



PYTHON PROGRAMMING

Python Programming

INTRODUCTION TO PYTHON

Python Programming

What is Python?

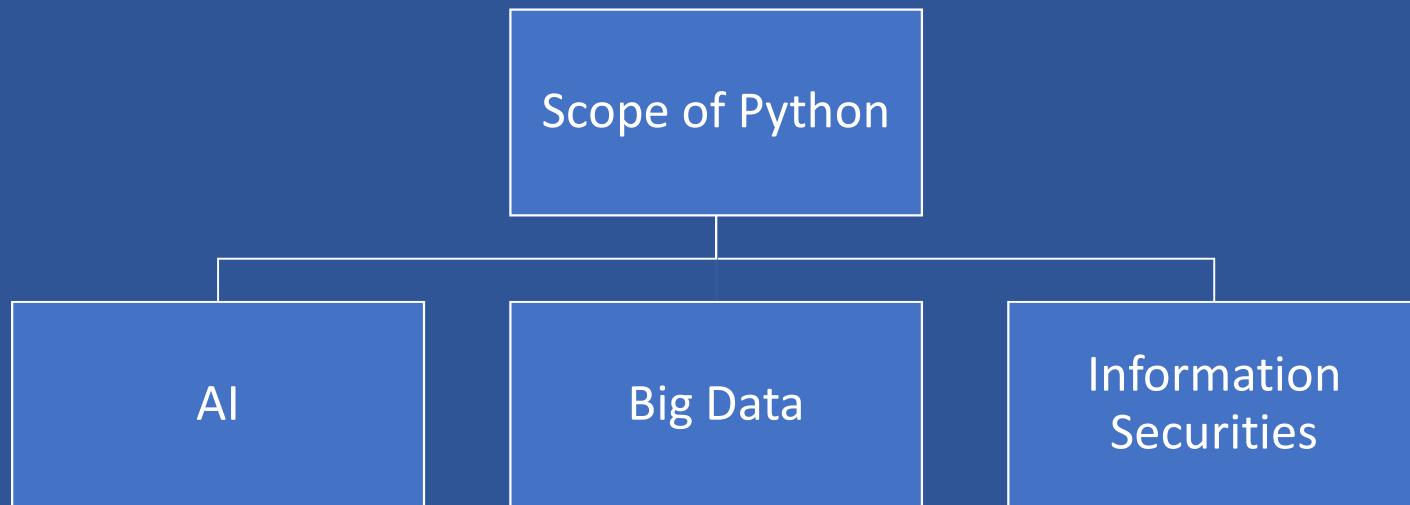
- Python is an interpreted high Level Programing Language
- Python Was Created By Guido van Rossum and its first release was in 1991
- You can develop desktop GUI Applications, Websites and Web Applications using Python which makes it a General Purpose Language.
- Python is Worlds Fastest growing language because of easiness and readability

Python Programming

Why Python?

- Python is more productive than other programming languages
- Companies can optimize their most expensive resource: employees
- Rich set of libraries and frameworks
- Large community

Python Programming



Python Programming

PYTHON SETUP

Python Programming

Software Requirements

Download Python

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

Download Notepad++

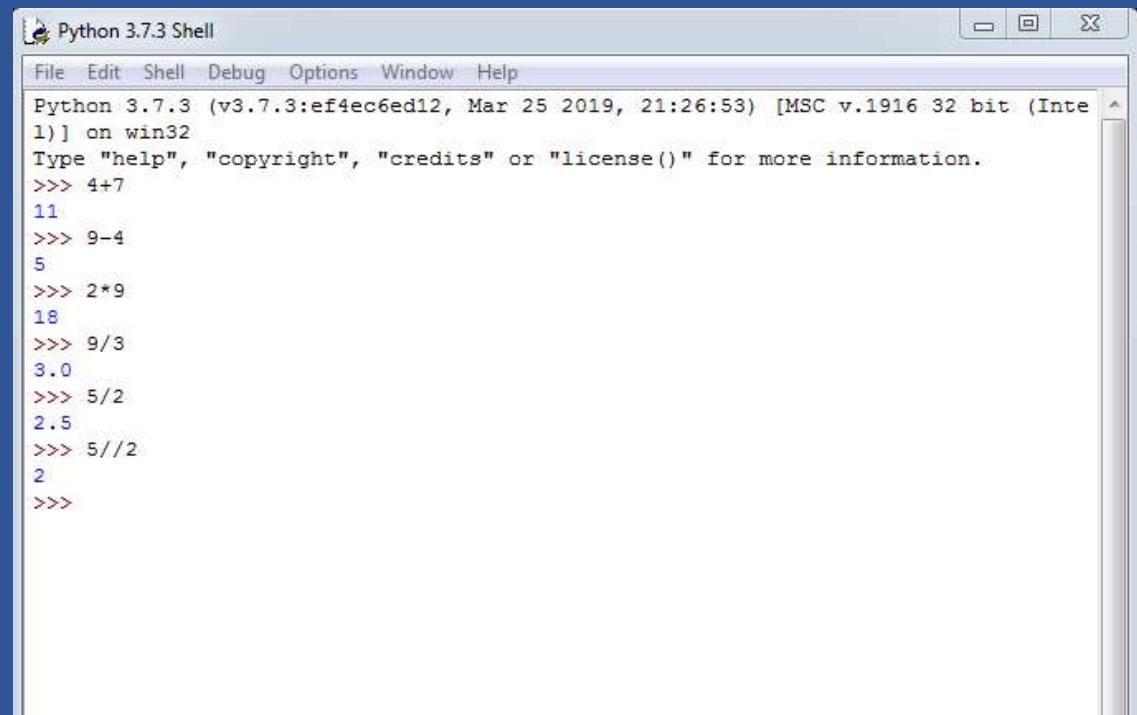
<https://notepad-plus-plus.org/download/v7.6.6.html>

Python Programming

Getting Started With Python

Python Programming

Simple Basics Operations

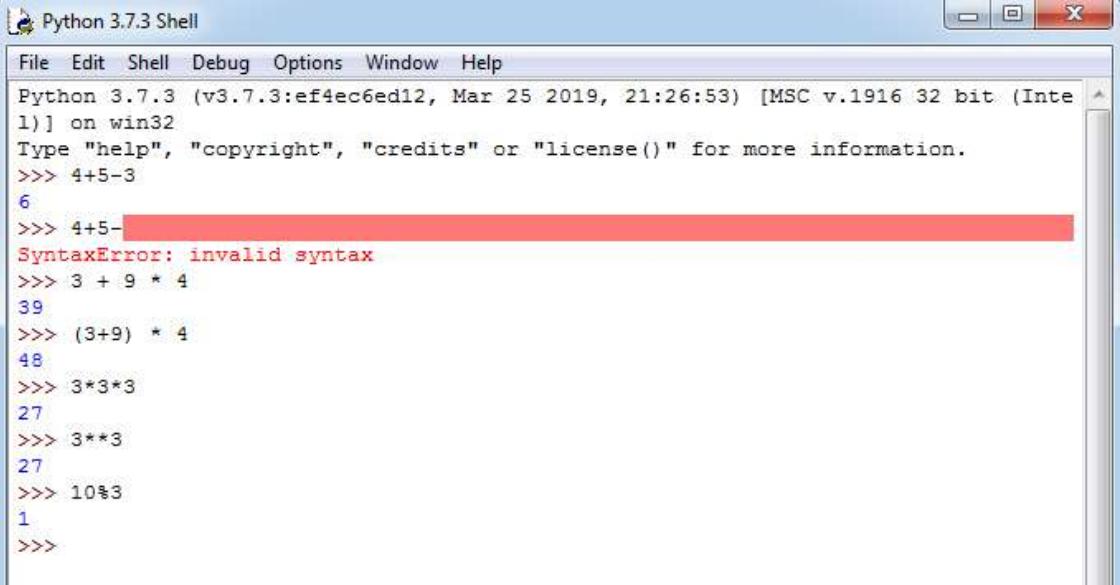


A screenshot of the Python 3.7.3 Shell window. The title bar says "Python 3.7.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following Python session:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1) ] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 4+7
11
>>> 9-4
5
>>> 2*9
18
>>> 9/3
3.0
>>> 5/2
2.5
>>> 5//2
2
>>>
```

Python Programming

Simple Basics Operations

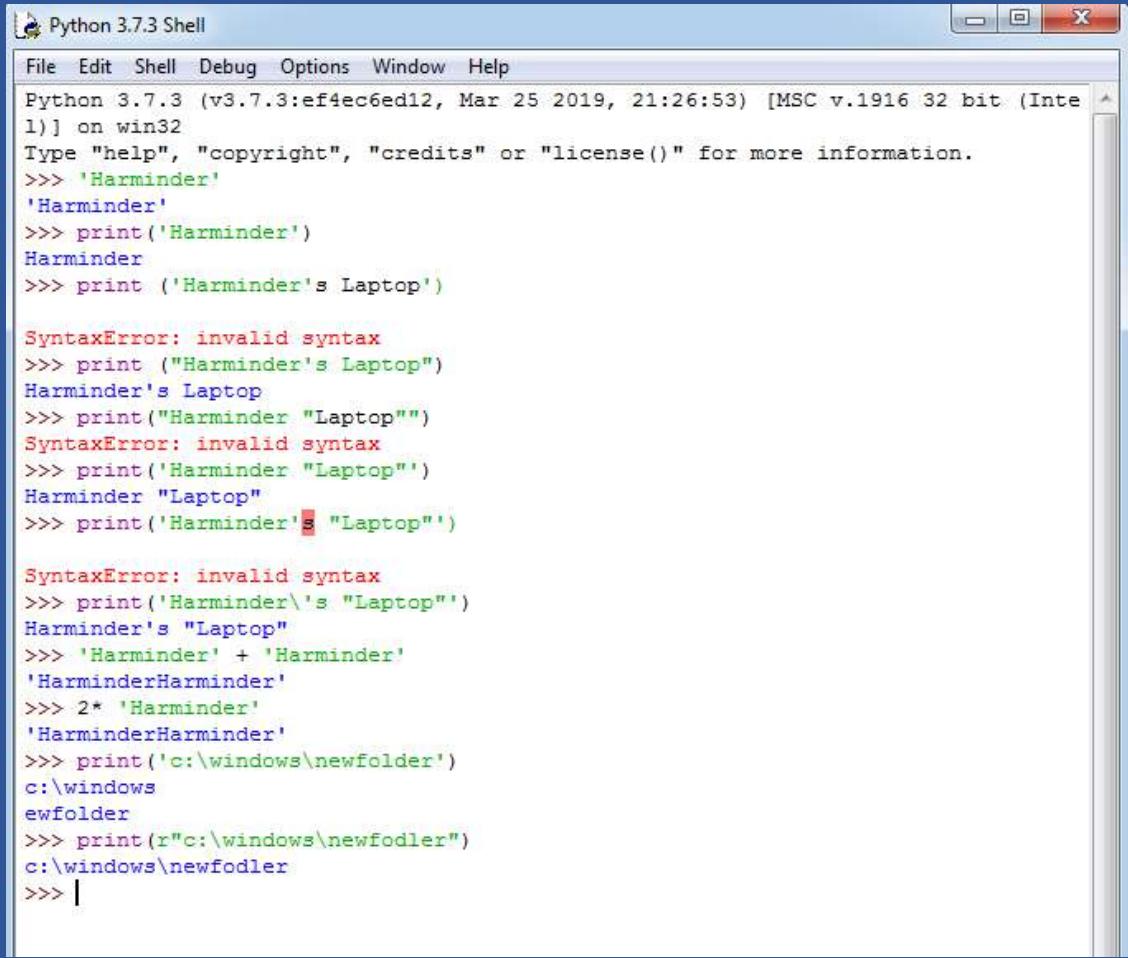


The screenshot shows a Windows desktop with a Python 3.7.3 Shell window open. The window title is "Python 3.7.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays Python code and its output:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 4+5-3
6
>>> 4+5-
SyntaxError: invalid syntax
>>> 3 + 9 * 4
39
>>> (3+9) * 4
48
>>> 3*3*3
27
>>> 3**3
27
>>> 10%3
1
>>>
```

Python Programming

Simple Basics Operations



The screenshot shows a window titled "Python 3.7.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area displays Python code and its execution results. The code consists of several print statements and demonstrates common syntax errors.

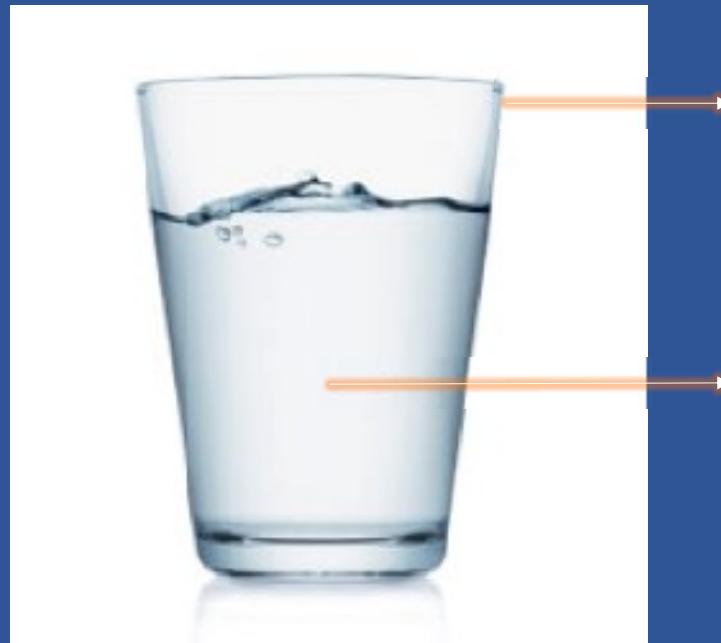
```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'Harminder'
'Harminder'
>>> print('Harminder')
Harminder
>>> print ('Harminder's Laptop')

SyntaxError: invalid syntax
>>> print ("Harminder's Laptop")
Harminder's Laptop
>>> print("Harminder "Laptop")
SyntaxError: invalid syntax
>>> print('Harminder "Laptop"')
Harminder "Laptop"
>>> print('Harminder's "Laptop"')

SyntaxError: invalid syntax
>>> print('Harminder\'s "Laptop"')
Harminder's "Laptop"
>>> 'Harminder' + 'Harminder'
'HarminderHarminder'
>>> 2* 'Harminder'
'HarminderHarminder'
>>> print('c:\windows\newfolder')
c:\windows
ewfolder
>>> print(r"c:\windows\newfodler")
c:\windows\newfodler
>>> |
```

