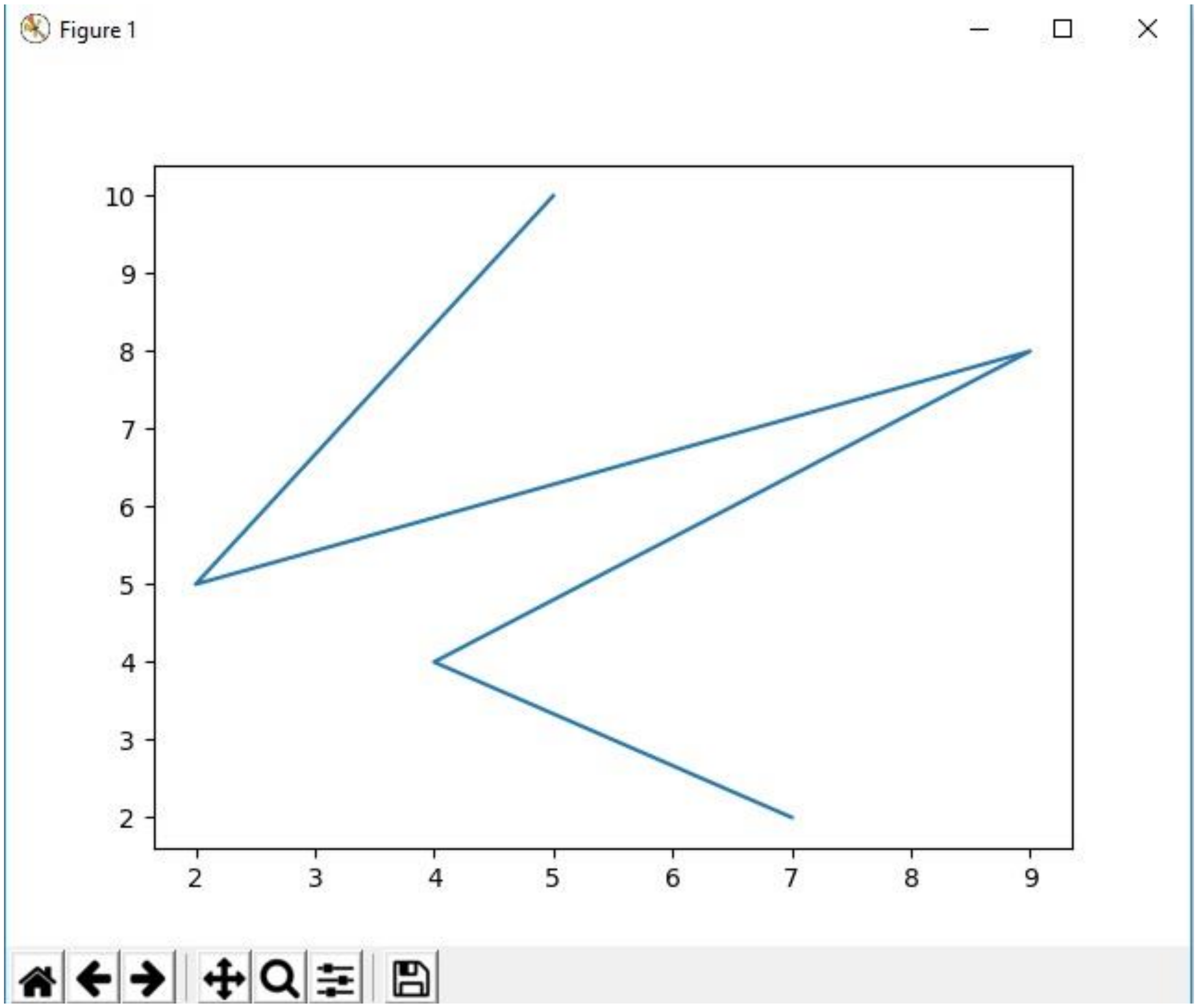
or

```
import matplotlib.pyplot as plt
```

Line Graph Program

```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot
plt.plot(x,y)

# function to show the plot
plt.show()
```

