PYTHON PROGRAMING

Surender Dabur

Python Developer/Ethical Hacker

Axpino Technologies

Python Programing

INTRODUCTION TO PYTHON

Axpino Technologies

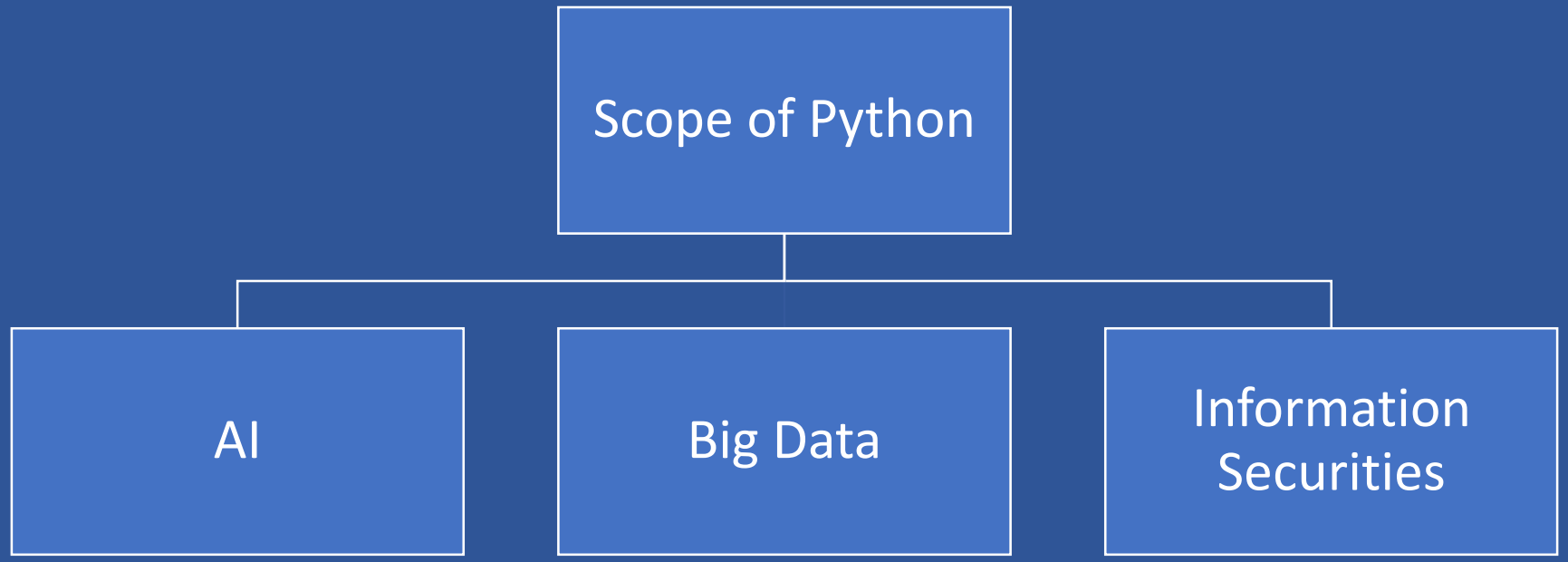
Python Programing

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

Why Python?