Python Programming

Variables

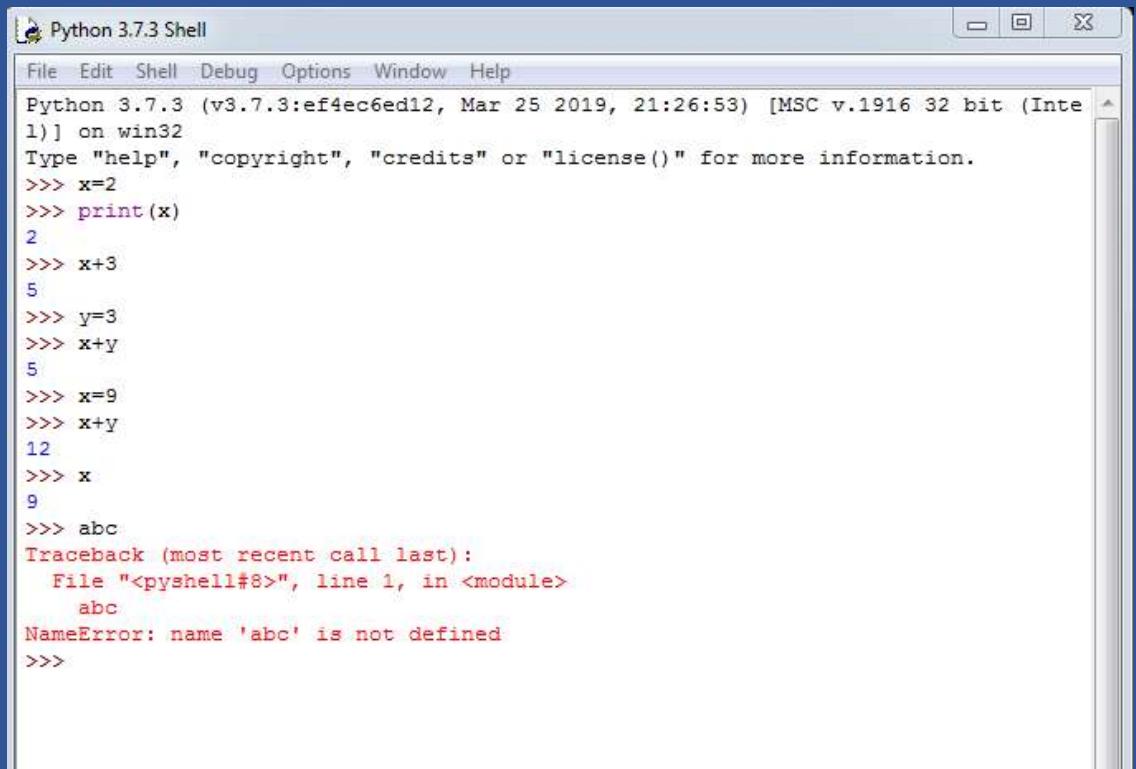


Variable Name
(Glass)

Value
(Water)

Python Programming

Variables

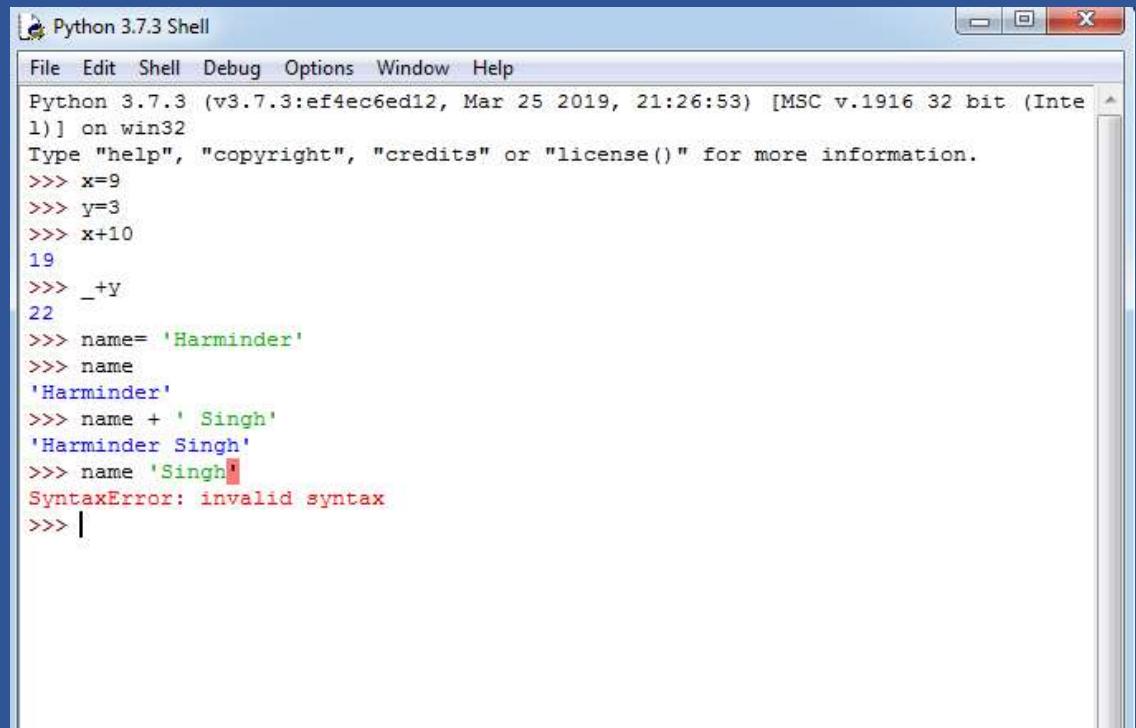


The screenshot shows a Python 3.7.3 Shell window. The code entered is:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> x=2
>>> print(x)
2
>>> x+3
5
>>> y=3
>>> x+y
5
>>> x=9
>>> x+y
12
>>> x
9
>>> abc
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    abc
NameError: name 'abc' is not defined
>>>
```

Python Programming

Variables



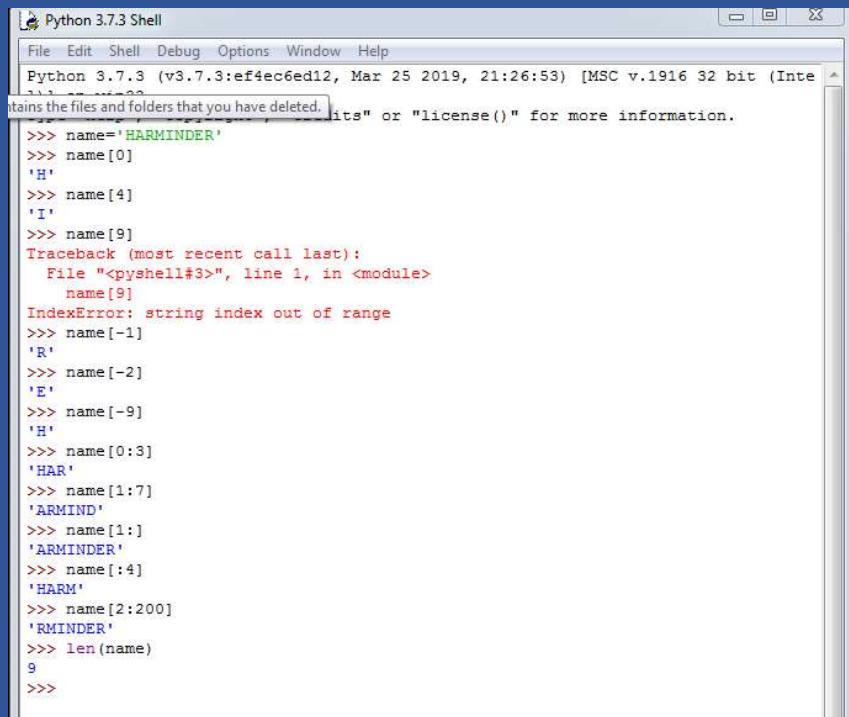
The screenshot shows a Windows-style window titled "Python 3.7.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area displays Python code and its output:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> x=9
>>> y=3
>>> x+10
19
>>> _+y
22
>>> name= 'Harminder'
>>> name
'Harminder'
>>> name + ' Singh'
'Harminder Singh'
>>> name 'Singh'
SyntaxError: invalid syntax
>>> |
```

Python Programming

Variables

-9 -8 -7 -6 -5 -4 -3 -2 -1
HARMINDER
0 1 2 3 4 5 6 7 8



A screenshot of the Python 3.7.3 Shell window. The title bar says "Python 3.7.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, Help. The version information at the top says "Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte...". A message about deleted files is visible. The code in the shell shows various string slicing operations on the name 'HARMINDER'.

```
>>> name='HARMINDER'
>>> name[0]
'H'
>>> name[4]
'I'
>>> name[9]
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    name[9]
IndexError: string index out of range
>>> name[-1]
'R'
>>> name[-2]
'E'
>>> name[-9]
'H'
>>> name[0:3]
'HAR'
>>> name[1:7]
'ARMIND'
>>> name[1:]
'ARMINDER'
>>> name[:4]
'HARM'
>>> name[2:200]
'RMINDER'
>>> len(name)
9
>>>
```

Python Programming

LISTS



Python Programming

LISTS

Defining Lists

```
nums = [23,34,46,67,89]
```

List Name

Elements

Python Programming

LISTS

Accessing Elements

-5 -4 -3 -2 -1
nums = [23,34,46,67,89]
 0 1 2 3 4

```
>>nums[1]
34
>>nums[4]
89
>>nums[2:]
[46,67,89]
>>nums[-2]
67
```

Python Programming

LISTS

Accessing Elements

```
names = ['Vipul','Surender','Anup','Shubham']
```

```
>>names  
['Vipul','Surender','Anup','Shubham']  
>>names[3]  
Shubham  
>>names[2:]  
['Anup','Shubham']  
>>names[-2]  
Anup
```

Python Programming

Lists can have heterogeneous values

LISTS

values = [8.2,'Surender',34]

Python Programming

LISTS

Multi Dimensional Lists

```
names = ['Vipul','Surender','Anup','Shubham']
value = [1,2,3,4]
mi = [names,value]
```

```
>>mi
[['Vipul','Surender','Anup','Shubham'] , [1,2,3,4]]
>>mi[0][3]
Shubham
>>mi[1]
[1,2,3,4]
```

Python Programming

Lists are Mutable (Appending a Element)

LISTS

```
>> nums = [1,2,3,4,5]  
>>nums.append(34)  
>>nums  
[1,2,3,4,5,34]
```

Python Programming

Lists are Mutable (Inserting a Element)

LISTS

```
>> nums = [1,2,3,4,5]
>>nums.insert(3,45)
>>nums
[1,2,3,45,4,5]
```

Python Programming

Lists are Mutable (Removing a Element)

LISTS

```
>> nums = [1,2,3,4,5]  
>>nums.remove(5)  
>>nums  
[1,2,3,4]
```

Python Programming

Lists are Mutable

(Removing a Element using index)

LISTS

```
>> nums = [1,2,3,4,5]  
>>nums.pop(2)  
>>nums  
[1,2,4,5]
```

Python Programming

Lists are Mutable

(Removing a Element from Last)

LISTS

```
>> nums = [1,2,3,4,5]
>>nums.pop()
5
>>nums
[1,2,3,4]
```

Python Programming

Lists are Mutable

(Removing multiple Elements)

LISTS

```
>> nums = [1,2,3,4,5]  
>>del nums[0:2]  
>>nums  
[3,4,5]
```

Python Programming

Lists are Mutable (Adding multiple Elements)

LISTS

```
>> nums = [1,2,3]
>>nums.extend([4,5,6])
>>nums
[1,2,3,4,5,6]
```

Python Programming

Lists are Mutable

(Searching Min Value in a List)

LISTS

```
>> nums = [23,19,85,13]  
>>min(nums)  
13
```

Python Programming

Lists are Mutable

(Searching Max Value in a List)

LISTS

```
>> nums = [23,19,85,13]  
>>max(nums)  
85
```

Python Programming

Lists are Mutable

(Calculate Sum of a List)

LISTS

```
>> nums = [23,19,85,13]  
>>sum(nums)  
140
```

Python Programming

Lists are Mutable

(Sorting a List)

LISTS

```
>> nums = [23,19,85,13]  
>>nums.sort()  
>>nums  
[13,19,23,85]
```

Python Programming

LISTS

Lists are Mutable
(Sorting a List Descending)

```
>> nums = [23,19,85,13]
>>nums.sort(reverse=true)
>>nums
[85,23,19,13]
```

Python Programming

TUPLE



Python Programming

TUPLE

Defining a Tuple

```
nums = (23,34,46,67,89)
```

↓
Tuple Name

↓
Elements

Python Programming

TUPLE

Accessing Elements

-5 -4 -3 -2 -1
nums = (23,34,46,67,89)
0 1 2 3 4

```
>>nums[1]  
34  
>>nums[4]  
89  
>>nums[2:]  
[46,67,89]  
>>nums[-2]  
67
```

Python Programming

TUPLE

Accessing Elements

```
names = ('Vipul','Surender','Anup','Shubham')
```

```
>>names  
('Vipul','Surender','Anup','Shubham')  
>>names[3]  
Shubham  
>>names[2:]  
('Anup','Shubham')  
>>names[-2]  
Anup
```

Python Programming

Tuple can have heterogeneous values

TUPLE

```
values = (8.2,'Surender',34)
```

Python Programming

TUPLE

Multi Dimensional TUPLE

```
names = ('Vipul','Surender','Anup','Shubham')  
value = (1,2,3,4)  
mi = (names,value)
```

```
>>mi  
(('Vipul','Surender','Anup','Shubham') , (1,2,3,4))  
>>mi[0][3]  
Shubham  
>>mi[1]  
(1,2,3,4)
```

Python Programming

TUPLE

Tuples are Immutable

```
>> tup = (1,2,3,4,5)  
>>tup[1] = 36
```

```
>>> tup[1] = 36  
Traceback (most recent call last):  
  File "<pyshell#12>", line 1, in <module>  
    tup[1] = 36  
TypeError: 'tuple' object does not support item assignment  
>>>
```

Python Programming

SETS

Python Programming

SETS

Defining a SET

nums = {23,34,46,67,89}

Set Name

Elements

Python Programming

SETS can have heterogeneous values

SETS

values = {8.2,'Surender',34}

Python Programming

SETS

Check if element exists in SET

```
values = {8.2,'Surender',34}  
print("surender" in values)
```

Python Programming

SETS

Adding Element to SETS

```
values = {8.2,'Surender',34}  
Values.add("hello")
```

Python Programming

SETS

Adding Multiple Element to SETS

```
values = {8.2,'Surender',34}  
values.update([3,4,5])
```

Python Programming

SETS

Removing Element From SETS (Gives an Error when Item is not in Set)

```
values = {8.2,'Surender',34}  
values.remove('Surender')
```

Python Programming

SETS

Removing Element From SETS
(No Error when Item is not in Set)

```
values = {8.2,'Surender',34}  
values.discard('Surender')
```

Python Programming

Removing Random Element From SETS

SETS

```
values = {8.2,'Surender',34}  
values.pop()
```

Python Programming

SETS

Clearing SET

```
values = {8.2,'Surender',34}  
values.clear()
```

Python Programming

SETTING PATH FOR WINDOWS

Python Programming

Setting Path for Windows

Checking If already Set



A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The window shows the following text:
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\DELL>python
'python' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\DELL>