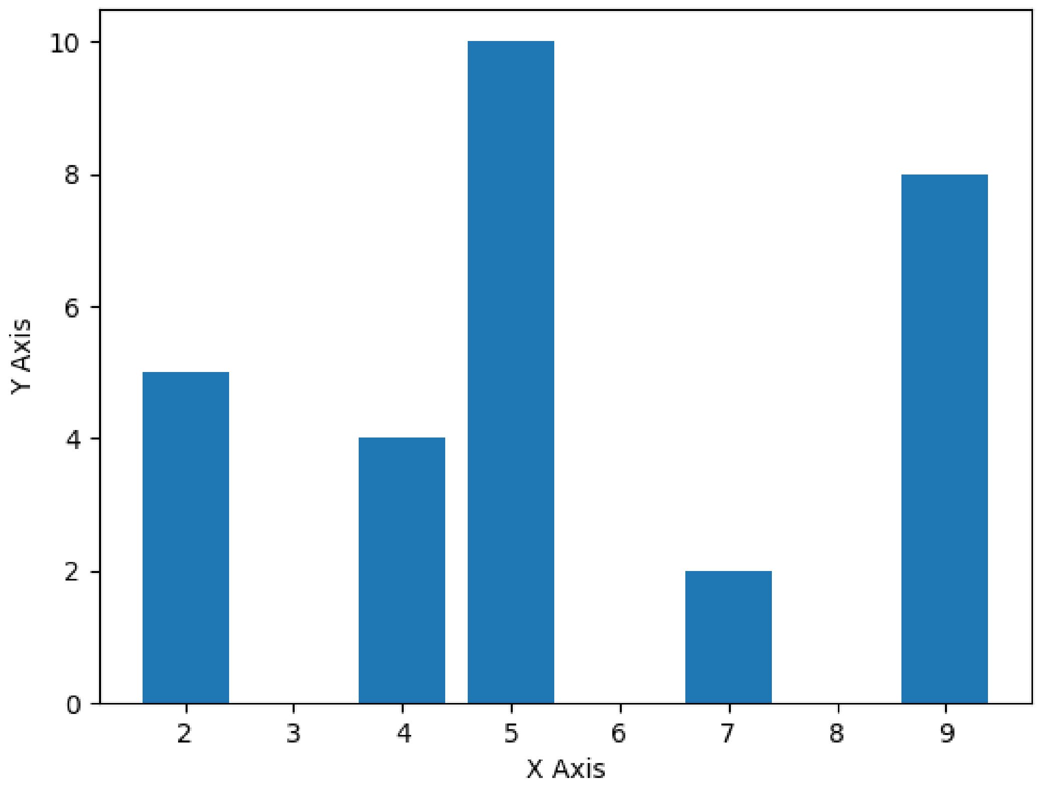
Output:



Bar Graph:

```
# importing matplotlib module
from matplotlib import pyplot as plt
# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]

plt.xlabel("X Axis")
plt.ylabel("Y Axis")
# Function to plot the bar
plt.bar(x,y)

# function to show the plot
plt.show()
```

Output:

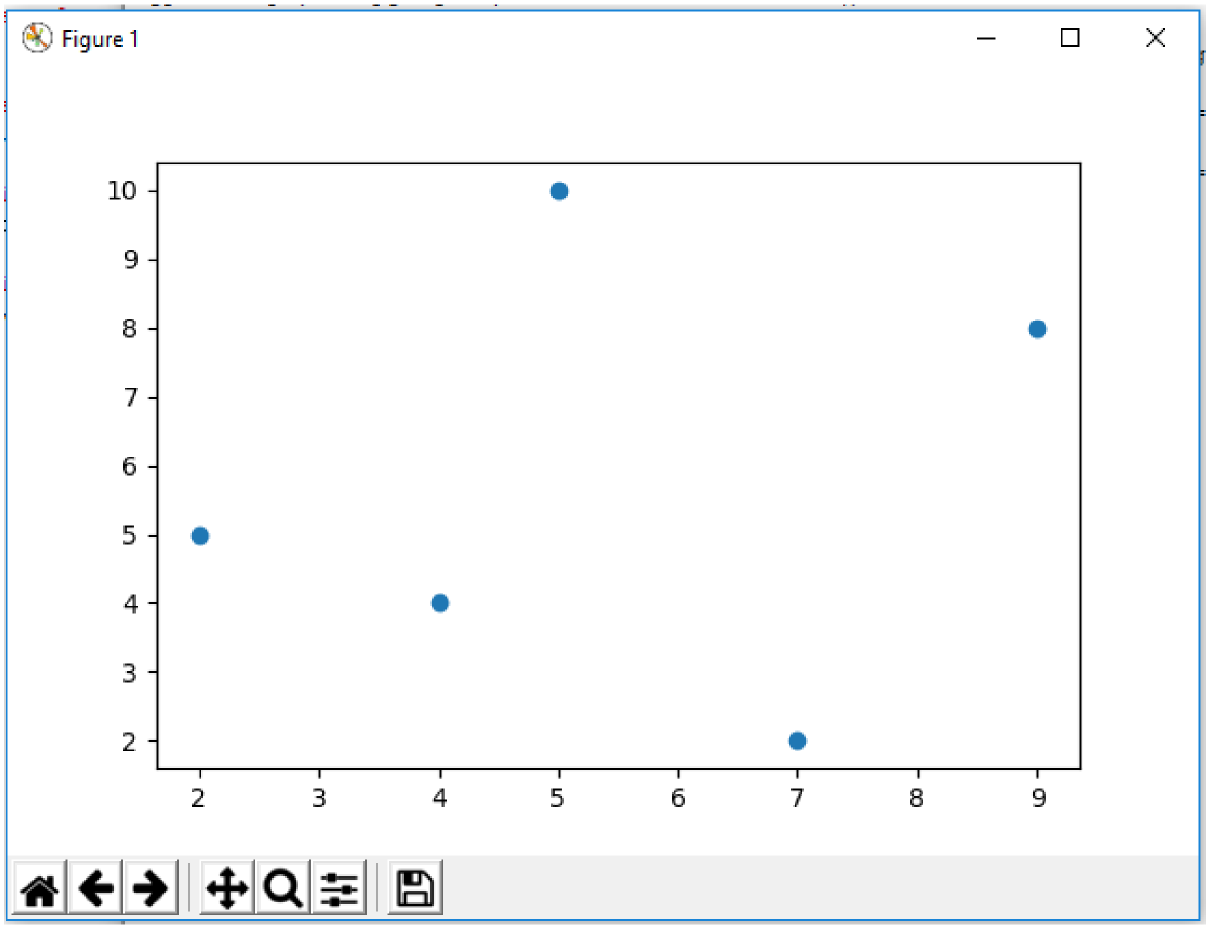


Scatter Graph:

```
# importing matplotlib module
from matplotlib import pyplot as plt

# x-axis values
x = [5, 2, 9, 4, 7]

# Y-axis values
y = [10, 5, 8, 4, 2]
```

```
# Function to plot scatter
plt.scatter(x, y)

# function to show the plot
plt.show()
```

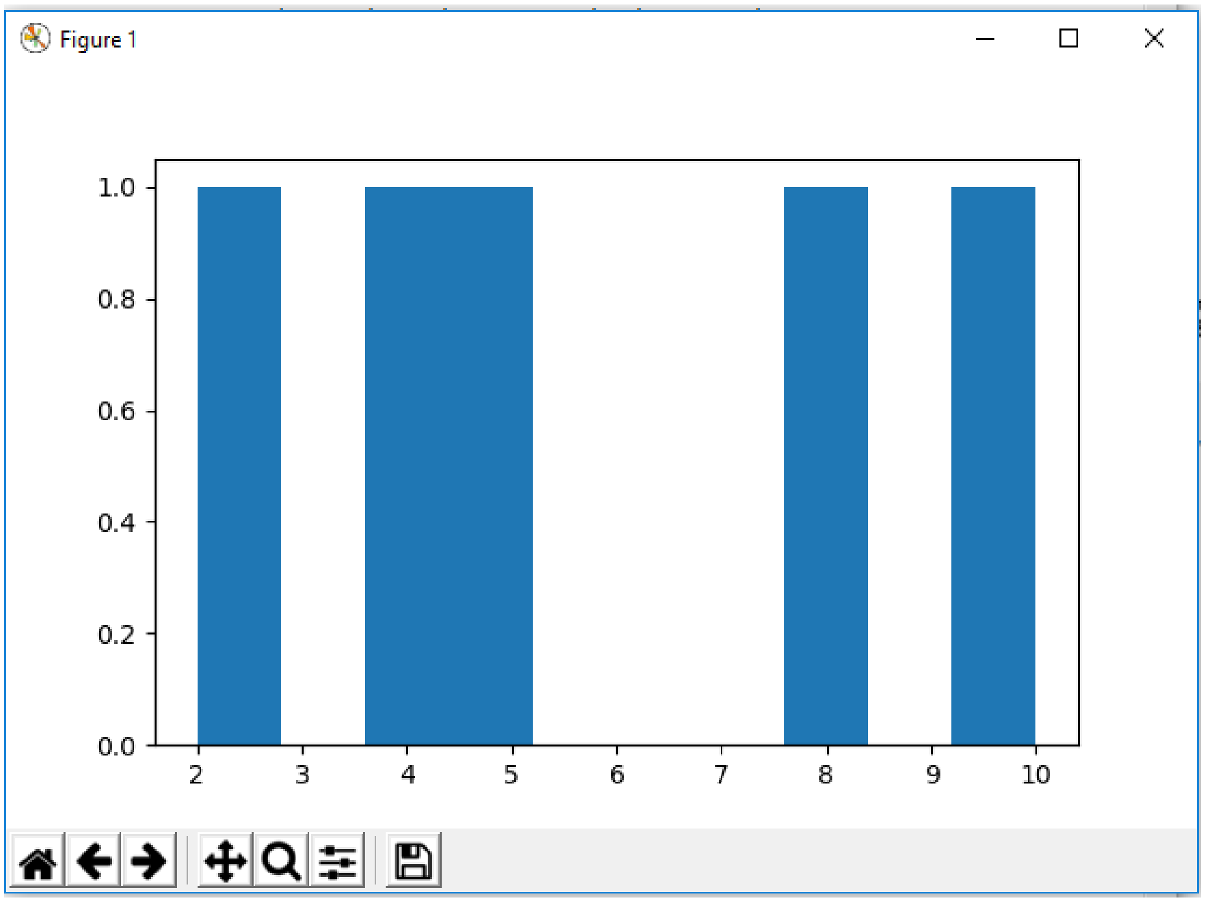
Output:



Histogram:

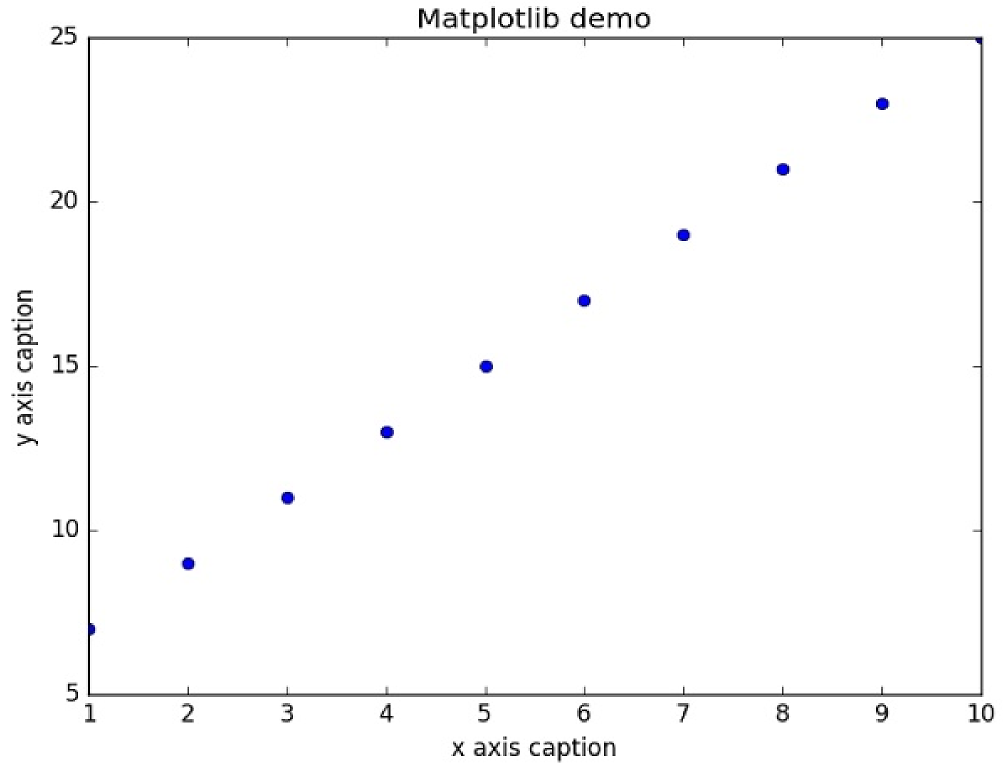
```
# importing matplotlib module
from matplotlib import pyplot as plt

# Y-axis values
y = [10, 5, 8, 4, 2]

# Function to plot histogram
plt.hist(y)

# Function to show the plot
plt.show()
```

Output:



import numpy as np from matplotlib
import pyplot as plt
x = np.arange(1,11)
y = 2 * x + 5
plt.title("Matplotlib demo")
plt.xlabel("x axis caption")
plt.ylabel("y axis caption")
plt.plot(x,y,"ob") plt.show()