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


Scope of Python
AI
Big Data
Information Securities

PYTHON SETUP

Software Requirements

Download Python

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

Download Notepad++

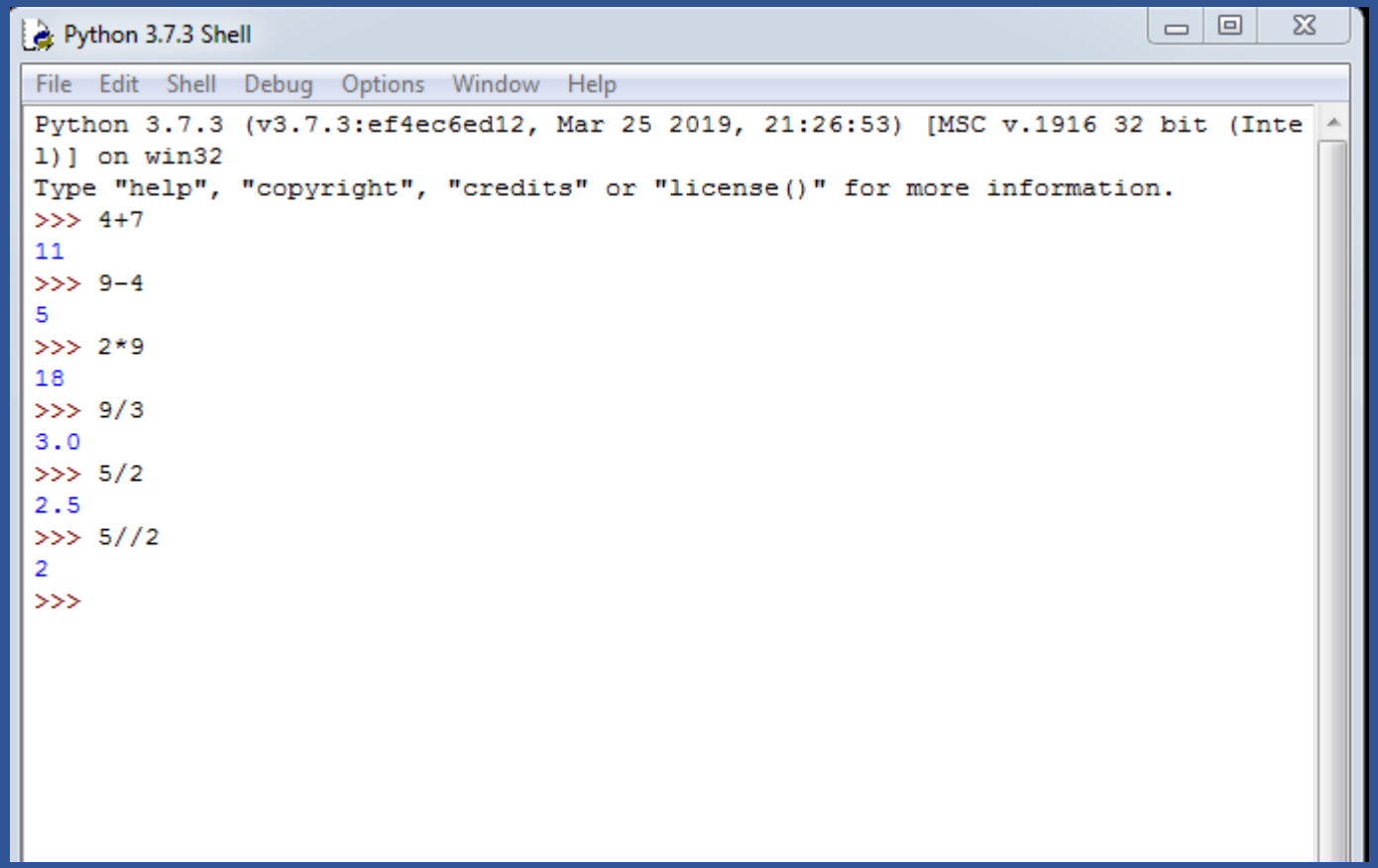
https://notepad-plus-plus.org/download/v7.6.6.html