Python Programming

Setting Path for Windows

Copy Following Paths

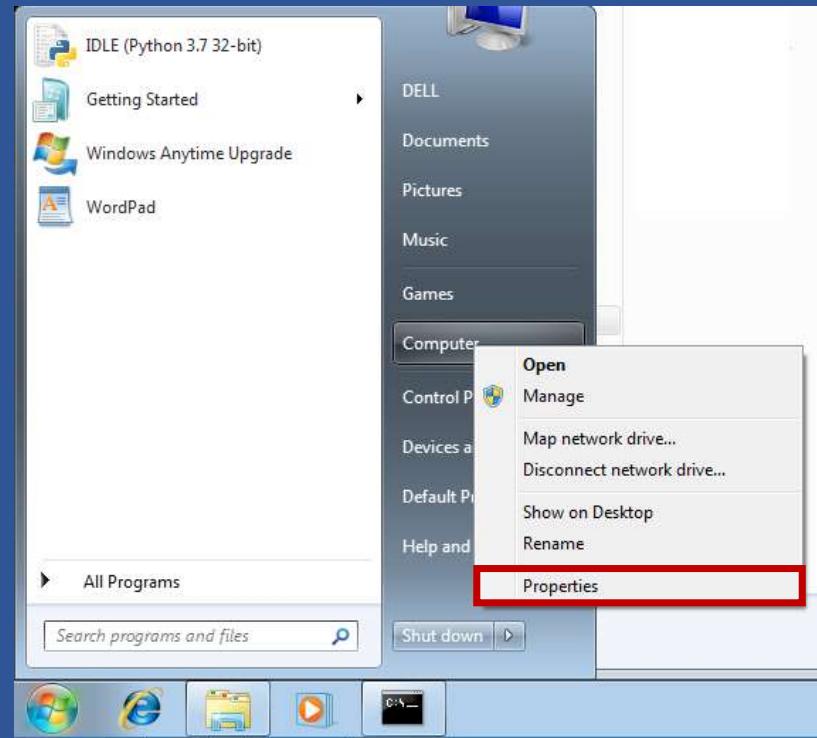
C:\Users\Your_Username\AppData\Local\Programs\Python\Python37-32

C:\Users\ Your_Username
\AppData\Local\Programs\Python\Pyt
hon37-32\Scripts

Python Programming

Setting Path for Windows

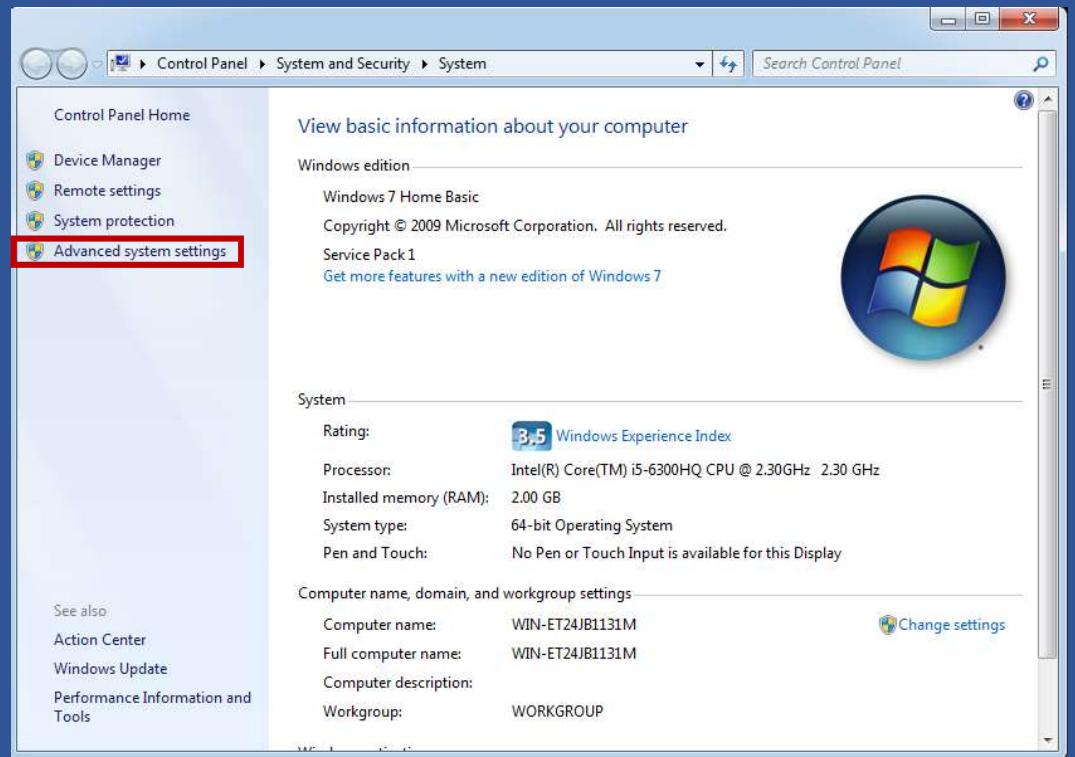
Go to Following Path



Python Programming

Setting Path for Windows

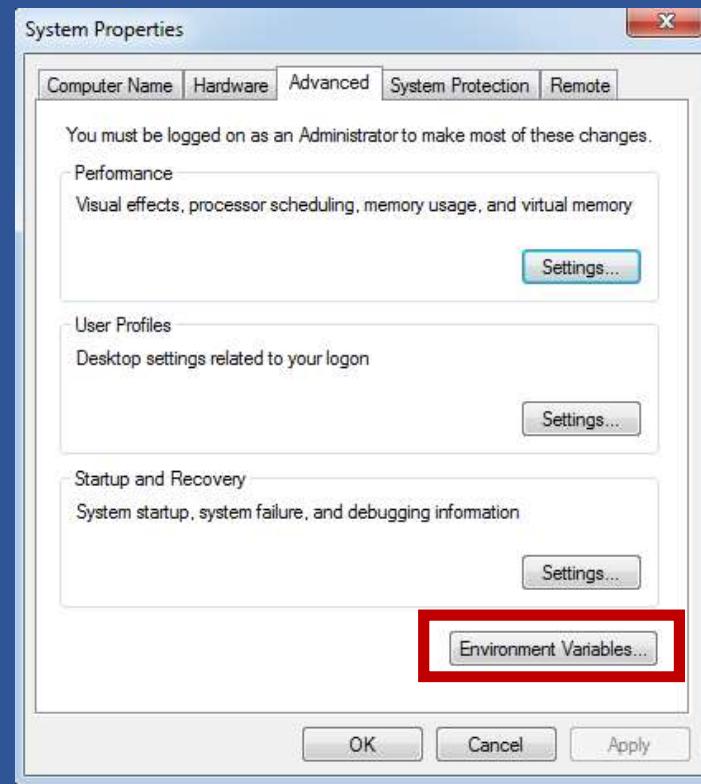
Go to Following Path



Python Programming

Setting Path for Windows

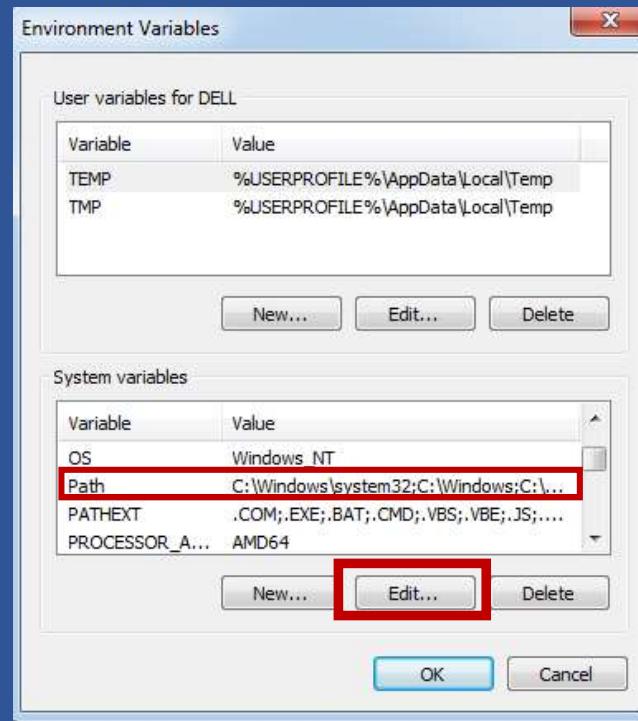
Go to Following Path



Python Programming

Setting Path for Windows

Go to Following Path



Python Programming

Setting Path for Windows

Go to Following Path

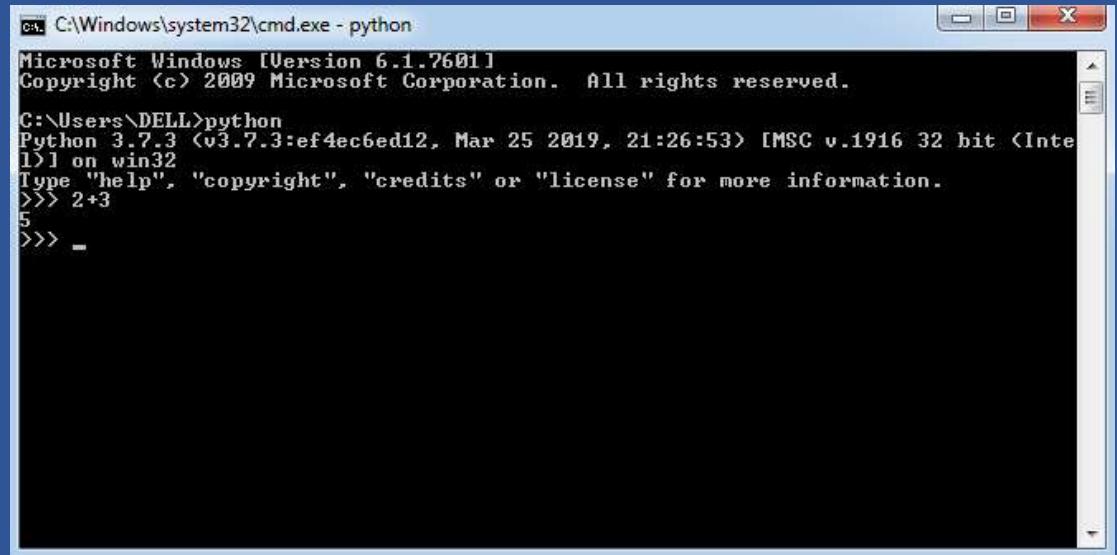


1. Copy both the paths after this semicolon
2. Separate both paths with semicolon

Python Programming

Setting Path for Windows

Verification



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - python". The window shows the following text:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)]
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> -
```

Python Programming

Variable Memory Concept

Python Programming

Variable (Memory Concept)

Variable Storage

num=5



<Memory Address>

Python Programming

Variable (Memory Concept)

Getting Address

```
>>num=5  
>>id(num)  
1936155808
```

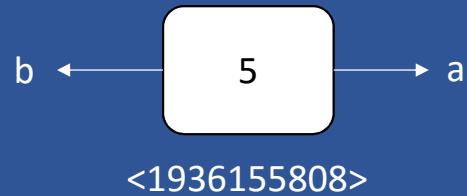


Python Programming

Variable (Memory Concept)

Variables with Same value has same
memory Address

```
>>a=5  
>>b=5  
>>id(a)  
1936155808  
>>id(b)  
1936155808
```

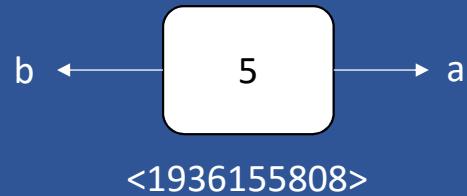


Python Programming

Variable (Memory Concept)

Variables with Same value has same
memory Address

```
>>a=5  
>>b=5  
>>id(a)  
1936155808  
>>id(b)  
1936155808
```

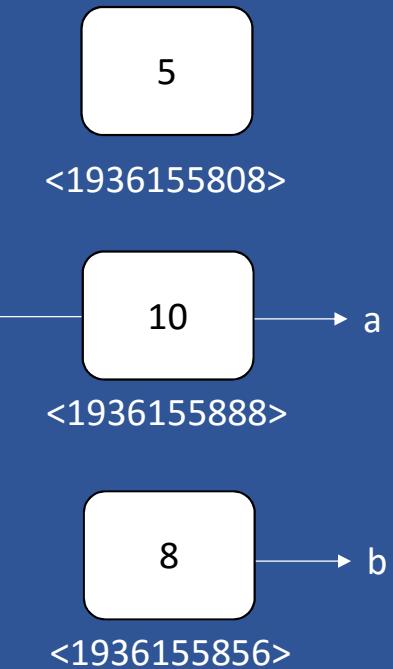


Python Programming

Variable (Memory Concept)

Concept of Garbage Value

```
>>a=5  
>>b=5  
>>k=a  
>>a=10  
>>b=8  
>>k=10
```



Python Programming

Variable

(Memory Concept)

Type of a Variable

```
>>a=5  
>>type(a)  
<class 'int'>  
>>b=4.6  
<class  
'float'>
```

Python Programming

Data Types

Python Programming

Data Types

None

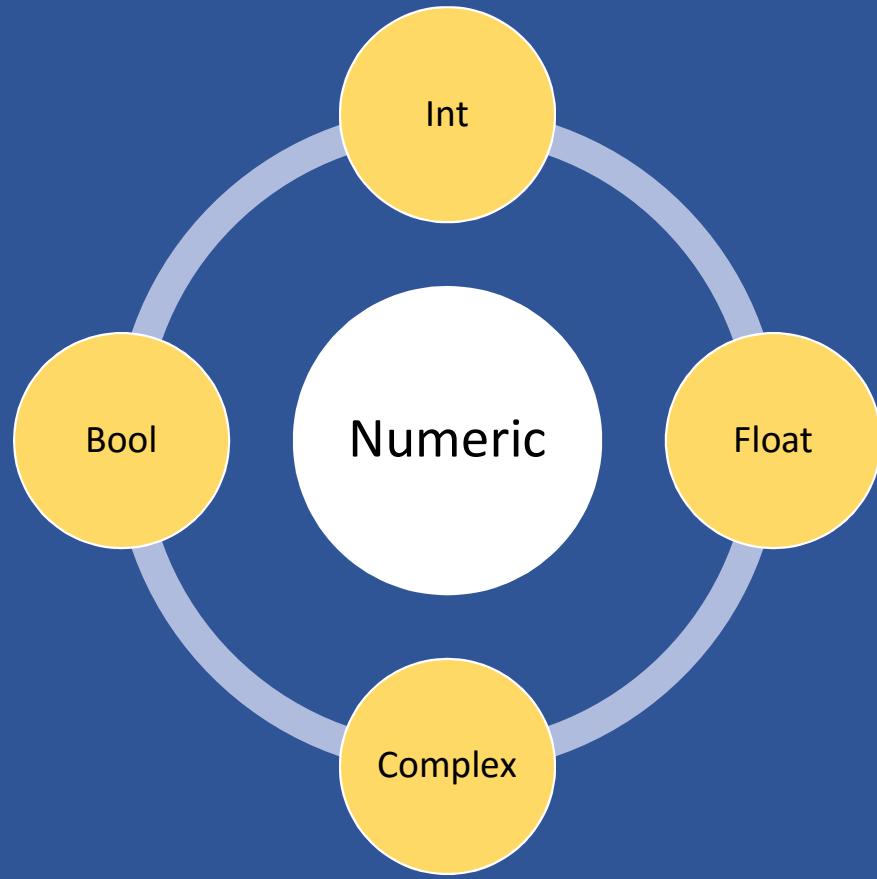
Numeric

Sequence

Dictionary

Python Programming

Data Types



Python Programming

Data Types

Numeric Examples

INT

```
>>num=5  
>>type(num)  
<class 'Int'>
```

FLOAT

```
>>num=5.7  
>>type(num)  
<class 'float'>
```

Complex

```
>>num = 6+9j  
>>type(num)  
<class 'complex'>
```

BOOL

```
>>a=5  
>>b=6  
>>a<b  
True
```

Python Programming

Data Types

Data Types Conversions

INT → FLOAT