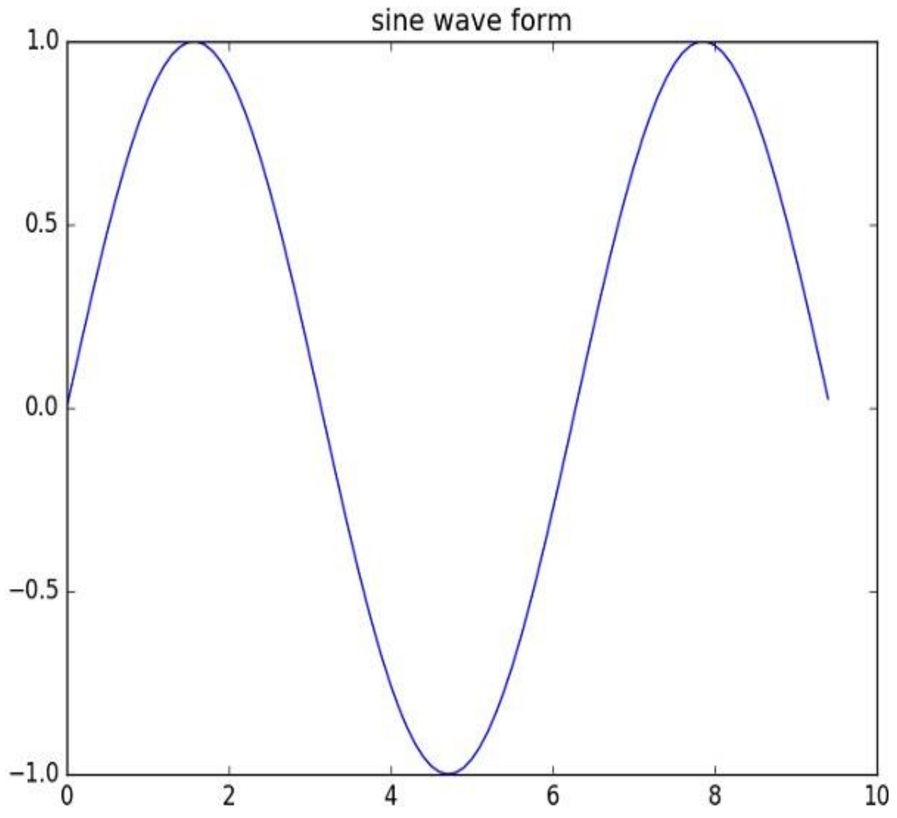
Output:



Sine Wave Plot

```
import numpy as np
import matplotlib.pyplot as plt
# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)
plt.title("sine wave form")
# Plot the points using matplotlib
plt.plot(x, y)
plt.show()
```

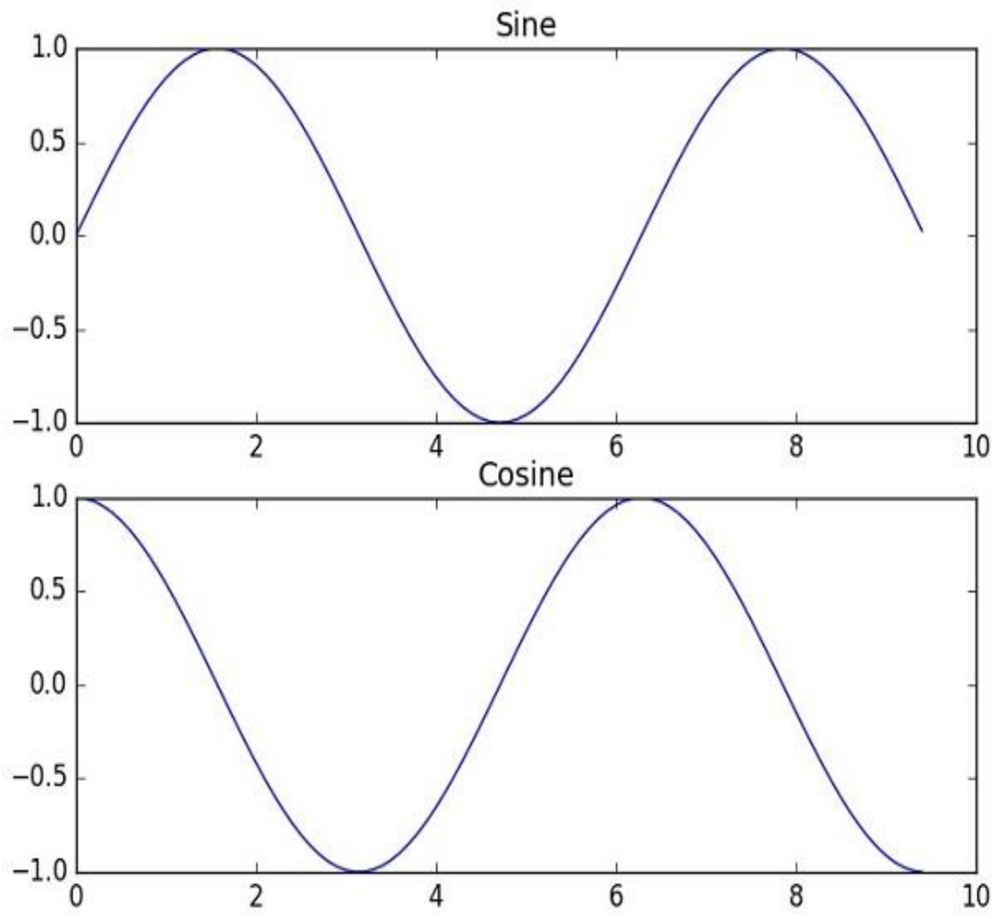
Output:



Subplot

```
import numpy as np import matplotlib.pyplot as plt
# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
# Set up a subplot grid that has height 2 and width 1, # and set the first such subplot as active.
plt.subplot(2, 1, 1)
# Make the first plot plt.
plot(x, y_sin) plt.title('Sine')
# Set the second subplot as active, and make the second plot.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')
plt.show()
```

Output:



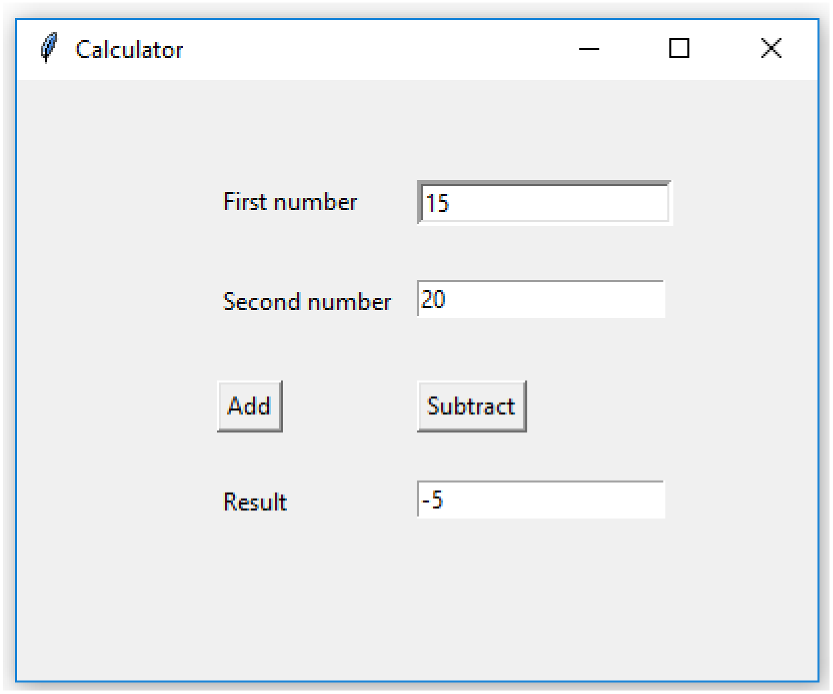
Barplot:

Week 16:
Demonstration of Tkinter package

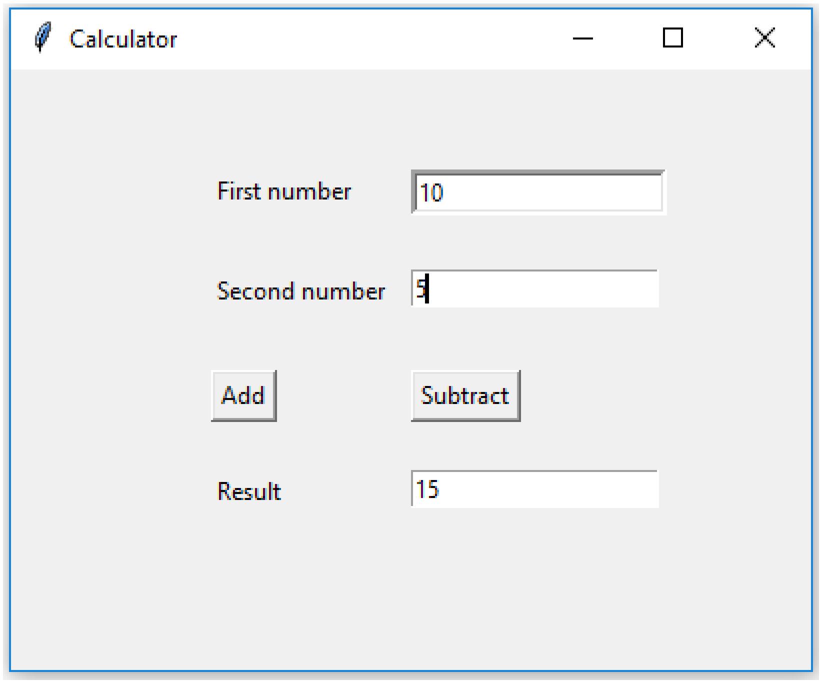
```
from tkinter import *
class MyWindow:
def __init__(self, win):
self.lbl1=Label(win, text='First number')
self.lbl2=Label(win, text='Second number')
self.lbl3=Label(win, text='Result')
self.t1=Entry(bd=3)
self.t2=Entry()
self.t3=Entry()
self.btn1 = Button(win, text='Add')
self.btn2=Button(win, text='Subtract')
self.lbl1.place(x=100, y=50)
self.t1.place(x=200, y=50)
self.lbl2.place(x=100, y=100)
self.t2.place(x=200, y=100)
self.b1=Button(win, text='Add', command=self.add)
self.b2=Button(win, text='Subtract')
self.b2.bind('<Button-1>', self.sub)
self.b1.place(x=100, y=150)
self.b2.place(x=200, y=150)
self.lbl3.place(x=100, y=200)
self.t3.place(x=200, y=200)
def add(self):
self.t3.delete(0, 'end')
num1=int(self.t1.get())
num2=int(self.t2.get())
result=num1+num2
self.t3.insert(END, str(result))
def sub(self, event):
self.t3.delete(0, 'end')
num1=int(self.t1.get())
num2=int(self.t2.get())
result=num1-num2
self.t3.insert(END, str(result))
window=Tk()
mywin=MyWindow(window)
window.title('Calculator')
window.geometry("400x300+10+10")
window.mainloop()
```