Getting Started With Python

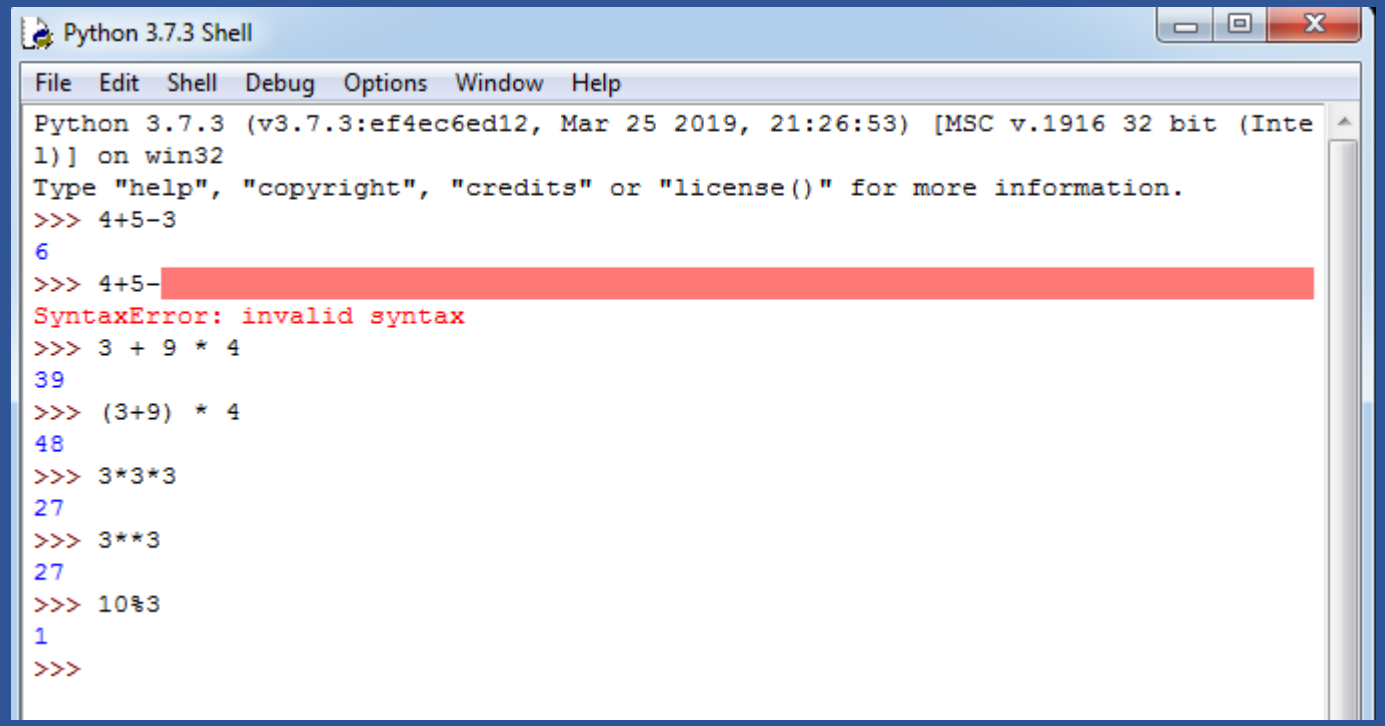
Axpino Technologies

Python Programing

Simple Basics Operations



Simple Basics Operations



Axpino Technologies

Python Programing

Simple Basics Operations

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
l)] 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
>>> |
```

Axpino Technologies

Python Programing

Variables

Variable Name
(Glass)

Value
(Water)

Variables

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
l)] 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
>>>
```

Variables

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
l)] 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
>>> |
```

Variables

-9	-8	-7	-6	-5	-4	-3	-2	-1
H	A	R	M	I	N	D	E	R
0	1	2	3	4	5	6	7	8

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
tains the files and folders that you have deleted.
its" or "license()" for more information.
>>> 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
>>>
```

LISTS

LISTS

Defining Lists

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

List Name

Elements

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
```

LISTS

Accessing Elements

names = [‘Vipul’,’Surender’,’Anup’,’Shubham’]

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

Lists can have heterogeneous values

LISTS

values = [8.2,'Surender',34]

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]
```

LISTS

Lists are Mutable

(Appending a Element)

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

LISTS

Lists are Mutable

(Inserting a Element)

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

LISTS

Lists are Mutable

(Removing a Element)

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

LISTS

Lists are Mutable

(Removing a Element using index)

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

LISTS

Lists are Mutable

(Removing a Element from Last)

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

LISTS

Lists are Mutable

(Removing multiple Elements)

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

LISTS

Lists are Mutable

(Adding multiple Elements)

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

LISTS

Lists are Mutable

(Searching Min Value in a List)

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

LISTS

Lists are Mutable

(Searching Max Value in a List)