```
>>num=5  
>>float(num)  
>>num  
5.0
```

FLOAT → INT

```
>>num=5.7  
>>int(num)  
>>num  
5
```

INT → COMPLEX

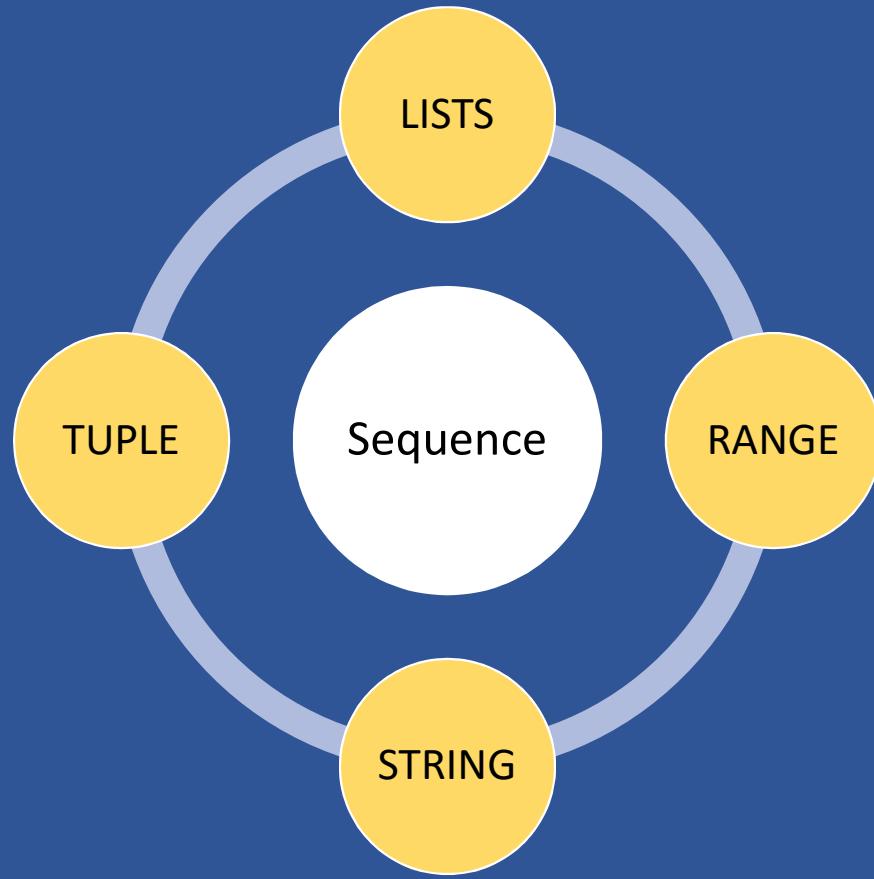
```
>>a = 6  
>>b = 7  
>>c = complex(a,b)  
>>c  
6+7j
```

BOOL → INT

```
>>a=5  
>>b=6  
>>c = a<b  
>>int(c)  
1
```

Python Programming

Data Types



Python Programming

Data Types

Sequence Examples

LISTS

```
>>a= [1,2,3,4]  
>>type(a)  
<class 'List'>
```

TUPLE

```
>>a=(1,2,3,4)  
>>type(a)  
<class 'Tuple'>
```

STRING

```
>>str = 'Harminder'  
>>type(str)  
<class 'String'>
```

RANGE

```
>>a=range(0,10,2)  
>>type(a)  
<class 'Range'>
```

Python Programming

Data Types

Dictionary

Definition

```
>>a= {'name':'Harminder','class':'1st'}  
>>type(a)  
<class 'Dict'>
```

Accessing Keys

```
>>a= {'name':'Harminder','class':'1st'}  
>>a.keys()  
dict_keys[['name','class']]
```

Accessing Values

```
>>a= {'name':'Harminder','class':'1st'}  
>>a.values()  
dict_values[['Harminder','1st']]
```

Accessing Specific Index

```
>>a= {'name':'Harminder','class':'1st'}  
>>a['class']  
'1st'
```

Python Programming

OPERATORS

Python Programming

Operators

Arithmetic Operators

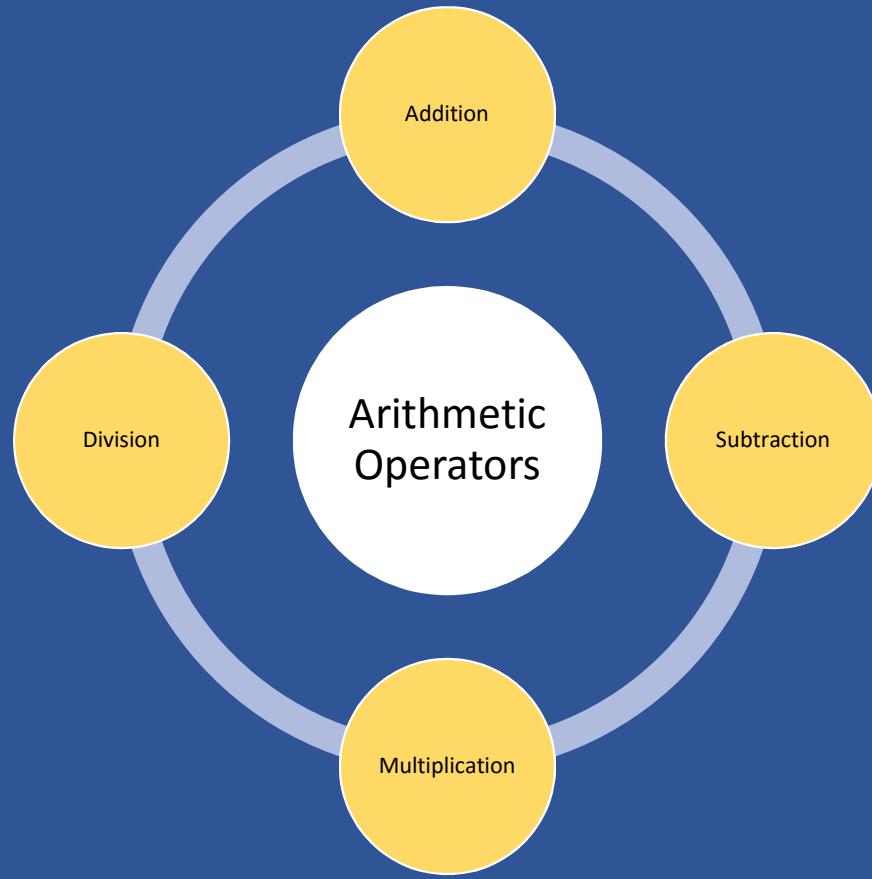
Assignment Operators

Relational Operators

Logical Operators

Python Programming

Operators



Python Programming

Operators

Arithmetic Operators

Addition

```
>>a=5  
>>b=6  
>>a+b  
11
```

Subtraction

```
>>a=5  
>>b=6  
>>b-a  
1
```

Multiplication

```
>>a=5  
>>b=6  
>>a*b  
30
```

Division

```
>>a=30  
>>b=5  
>>a/b  
6
```

Python Programming

Operators



Python Programming

Operators

Assignment Operators

Addition Assignment

```
>>a=5  
>>a += 2  
>>a  
7
```

Subtraction Assignment

```
>>a=5  
>>a-=2  
>>a  
3
```

Multiplication Assignment

```
>>a=5  
>>a*=3  
>>a  
15
```

Division Assignment

```
>>a=25  
>>a /= 5  
>>a  
5.0
```

Python Programming

Operators

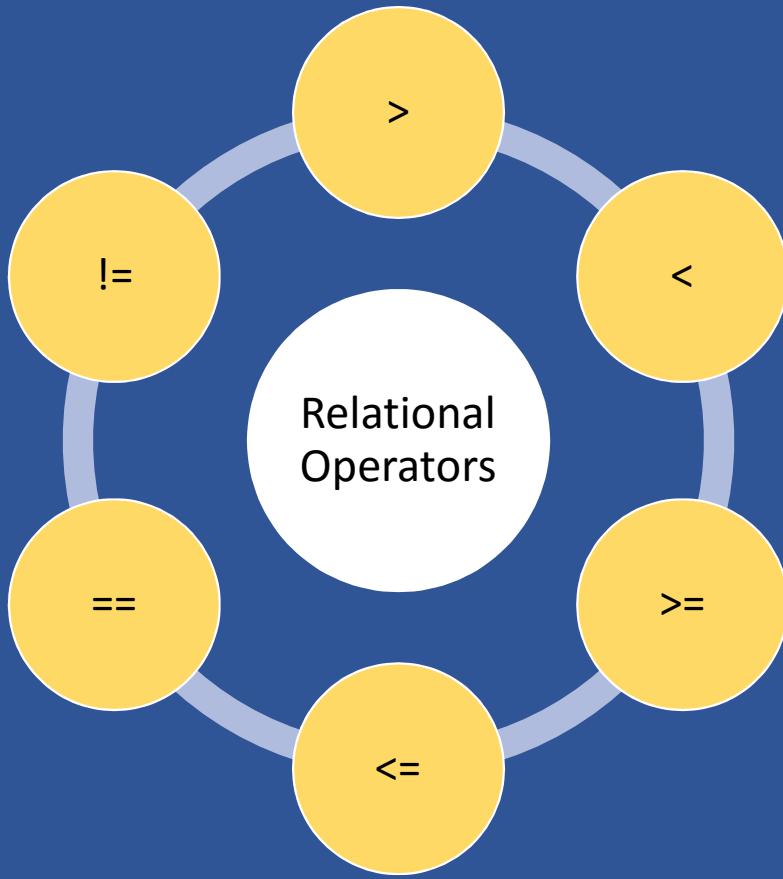
Assignment Operators

Assigning Multiple Variables at once

```
>>a,b=5,8  
>>a  
5  
>>b  
8
```

Python Programming

Operators



Python Programming

Operators

Relational Operators

<

```
>>a=5  
>>b=2  
>>a<b  
False
```

>

```
>>a=5  
>>b=2  
>>a>b  
True
```

>=

```
>>a=5  
>>b=2  
>>a>=b  
True
```

<=

```
>>a=5  
>>b=2  
>>a<=b  
False
```

Python Programming

Operators

Relational Operators

==

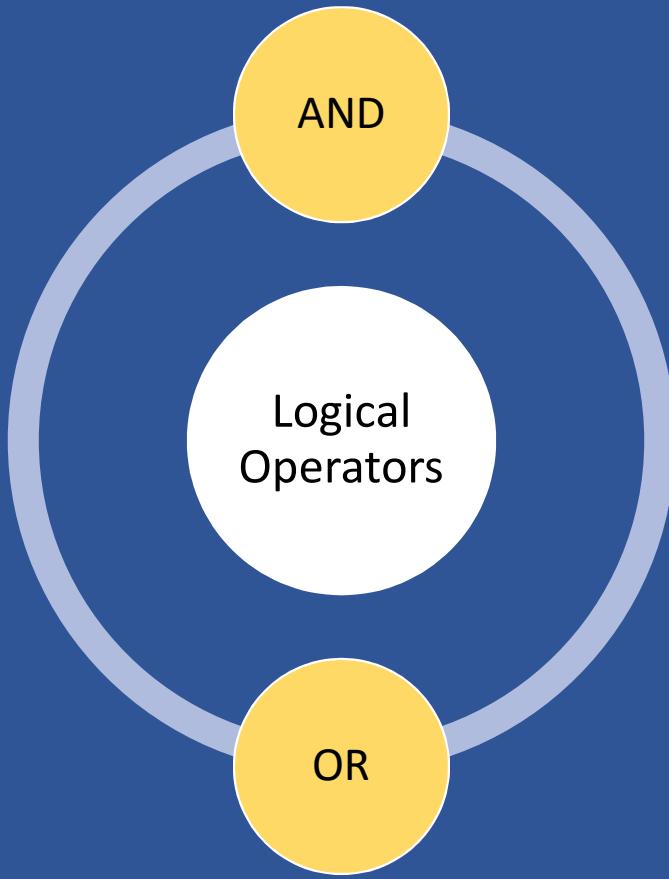
```
>>a=5  
>>b=5  
>>a==b  
True
```

!=

```
>>a=5  
>>b=2  
>>a!=b  
True
```

Python Programming

Operators



Python Programming

Operators

Logical Operators

AND

```
>>a=5  
>>b=2  
>>a>5 and b=2  
False
```

OR

```
>>a=5  
>>b=2  
>>a>5 or b=2  
True
```

Python Programming

BITWISE OPERATOR

Python Programming

Bitwise Operators

AND (&)

OR (|)

XOR (^)

Left Shift (<<)

Right Shift (>>)

Python Programming

Bitwise Operators

Decimal to Binary Conversion

12 → 1100

2	12	
2	6	0
2	3	0
	1	1

Python Programming

Bitwise Operators

Binary to Decimal Conversion

1100 → 12

1 1 0 0
 $2^3 + 2^2 + 2^1 + 2^0$

$$8+4=12$$

Python Programming

Bitwise Operators

Bitwise (AND)

$$12 \& 13 = 12$$

`00001100` -> 12

`00001101` -> 13

`00001100` -> 12

Python Programming

Bitwise Operators

Bitwise (OR)

$$12 \mid 13 = 13$$

`00001100 -> 12`

`00001101 -> 13`

`00001101 -> 13`

Python Programming

Bitwise Operators

Bitwise (XOR)

$12 \wedge 13 = 1$

$00001100 \rightarrow 12$

$00001101 \rightarrow 13$

$00000001 \rightarrow 1$

Python Programming

Bitwise Operators

Left Shift (<<)

$$10 << 2 = 40$$

0000`1010`.000 -> 10
0000`101000`.0 -> 40

Python Programming

Bitwise Operators

Right Shift (>>)

$$10 >> 2 = 2$$

00001010.000 -> 10
000010.10000 -> 2

Python Programming

Math Module

Python Programming

Importing Math Module

Math Module

>>Import math

Python Programming

Math Module

Finding Square Root

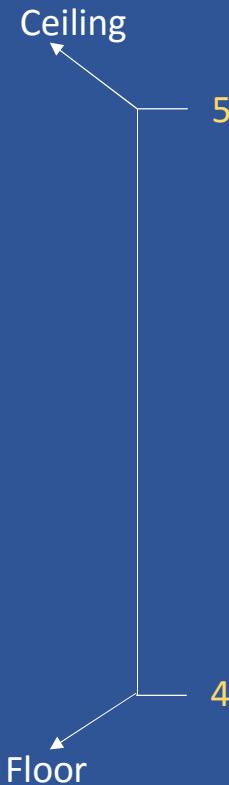
```
>>import math  
>>x=math.sqrt(25)
```

```
>>x
```

```
5
```

Python Programming

Math Module



Math Functions

Floor

```
>>x=math.floor(4.9)  
>>x  
4
```

Ceil

```
>>x=math.ceil(4.1)  
>>x  
5
```

Power

```
>>x=math.pow(4,2)  
>>x  
16
```

Python Programming

Alice Math Module

Math Module