Output:

Subtraction Output:



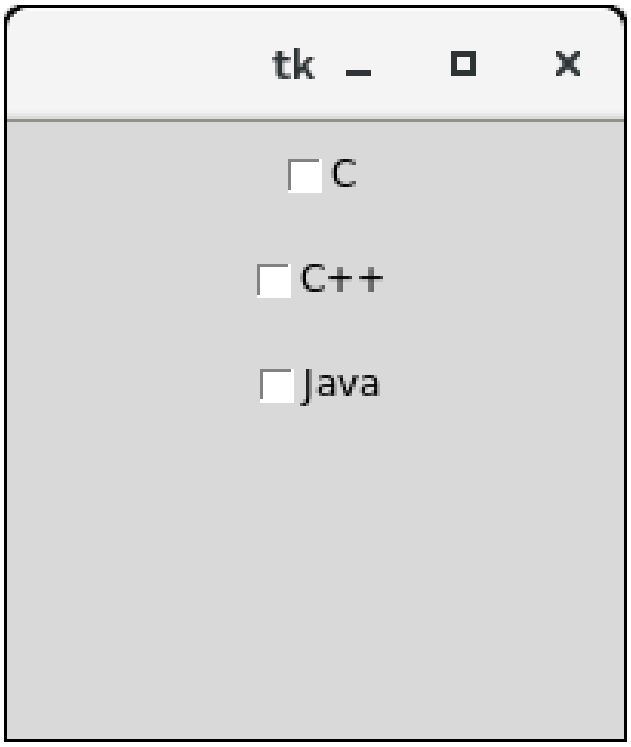
Addition Output:



```
from tkinter import *
top = Tk()
top.geometry("200x200")
checkvar1 = IntVar()
checkvar2 = IntVar()
checkvar3 = IntVar()
chkbtn1 = Checkbutton(top, text = "C", variable = checkvar1,
onvalue = 1, offvalue = 0, height = 2, width = 10)
chkbtn2 = Checkbutton(top, text = "C++", variable = checkvar2,
 onvalue = 1, offvalue = 0, height = 2, width = 10)
chkbtn3 = Checkbutton(top, text = "Java", variable = checkvar3,
onvalue = 1, offvalue = 0, height = 2, width = 10)
chkbtn1.pack()
chkbtn2.pack()
chkbtn3.pack()
top.mainloop()
```

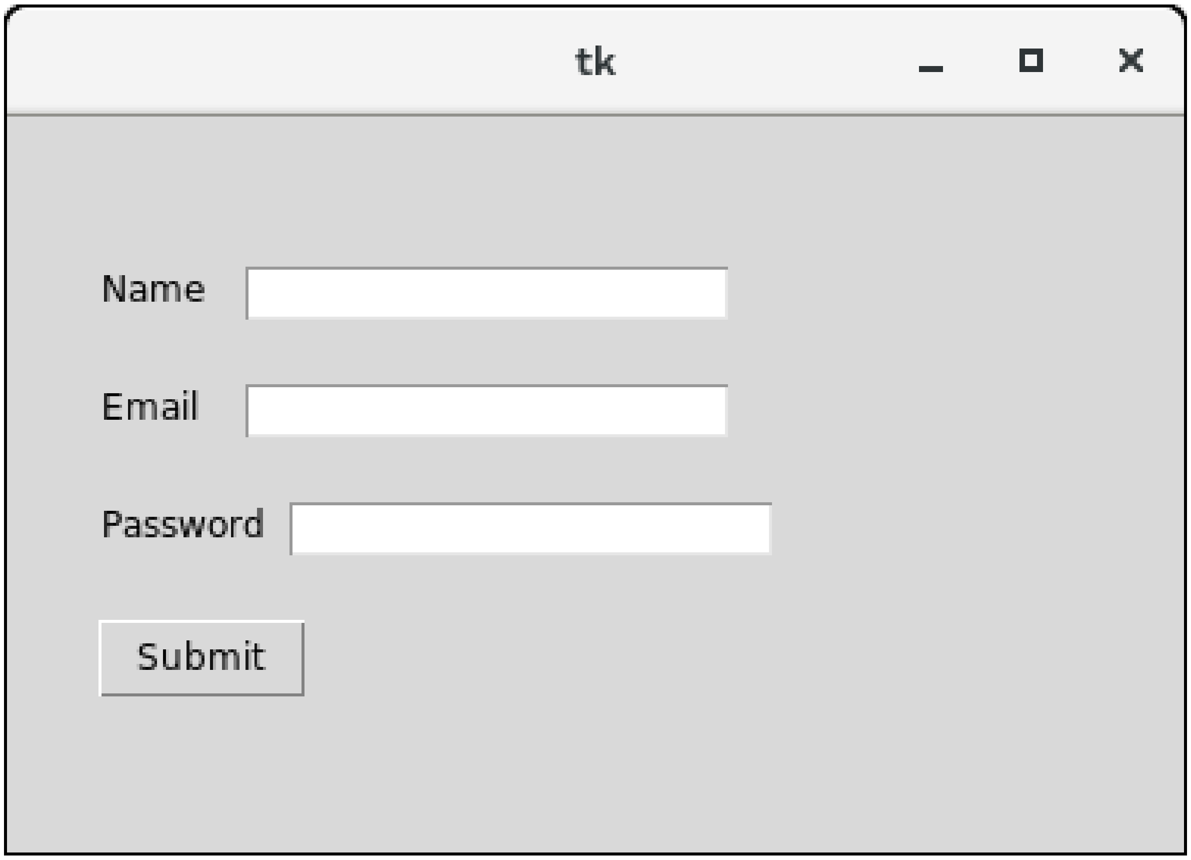
Output:

Output:



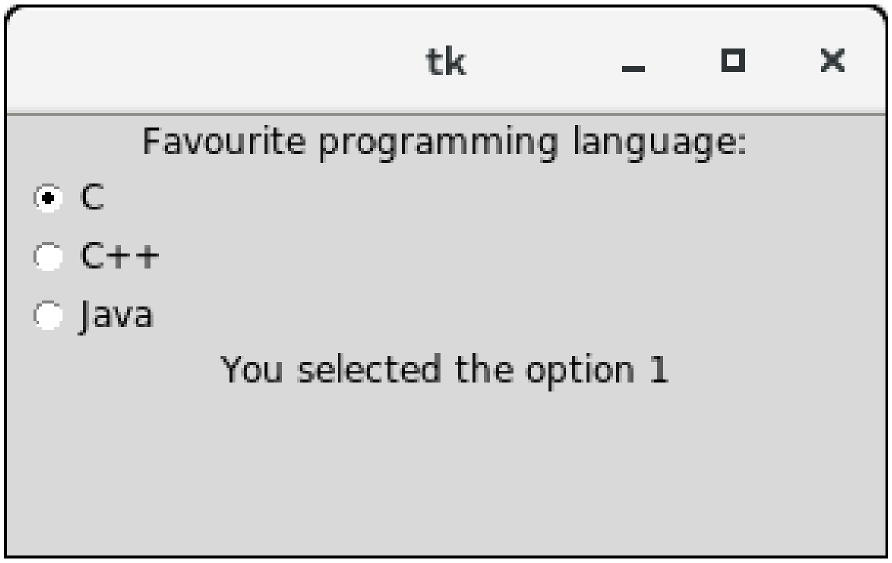
```
from tkinter import *
top = Tk()
top.geometry("400x250")
name = Label(top, text = "Name").place(x = 30,y = 50)
email = Label(top, text = "Email").place(x = 30, y = 90)
password = Label(top, text = "Password").place(x = 30, y = 130)
sbmitbtn = Button(top, text = "Submit",
activebackground = "  pink", activeforeground = "blue").place(x = 30, y = 170)
e1 = Entry(top).place(x = 80, y = 50)
e2 = Entry(top).place(x = 80, y = 90)
e3 = Entry(top).place(x = 95, y = 130)
top.mainloop()
```

Output:



```
from tkinter import *
def selection():
selection = "You selected the option " + str(radio.get())
label.config(text = selection)
top = Tk()
top.geometry("300x150")
radio = IntVar()
lbl = Label(text = "Favourite programming language:")
lbl.pack()
R1 = Radiobutton(top, text="C", variable=radio, value=1,   command=selection)
R1.pack( anchor = W )
R2 = Radiobutton(top, text="C++", variable=radio, value=2,  command=selection)
R2.pack( anchor = W )
R3 = Radiobutton(top, text="Java", variable=radio, value=3,  command=selection)
R3.pack( anchor = W)
label = Label(top)
label.pack()
top.mainloop()
```

Output:



Favourite programming language:

- [✓ C]
- C++
- Java

You selected the option 1