```
>>import math as m  
>>x=m.sqrt(25)  
>>x  
5
```

Python Programming

Importing Specific functions of Math Module

Math Module

>>from math import sqrt

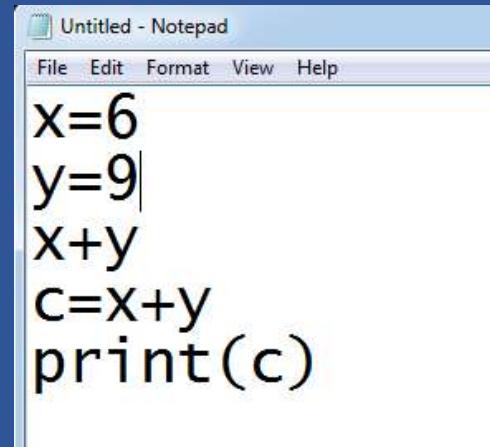
Python Programming

Creating & Running Python Files

Python Programming

Creating & Running Py Files

Write a Program on Notepad/IDE



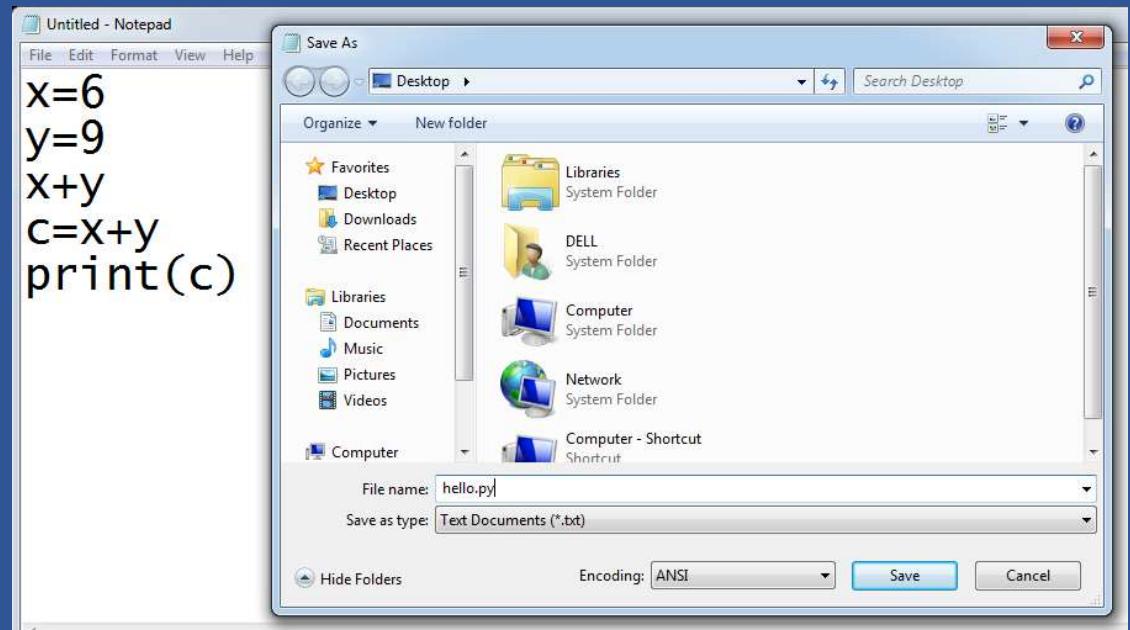
A screenshot of the Microsoft Notepad application window titled "Untitled - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main text area contains the following Python code:

```
x=6  
y=9  
x+y  
c=x+y  
print(c)
```

Python Programming

Creating & Running Py Files

Save file with PY Extension



Python Programming

Creating & Running Py Files

Open CMD and Change path to file's
location

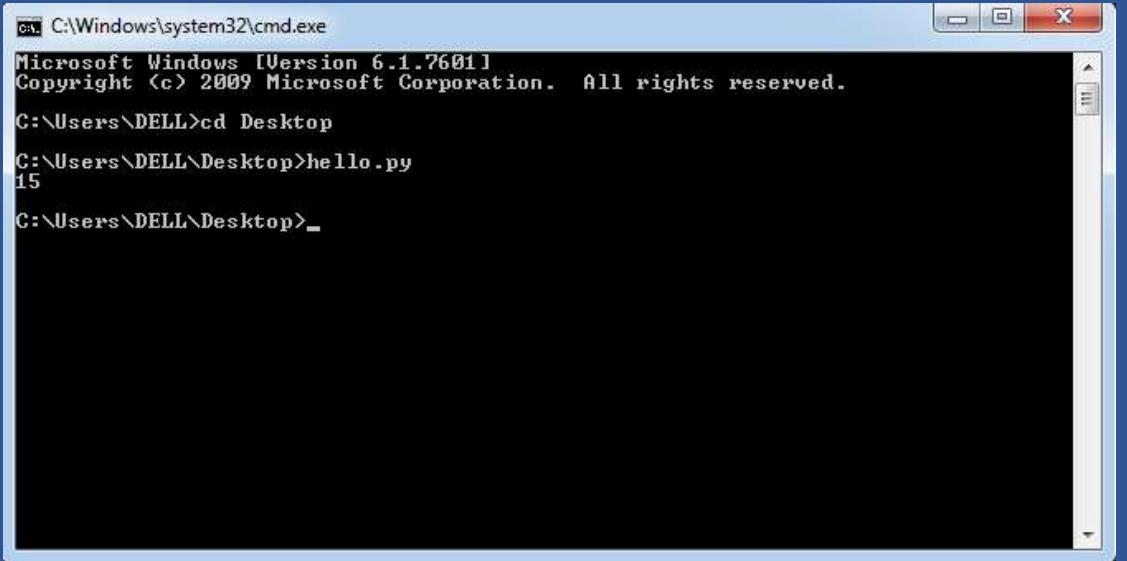


A screenshot of a Windows Command Prompt window titled "cmd C:\Windows\system32\cmd.exe". The window shows the following text:
Microsoft Windows [Version 6.1.7601]
Copyright © 2009 Microsoft Corporation. All rights reserved.
C:\Users\DELL>cd Desktop
C:\Users\DELL\Desktop>

Python Programming

Creating & Running Py Files

Call the python file



A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The window shows the following text:
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\DELL>cd Desktop
C:\Users\DELL\Desktop>hello.py
15
C:\Users\DELL\Desktop>

Python Programming

User Input

Python Programming

User Input

Input Function

```
x=input("Please Enter Your Input")
```

Python Programming

User Input

Input Function only accept strings

```
x=input("Please Enter Your Input")  
Print(type(a))
```

```
Please Enter Your Input 1  
<class 'str'>
```

Python Programming

User Input

Input Function only accept strings

```
x=input("Please Enter First Number")  
y=input("Please Enter Second Number")  
c=x+y  
print(c)
```

```
Please Enter First Number 1  
Please Enter Second Number 2  
12
```

Python Programming

User Input

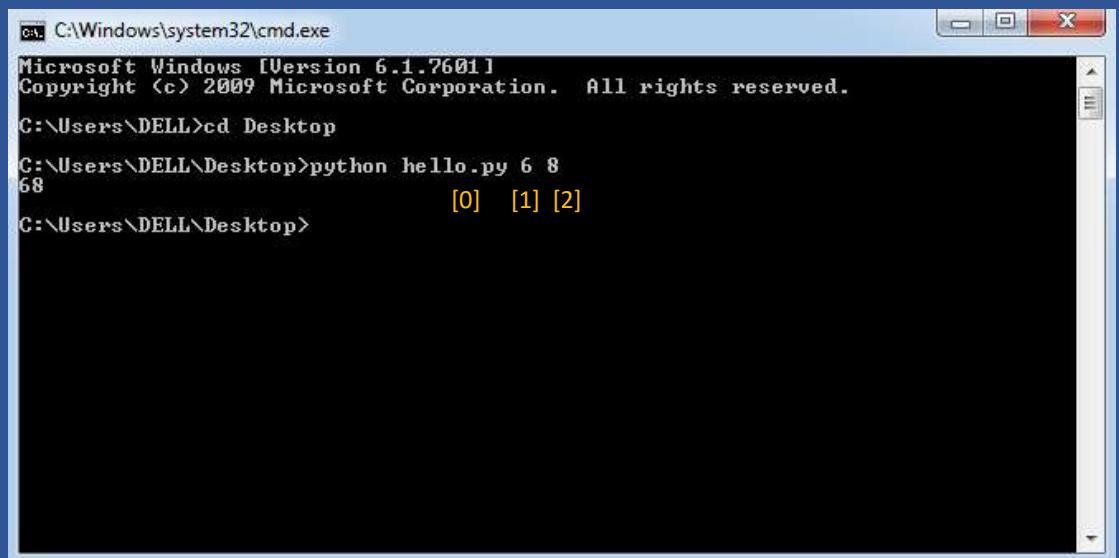
Passing Argument Input in CMD

```
Import sys  
x=sys.argv[1]  
y=sys.argv[2]  
c=x+y  
print(c)
```

Python Programming

User Input

Passing Argument Input in CMD



A screenshot of a Windows Command Prompt window titled "cmd C:\Windows\system32\cmd.exe". The window shows the following command-line session:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd Desktop
C:\Users\DELL\Desktop>python hello.py 6 8
68
[0] [1] [2]
C:\Users\DELL\Desktop>
```

Python Programming

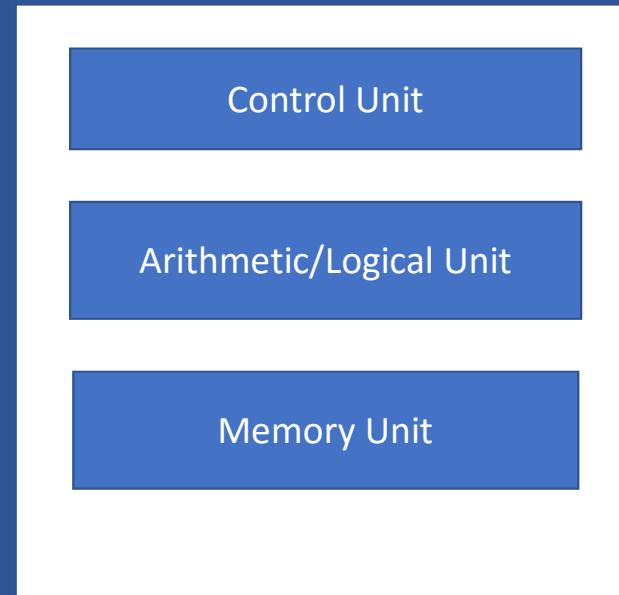
Control Flow Statements



Python Programming

Control Flow Statements

Central Processing Unit



Python Programming

Control Flow Statements

IF Statement Needs Indentation

```
X=5
If x==3:
    print("equal to five")
print("hello")
```

Python Programming

Control Flow Statements

Else Statement

```
X=5
if x==5:
    print("equal to five")
else:
    print("Not Equal")
```

Python Programming

Control Flow Statements

Nested IF Statement

```
X=5
if x>=5:
    print("x is greater")
    if x==5:
        print("x is equal")
else:
    print("x is smaller")
```

Python Programming

Control Flow Statements

Nested IF Statement

```
X=5
If x==1:
    print("One")
elif x==2:
    print("Two")
elif x==3:
    print("Three")
else:
    print("Wrong Input")
```

Python Programming

Loops



Python Programming

Loops

While Loop

```
x=0           -----> Initialization  
while x<=5:  
    print(i)  
    x=x+1  
                    ↓  
                    Condition  
                    Increment
```

Python Programming

Loops

While Loop (Reverse)

```
x=5           -----> Initialization  
while x>=0:  
    print(i)  
    x=x-1  
                    ↓  
                    Condition  
                    Increment
```

Python Programming

Loops

While Loop (Nested)

```
x=0  
while x<=5:  
    print("Python",end="")  
    j=0  
    while j<=5:  
        print("Rocks",end="")  
        j=j+1  
    x=x+1  
    print()
```

Python Programming

Loops

For Loop with List

```
a = ["Harminder",1,"Surender"]
```

```
for i in a:  
    print(i)
```

Python Programming

Loops

For Loop with String

```
a = "Harminder"
```

```
for i in a:  
    print(i)
```

Python Programming

Loops

For Loop with Tuple

```
a = ("hi","harminder","surender")
for i in a:
    print(i)
```

Python Programming

Loops

For Loop with Sets

```
a = {"hi","harminder","surender"}  
for i in a:  
    print(i)
```

Python Programming

Loops

For Loop with Range

```
for i in range(10):  
    print(i)
```

Python Programming

Loops

For Loop with Range

```
for i in range(10,21,1):  
    print(i)
```

Python Programming

Loops

For Loop with Range

```
for i in range(20,0,-1):  
    print(i)
```

Python Programming

Loops

Nested For Loop

```
for i in range(5):
    for j in range(5):
        print(j,end="")
    print()
```

Python Programming

Loops

Break Statement

```
for i in range(1,10,1):  
    if i==5:  
        break  
    print(i)
```

Python Programming

Loops

Continue Statement

```
for i in range(1,10,1):
    if i==5:
        continue
    print(i)
```

Python Programming

Loops

Pass Statement

```
for i in range(1,100,1):
    if i%2!=0:
        pass
    else:
        print(i)
```

Python Programming

Loops

For Else

```
a=[1,2,3,4,6,7]
for i in a:
    if i%5==0:
        break
else:
    print("Not Found")
```

Python Programming

Functions

Python Programming

Functions

Function Definition

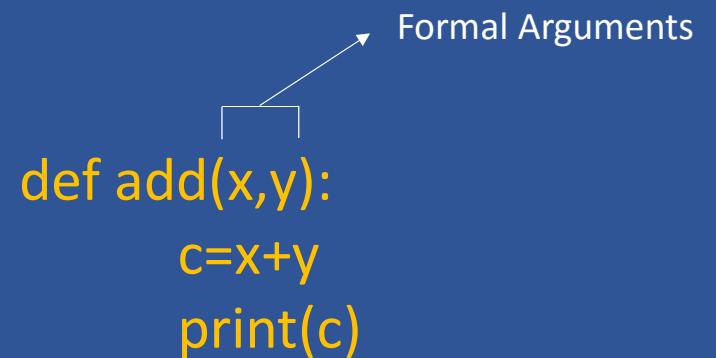
```
def greet():
    print("Hello")
    print("Good Morning")
```

Python Programming

Functions

Passing Parameter to function

```
def add(x,y):  
    c=x+y  
    print(c)
```



```
add(4,5)
```

Python Programming

Functions

Types of arguments

Position

Keyword

Default

Variable length

Python Programming

Functions

Position Argument

```
def person(name,age):  
    print(name)  
    print(age)
```

```
person("Harminder",29)
```

Python Programming

Functions

Keyword Argument

```
def person(name,age):  
    print(name)  
    print(age)  
person(age=11,name="Harminder")
```

Python Programming

Functions

Default Argument

```
def person(name,age):  
    print(name)  
    print(age)  
person(age=11,name="Harminder")
```

Python Programming

Functions

Variable Length Argument

```
def sum(a,*b):  
    for i in b:  
        c=a+i  
    print(c)
```

```
sum(2,4,6)
```

Python Programming

Functions

Keyworded Variable Length Argument

```
def person(a,**b):  
    print(a)  
    print(b)
```

```
person("Harminder",city="faridabad",age=19)
```

Python Programming

Functions

Keyworded Variable Length Argument

```
def person(a,**b):  
    print(a)  
    for i,j in b.items():  
        print(i,j)
```

```
person("Harminder",city="faridabad",age=19)
```

Python Programming

Functions

Returning values from function

```
def add(x,y):  
    c=x+y  
    return c
```

```
a=add(4,5)  
print(a)
```

Python Programming

Functions

Returning multiple values from function

```
def add_sub(x,y):  
    c=x+y  
    d=x-y  
    return c,d
```

```
a,b=add_sub(4,5)  
print(a,b)
```

Python Programming

Functions

Global & Local Variables

```
a=10  
def hello():  
    a=15  
    print(a)
```

hello()

print(a)

Python Programming

Functions

Local Variables can only be used inside
function

```
def hello():
    a=15
    print(a)
```

```
print(a)
```

This code will generate a Error

Python Programming

Functions

Global Variables can be used anywhere in
Program

```
a=10  
def hello():  
    print(a)
```

```
hello()
```

Python Programming

Functions

Changing Value of a global Variable

```
a=10  
def hello():  
    global a  
    a=15  
    print(a)
```

```
hello()  
print(a)
```

Python Programming

Functions

Passing List/tuple/set to a function

```
def hello(a,b,c):  
    print(a)  
    print(b)  
    print(c)
```

```
a=[1,2,3,4,5]  
b=(1,2,3,4,5)  
c={1,2,3,4,5}
```

```
hello(a,b,c)
```

Python Programming

Functions

Anonymous Function(LAMBDA)

f= lambda a,b:a+b

result = f(5,6)

print(result)

Python Programming

Functions

Using Filter with lambda

```
nums = [2,3,45,6,7,8,80]
```

```
r= filter(lambda n:n%2==0,nums)
```

```
for i in r:  
    print(i)
```

Python Programming

Functions

Using Map with lambda

```
nums = [2,3,45,6,7,8,80]
```

```
r= map(lambda n:n*2,nums)
```

```
for i in r:  
    print(i)
```

Python Programming

Functions

Using Reduce with lambda

```
from functools import reduce  
nums = [2,3,45,6,7,8,80]  
  
r= reduce(lambda a,b:a+b,nums)  
  
print(r)
```

Python Programming

Functions

Creating Modules in Python

```
def add(a,b):  
    c=a+b  
    return c
```

```
def sub(a,b):  
    c=a-b  
    return c
```

```
def mul(a,b):  
    c=a*b  
    return c
```

Python Programming

Functions

Using User Defined Modules in Python

```
import hello as h
```

```
r=h.add(3,4)
```

```
print(r)
```

Python Programming

Special Variable

Python Programming

```
import hi as h

def fun1():
    print("This is function 1")
    h.add()

def fun2():
    print("This is function 2")

def main():
    fun1()
    fun2()
```

main()

```
def add():
    print("This is add function")

def mul():
    print("This is mul function")

def hello():
    add()
    mul()

if __name__ == "__main__":
    hello()
```

Python Programming

Object Oriented Programming

Python Programming

OOP

Class Definition

```
class a:  
    def add(self):  
        print("This is add function")
```

```
obj = a()  
a.add(obj)  
obj.add()
```

Python Programming

OOP

Multiple Object of a class

class a:

 def add(self):

 print("This is add function")

obj = a()

obj2=a()

obj.add()

obj2.add()

Python Programming

OOP

__init__ Method

class a:

def __init__(self)

print("This is init function")

obj = a()

Python Programming

OOP

Passing Arguments to a Method

```
class person:  
    def a(self,name):  
        print("Hi",name)
```

```
obj = person()  
obj.a("Harminder")
```

Python Programming

OOP

Accessing Variable

```
class person:
```

```
    x=1
```

```
obj = person()  
print(obj.x)
```

Python Programming

Binding variable to object

OOP

```
class a:  
    def b(self,k=5,n=4):  
        self.k=k  
        self.n=n  
    def c(self):  
        print(self.k,self.n)
```

```
obj = a()  
obj.b(44,67)  
obj.c()
```

```
obj2 = a()  
obj2.b()  
obj2.c()
```

Python Programming

Binding variable to object using
`_init_`

OOP

```
class a:  
    def __init__(self,k=5,n=4):  
        self.k=k  
        self.n=n  
    def c(self):  
        print(self.k,self.n)
```

```
obj = a(44,67)  
obj.c()
```

```
obj2 = a()  
obj2.c()
```

Python Programming

Instance Variable

OOP

class a:

```
def __init__(self):  
    self.b=5
```

```
c1=a()
```

```
c2=a()
```

```
print(c1.b)
```

```
print(c2.b)
```

```
c1.b=10
```

```
print(c1.b)
```

```
print(c2.b)
```

Python Programming

Class Variable

OOP

```
class a:  
    x=4  
    def __init__(self):  
        self.b=5
```

```
c1=a()  
c2=a()  
print(c1.x)  
print(c2.x)  
a.x=15  
print(c1.x)  
print(c2.x)
```

```
c1.x=55  
print(c1.x)  
print(c2.x)
```

Python Programming

Types of Methods

(Instance Methods)

OOP

```
class student:  
    def __init__(self,m1,m2,m3):  
        self.m1=m1  
        self.m2=m2  
        self.m3=m3  
  
    def avg(self):  
        return (self.m1+self.m2+self.m3)/3  
  
s1 = student(23,56,44)  
s2 = student(90,89,45)  
print(s1.avg())  
print(s2.avg())
```

Python Programming

Types of Methods

(Instance Methods)

Accessors

Mutators

```
class student:  
    def __init__(self):  
        self.a="Harminder"  
  
    def get_a(self):  
        print(self.a)
```

```
s1 = student()  
s1.get_a()
```

OOP

Python Programming

Types of Methods

(Instance Methods)



OOP

```
class student:  
    def __init__(self):  
        self.a="Harminder"
```

```
def set_a(self):  
    self.a="surender"  
    return self.a
```

```
s1 = student()  
print(s1.a)  
print(s1.set_a())
```

Python Programming

Types of Methods

(Class Methods)

OOP

```
class student:  
    school="vsics"  
  
    @classmethod  
    def get_school(cls):  
        print(cls.school)  
  
s1=student()  
student.get_school()
```

Python Programming

Types of Methods

(Static Methods)

OOP

```
class student:  
    @staticmethod  
    def a():  
        print("hi")
```

```
s1=student()  
s1.a()
```

Python Programming

Inner Class

OOP

```
class student:  
    def a(self):  
        print("hi")  
        self.obj = self.b()  
        self.obj.hello()
```

```
class b:  
    def hello(self):  
        print("hello")
```

```
s1=student()  
s1.a()
```

Python Programming

Inner Class

(using Object of inner class outside main class)

OOP

```
class student:  
    def a(self):  
        print("hi")  
        self.obj = self.b()
```

```
class b:  
    def hello(self):  
        print("hello")
```

```
s1=student()  
s1.a()  
s1.obj.hello()
```

Python Programming

Inner Class

(Defining Object of inner class outside main class)

OOP

```
class student:  
    def a(self):  
        print("hi")
```

```
class b:  
    def hello(self):  
        print("hello")
```

```
s1=student()  
obj=s1.b()  
obj.hello()
```

Python Programming

Inheritance

OOP

```
class a:  
    def feature1(self):  
        print("Feature 1 is working")  
  
    def feature2(self):  
        print("Feature 2 is working")  
  
class b(a):  
    def feature3(self):  
        print("Feature 3 is working")  
  
    def feature4(self):  
        print("Feature 4 is working")  
  
obj1 = b()  
obj1.feature1()
```

Python Programming

Multi Level Inheritance

OOP

```
class a:  
    def feature1(self):  
        print("Feature 1 is working")  
  
    def feature2(self):  
        print("Feature 2 is working")  
  
class b(a):  
    def feature3(self):  
        print("Feature 3 is working")  
  
    def feature4(self):  
        print("Feature 4 is working")  
  
class c(b):  
    def feature5(self):  
        print("Feature 5 is working")  
  
obj1 = c()  
obj1.feature1()
```

Python Programming

Multiple Inheritance

OOP

```
class a:  
    def feature1(self):  
        print("Feature 1 is working")  
  
    def feature2(self):  
        print("Feature 2 is working")  
  
class b:  
    def feature3(self):  
        print("Feature 3 is working")  
  
    def feature4(self):  
        print("Feature 4 is working")  
  
class c(a,b):  
    def feature5(self):  
        print("Feature 5 is working")  
  
obj1 = c()  
obj1.feature1()
```

Python Programming

OOP

Constructor Behavior in Single/Multi level inheritance

```
class a:  
    def __init__(self):  
        print("Init of a")  
    def feature1(self):  
        print("This is feature 1")  
    def feature2(self):  
        print("This is Feature 2")
```

```
class b(a):  
    def __init__(self):  
        super().__init__()  
        print("Init of b")  
    def feature3(self):  
        print("This is feature 3")  
    def feature4(self):  
        print("This is Feature 4")
```

```
k = b()
```

Python Programming

OOP

Constructor Behavior in Multiple Inheritance (MRO)

```
class a:  
    def __init__(self):  
        super().__init__()  
        print("Init of a")  
    def feature1(self):  
        print("This is feature 1")  
    def feature2(self):  
        print("This is Feature 2")  
  
class b:  
    def __init__(self):  
        super().__init__()  
        print("Init of b")  
    def feature3(self):  
        print("This is feature 3")  
    def feature4(self):  
        print("This is Feature 4")  
  
class c(a,b):  
    def __init__(self):  
        super().__init__()  
        print("Init of c")  
    def feat(self):  
        print("This is feat")  
  
k = c()
```

Python Programming

OOP

Polymorphism

Duck Typing

Operator Overloading

Method Overriding

Python Programming

Using methods in other classes

OOP

class b:

def k(self):

print("This is k function")

class a:

def a(self,obj2):

obj2.k()

obj2=b()

obj = a()

obj.a(obj2)

Python Programming

Duck Typing

OOP

```
class b:  
    def k(self):  
        print("This is k function")
```

```
class a:  
    def a(self,obj2):  
        obj2.k()
```

```
class d:  
    def k(self):  
        print("This is k in d")
```

```
obj2=d()  
obj = a()  
obj.a(obj2)
```

Python Programming

OOP

Operator Overloading

Operators

5 + 2

Operands

Python Programming

OOP

Operator Overloading (Everything in python is a class)

```
a=4  
b=5  
c=a+b  
print(c)
```

```
print(int.__add__(a,b))
```

Python Programming

OOP

Operator Overloading (Int class has various methods)

+

• `__add__()`

-

• `__sub__()`

*

• `__mul__()`

Python Programming

OOP

Overloading Addition Operator

```
class a:  
    def __init__(self,m1,m2):  
        self.m1=m1  
        self.m2=m2  
    def __add__(obj1,obj2):  
        x = obj1.m1+obj2.m1  
        y = obj1.m2+obj2.m2  
        z = a(x,y)  
        return z
```

```
s1 = a(3,4)  
s2 = a(44,55)
```

```
s3 = s1+s2  
print(s3.m1)
```

Python Programming

OOP

Overloading Greater than Operator

```
class a:  
    def __init__(self,m1,m2):  
        self.m1=m1  
        self.m2=m2  
    def __gt__(obj1,obj2):  
        x = obj1.m1+obj1.m2  
        y = obj2.m1+obj2.m2  
        if x>y:  
            return True  
        else:  
            return False
```

```
s1 = a(3,4)  
s2 = a(44,55)  
  
if s1>s2:  
    print("s1 wins")  
else:  
    print("s2 wins")
```

Python Programming

OOP

Method Overriding

```
class a:  
    def greet(self):  
        print("Welcome to class a")  
  
class b(a):  
    def greet(self):  
        print("Welcome to class b")  
  
obj = b()  
obj.greet()
```

Python Programming

OOP

Iterator

```
a = [2,33,45,67,890,3]
```

```
c = iter(a)
```

```
print(c.__next__())
```

```
for i in a:
```

```
    print(c.__next__())
```

Python Programming

OOP

Generators Example 1

```
def hello():
    yield 1
    yield 2
    yield 3

values=hello()
print(values.__next__())
print(values.__next__())
print(values.__next__())
```

Python Programming

OOP

Generators Example 2

```
def sq():
    n=1
    while n<=10:
        yield n*n
        n+=1

values=sq()

print(next(values))
for i in values:
    print(i)
```

Python Programming

Exception Handling

Python Programming

Exception Handling

Types of Errors

Compile Time

Logical

Runtime Error

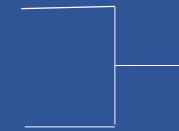
Python Programming

Exception Handling

Types of Statements

a=25

b=5



Normal Statement

c=a/b



Critical Statement

Python Programming

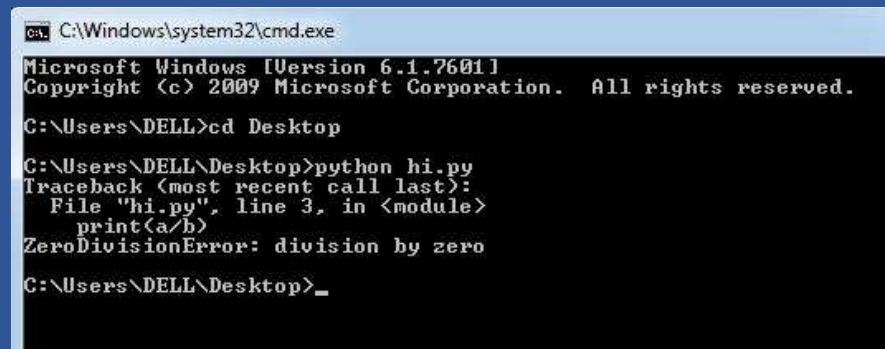
Exception Handling

Runtime Error Example

a=25

b=0

print(a/b)



A screenshot of a Windows command prompt window titled 'C:\Windows\system32\cmd.exe'. The window shows the following text:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd Desktop
C:\Users\DELL\Desktop>python hi.py
Traceback (most recent call last):
  File "hi.py", line 3, in <module>
    print(a/b)
ZeroDivisionError: division by zero
C:\Users\DELL\Desktop>
```

Python Programming

Exception Handling

Runtime Error Example

a=5

b=0

try:

 print(a/b)

except Exception:

 print("You cannot divide a number
 by zero")

print("bye")

Python Programming

Exception Handling

Try/Except/finally

a=5

b=2

try:

```
    print("Calculation mode started")
    print(a/b)
```

except Exception:

```
    print("You cannot divide a number by
zero")
```

finally:

```
    print("Calculation mode closed")
```

Python Programming

Exception Handling

Handling Specific Errors

a=5

b=2

try:

```
    print("Calculation mode started")
    print(a/b)
```

except ZeroDivisionError:

```
    print("You cannot divide a number by
zero")
```

finally:

```
    print("Calculation mode closed")
```

Python Programming

Multi Threading

Python Programming

Multi Threading

Multi Threading

```
from threading import *
from time import sleep
class hello(Thread):
    def run(self):
        for i in range(0,50):
            print("hello")
            sleep(1)
```

```
class hi(Thread):
    def run(self):
        for i in range(0,50):
            print("hi")
            sleep(1)
```

```
t1=hello()
t2=hi()
```

```
t1.start()
sleep(0.2)
t2.start()
```

Python Programming

Concept of Join

Multi Threading

```
from threading import *
from time import sleep
class hello(Thread):
    def run(self):
        for i in range(0,10):
            print("hello")
            sleep(1)
```

```
class hi(Thread):
    def run(self):
        for i in range(0,10):
            print("hi")
            sleep(1)
```

```
t1=hello()
t2=hi()
```

```
t1.start()
sleep(0.2)
t2.start()
```

```
t1.join()
print("bye")
```

Python Programming

File Handling

Python Programming

File Handling

Opening a file in Python

`open("filename","mode")`

"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

Python Programming

File Handling

Reading Complete file

```
f=open("hello.txt","r")  
print(f.read())
```

Python Programming

File Handling

Reading bits of a file

```
f=open("hello.txt","r")  
print(f.read(6))
```

Python Programming

File Handling

Reading one line at a time

```
f=open("hello.txt","r")  
print(f.readline())  
print(f.readline())
```

Python Programming

File Handling

Reading Bits of a line

```
f=open("hello.txt","r")  
print(f.readline())  
print(f.readline(4))
```

Python Programming

File Handling

Writing a file

```
f=open("hello.txt","w")  
f.write("hi who are  
you??")
```

Python Programming

File Handling

Append to a file

```
f=open("hello.txt","a")  
f.write("hi who are  
you??")
```

Python Programming

File Handling

Using for loop with file handler

```
f=open("hello.txt","r")
```

```
f1=open("hi.txt","a")
```

```
for i in f:  
    f1.write(i)
```

Python Programming

File Handling

Removing a file

```
import os  
os.remove("hi.txt")
```

Python Programming

File Handling

Removing a file

```
import os
```

```
if os.path.exists("hello.txt"):  
    os.remove("hello.txt")  
else:  
    print("no file")
```

Python Programming

DJANGO

Python Programming

What is Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Python Programming

Why Django

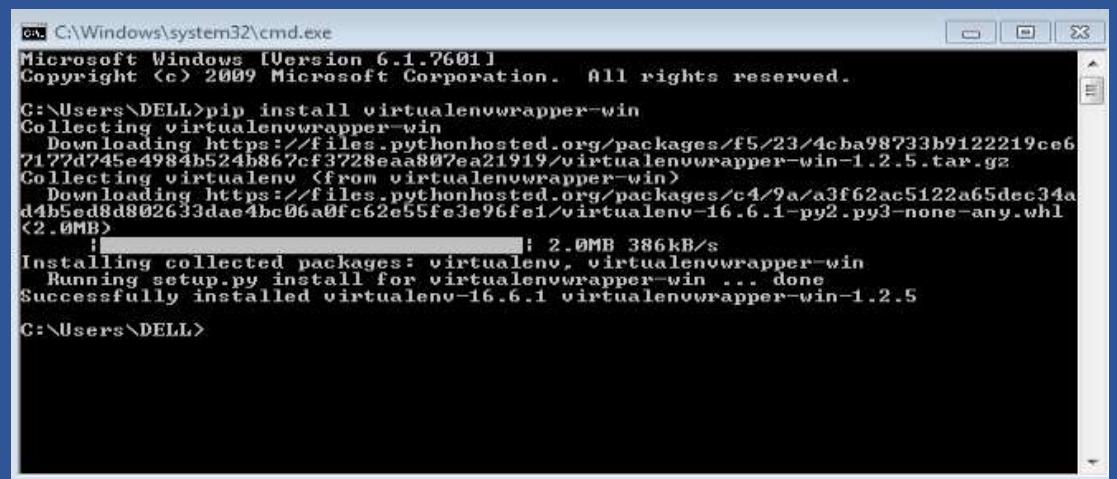
- Fast
- Secure
- Scalable

Python Programming

Installation

Firstly we will install virtual environment wrapper

pip install virtualenvwrapper-win



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright © 2009 Microsoft Corporation. All rights reserved.

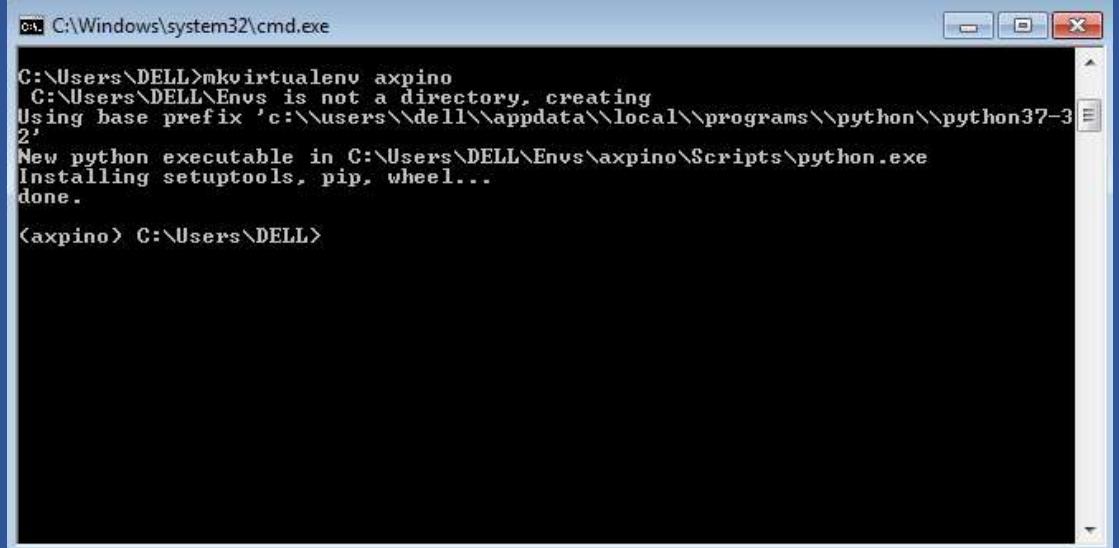
C:\Users\DELL>pip install virtualenvwrapper-win
Collecting virtualenvwrapper-win
  Downloading https://files.pythonhosted.org/packages/f5/23/4cba98733b9122219ce62177d745e4984b524b867cf3728eaa807ea21919/virtualenvwrapper-win-1.2.5.tar.gz
Collecting virtualenv (from virtualenvwrapper-win)
  Downloading https://files.pythonhosted.org/packages/c4/9a/a3f62ac5122a65dec34ad4b5ed8d802633dae4bc06a0fc62e55fe3e96fe1/virtualenv-16.6.1-py2.py3-none-any.whl (2.0MB)
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! : 2.0MB 386kB/s
Installing collected packages: virtualenv, virtualenvwrapper-win
  Running setup.py install for virtualenvwrapper-win ... done
Successfully installed virtualenv-16.6.1 virtualenvwrapper-win-1.2.5
C:\Users\DELL>
```

Python Programming

Installation

Firstly we will create a virtual environment for Django

`mkvirtualenv axpino`



```
C:\Windows\system32\cmd.exe
C:\Users\DELL>mkvirtualenv axpino
C:\Users\DELL\Envs is not a directory, creating
Using base prefix 'c:\\users\\dell\\appdata\\local\\programs\\python\\python37-3
2'
New python executable in C:\Users\DELL\Envs\axpino\Scripts\python.exe
Installing setuptools, pip, wheel...
done.

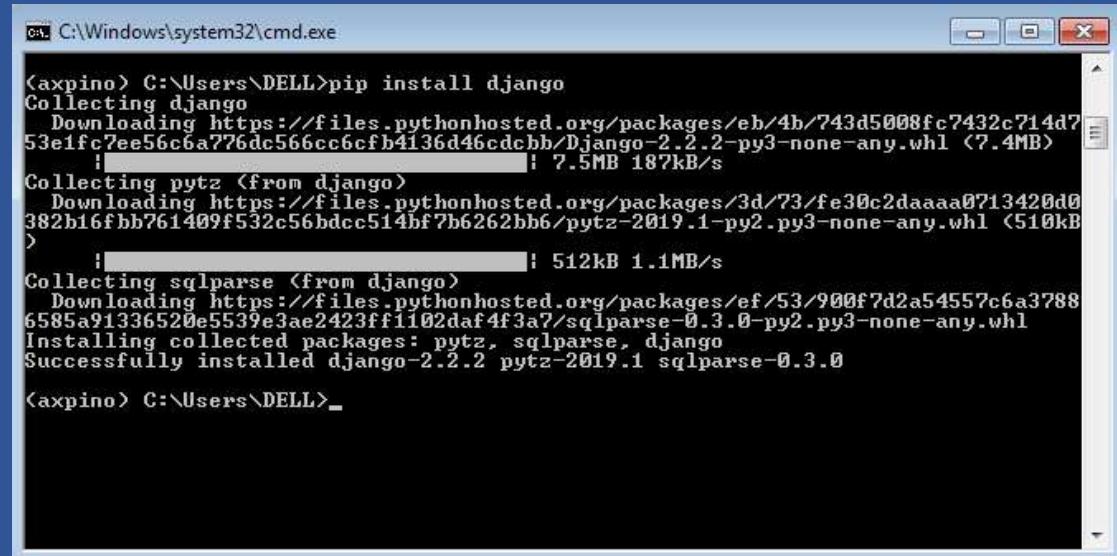
(axpino) C:\Users\DELL>
```

Python Programming

Installation

Now we will install Django into our Environment

pip install django



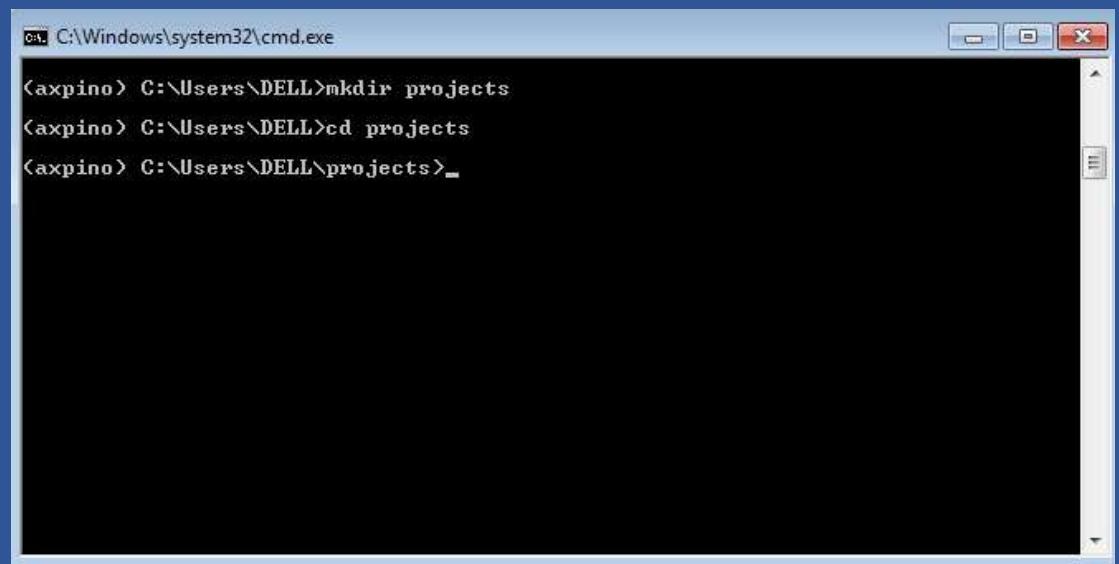
```
C:\Windows\system32\cmd.exe
C:\Users\DELL>pip install django
Collecting django
  Downloading https://files.pythonhosted.org/packages/eb/4b/743d5008fc7432c714d753e1fc7ee56c6a776dc566cc6cfb4136d46cdccb/Django-2.2.2-py3-none-any.whl (7.4MB)
    |████████| 7.5MB 187KB/s
Collecting pytz <from django>
  Downloading https://files.pythonhosted.org/packages/3d/73/fe30c2daaaa0713420d0382b16fbb761409f532c56bdcc514bf7b6262bb6/pytz-2019.1-py3-none-any.whl (510kB)
    |████████| 512kB 1.1MB/s
Collecting sqlparse <from django>
  Downloading https://files.pythonhosted.org/packages/ef/53/900f7d2a54557c6a37886585a91336520e5539e3ae2423ff1102daf4f3a7/sqlparse-0.3.0-py2.py3-none-any.whl
Installing collected packages: pytz, sqlparse, django
Successfully installed django-2.2.2 pytz-2019.1 sqlparse-0.3.0
C:\Users\DELL>
```

Python Programming

Installation

Now we will create a folder to store our projects
and navigate to it

mkdir projects
cd projects



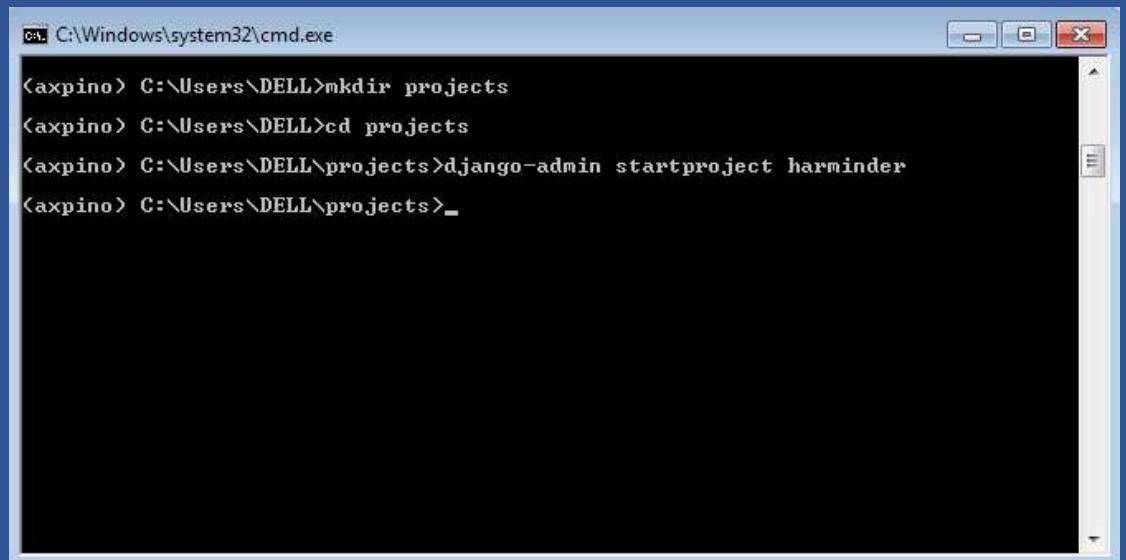
A screenshot of a Windows Command Prompt window titled 'cmd C:\Windows\system32\cmd.exe'. The window shows three commands being run:
<axpino> C:\Users\DELL>mkdir projects
<axpino> C:\Users\DELL>cd projects
<axpino> C:\Users\DELL\projects>_

Python Programming

Installation

Now we will create our first project

django-admin startproject projectname



A screenshot of a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. The window contains the following command-line session:

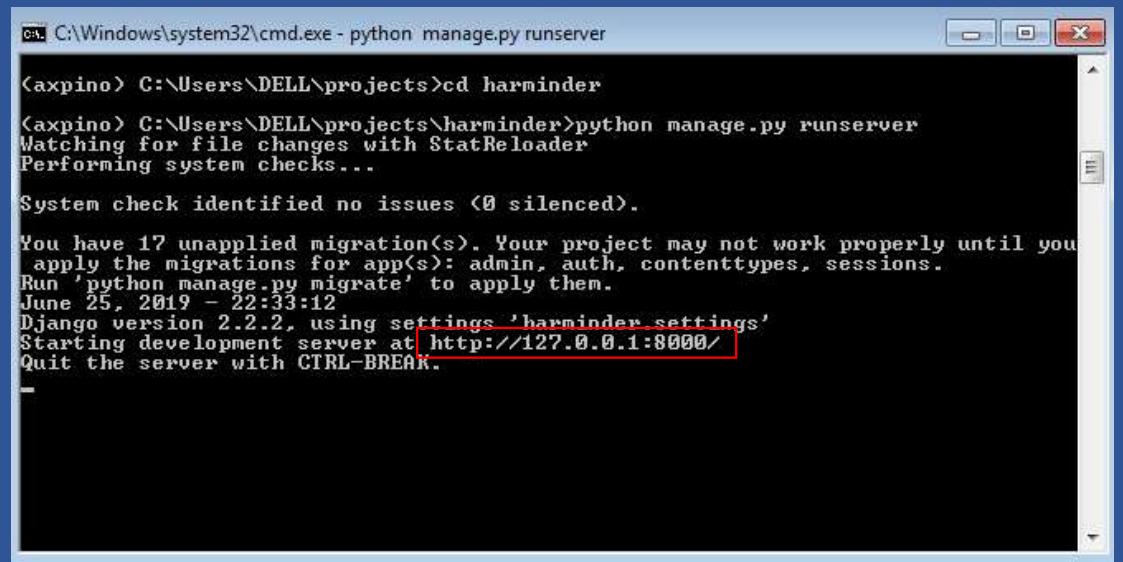
```
C:\Users\DELL>mkdir projects
C:\Users\DELL>cd projects
C:\Users\DELL\projects>django-admin startproject harminder
C:\Users\DELL\projects>_
```

Python Programming

Installation

Navigate to project folder and start server

```
cd projectname  
python manage.py runserver
```



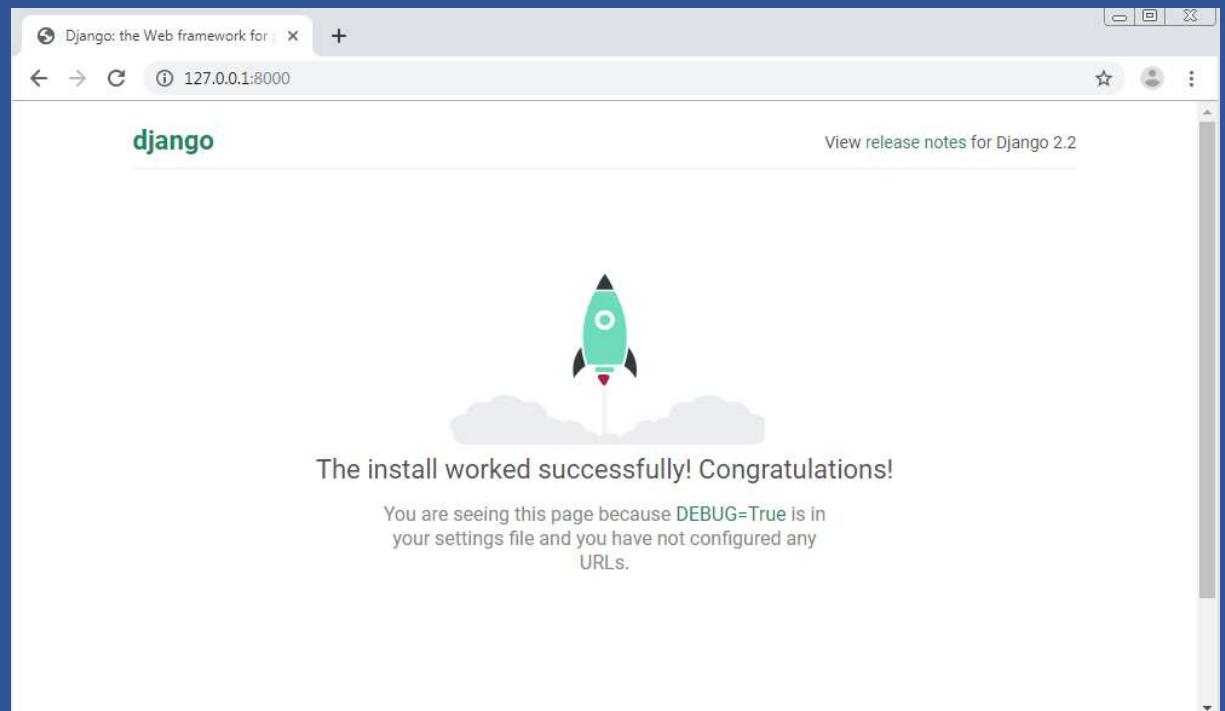
The screenshot shows a Windows Command Prompt window titled 'cmd C:\Windows\system32\cmd.exe - python manage.py runserver'. The command entered is 'python manage.py runserver'. The output indicates that migrations are pending for admin, auth, contenttypes, and sessions. It shows the server is running at 'http://127.0.0.1:8000/'. A red box highlights the URL.

```
C:\Windows\system32\cmd.exe - python manage.py runserver  
(axpino) C:\Users\DELL\projects>cd harminder  
(axpino) C:\Users\DELL\projects\harminder>python manage.py runserver  
Watching for file changes with StatReloader  
Performing system checks...  
  
System check identified no issues (0 silenced).  
  
You have 17 unapplied migration(s). Your project may not work properly until you  
apply the migrations for app(s): admin, auth, contenttypes, sessions.  
Run 'python manage.py migrate' to apply them.  
June 25, 2019 - 22:33:12  
Django version 2.2.2, using settings 'harminder.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.
```

Python Programming

Installation

Now lets access the main url of project



Python Programming

Creating views

Firstly we will create a app inside our project

Select the Environment using command

>>workon env_name

Navigate to projects folder

>>cd projects

Navigate to your project

>>cd project_name

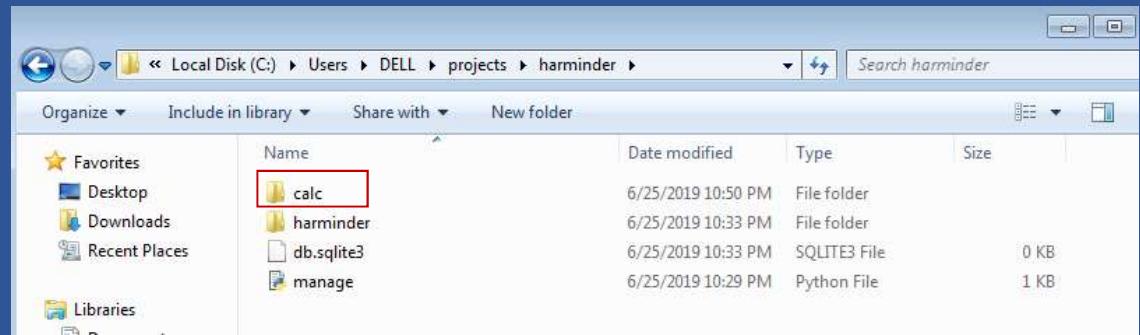
Create a app using command

>>python manage.py startapp app_name

Python Programming

Creating views

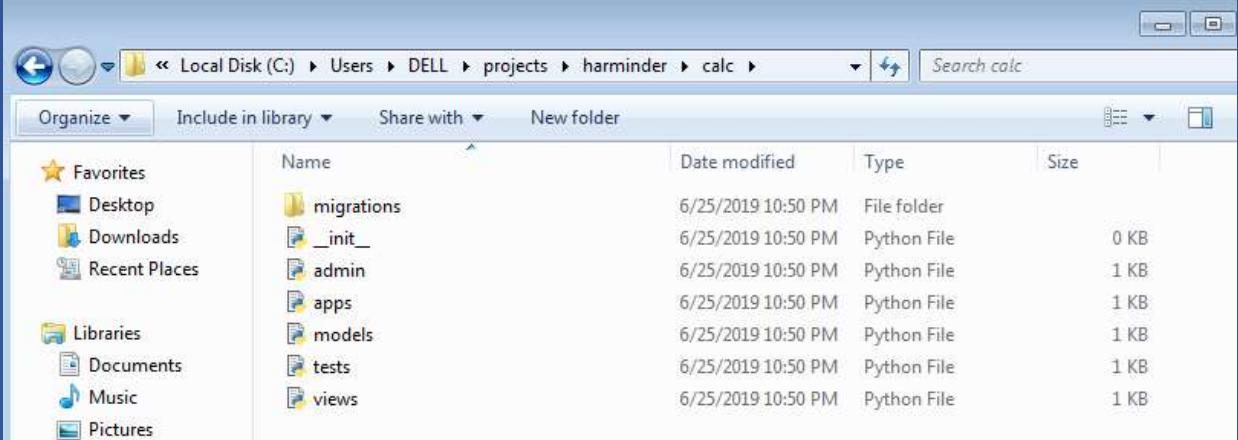
Lets check our app folder



Python Programming

Creating views

Lets check our app folder



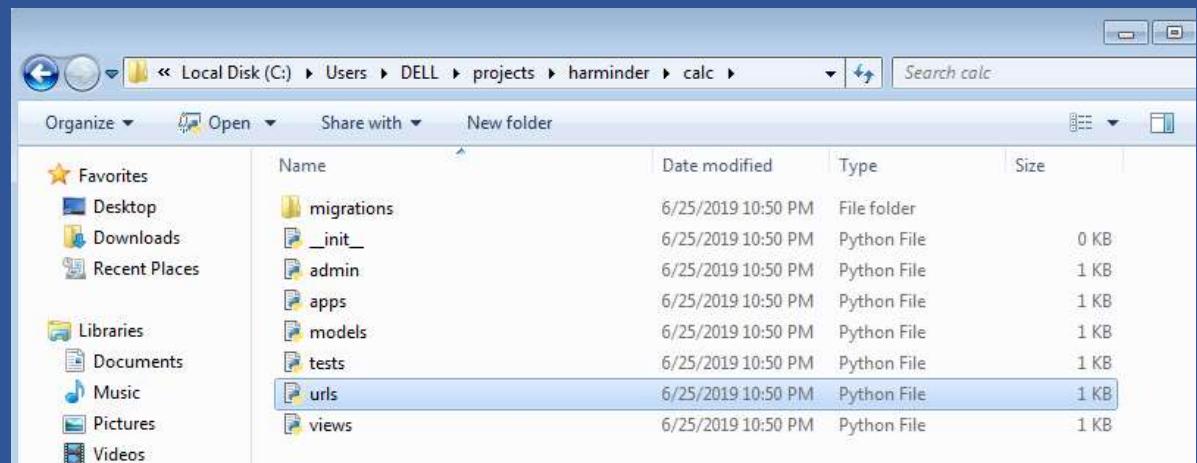
A screenshot of a Windows File Explorer window. The path in the address bar is Local Disk (C:) > Users > DELL > projects > harminder > calc. The window title is "Search calc". The left sidebar shows Favorites (Desktop, Downloads, Recent Places), Libraries (Documents, Music, Pictures), and a separator line. The main area displays a list of files and folders:

	Name	Date modified	Type	Size
	migrations	6/25/2019 10:50 PM	File folder	0 KB
	init	6/25/2019 10:50 PM	Python File	1 KB
	admin	6/25/2019 10:50 PM	Python File	1 KB
	apps	6/25/2019 10:50 PM	Python File	1 KB
	models	6/25/2019 10:50 PM	Python File	1 KB
	tests	6/25/2019 10:50 PM	Python File	1 KB
	views	6/25/2019 10:50 PM	Python File	1 KB

Python Programming

To create a url we need to create a file with name urls in our app folder

Creating views



Python Programming

Creating views

Let us add a path in urls file in our app

```
from django.urls import path  
from . import views  
  
urlpatterns = [  
    path("",views.home,name="home")  
]
```

Python Programming

Creating views

Let's create a view function

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.
def home(request):
    return HttpResponse("This is my webpage")
```

Python Programming

Creating views

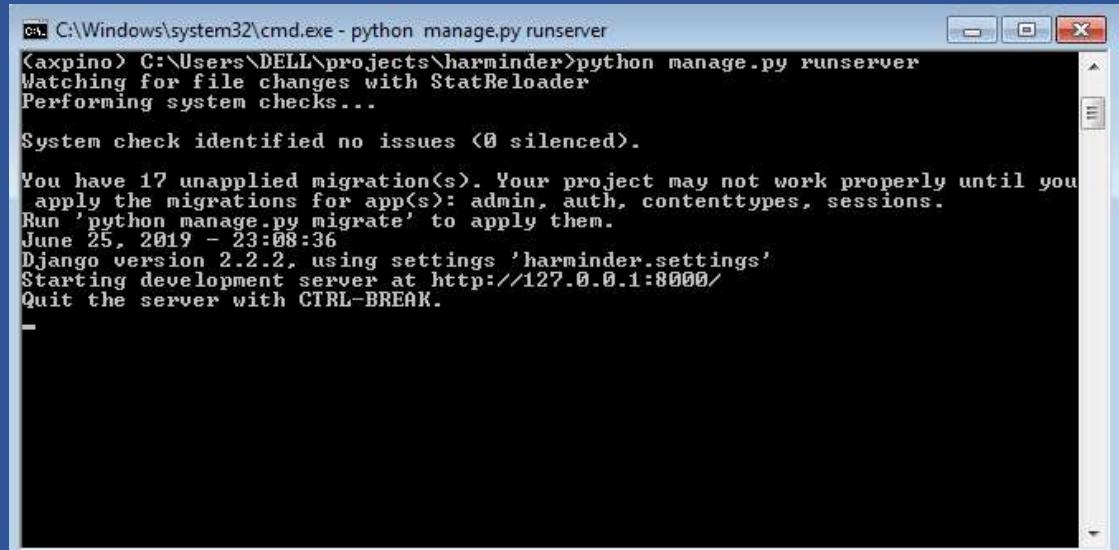
Let add path in main project url

```
from django.contrib import admin  
from django.urls import path,include  
  
urlpatterns = [  
    path("",include('calc.urls')),  
    path('admin/', admin.site.urls),  
]
```

Python Programming

Creating views

Restart the Server



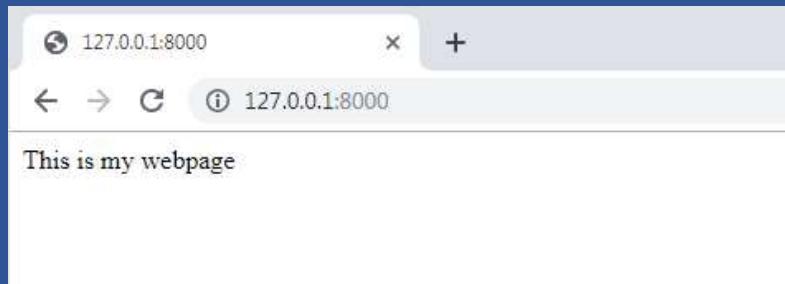
```
C:\Windows\system32\cmd.exe - python manage.py runserver
Kaxpino> C:\Users\DELL\projects\harminder>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues <0 silenced>.

You have 17 unapplied migration(s). Your project may not work properly until you
apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
June 25, 2019 - 23:08:36
Django version 2.2.2, using settings 'harminder.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Python Programming

Access the project using browser

Creating views



Python Programming