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

LISTS

Lists are Mutable

(Calculate Sum of a List)

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

LISTS

Lists are Mutable

(Sorting a List)

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

LISTS

Lists are Mutable

(Sorting a List Descending)

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

TUPLE

TUPLE

Defining a Tuple

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

Tuple Name

Elements

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
```

TUPLE

Accessing Elements

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

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

Tuple can have heterogeneous values

TUPLE

values = (8.2,'Surender',34)

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)
```

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
>>>
```

SETS

SETS

Defining a SET

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

Set Name

Elements

SETS can have heterogeneous values

SETS

values = {8.2,'Surender',34}

SETS

Check if element exists in SET

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

SETS

Adding Element to SETS

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

Adding Multiple Element to SETS

SETS

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

SETS

Removing Element From SETS

(Gives an Error when Item is not in Set)

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

SETS

Removing Element From SETS

(No Error when Item is not in Set)

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

Removing Random Element From SETS

SETS

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

SETS

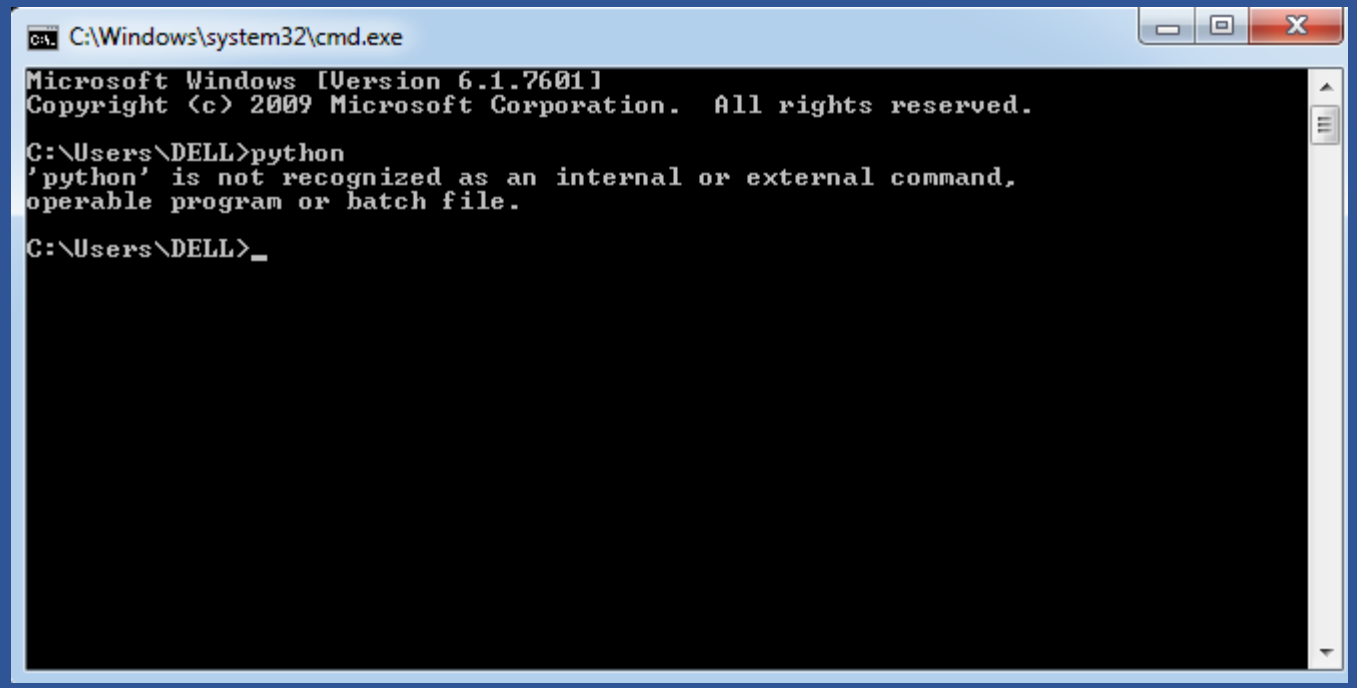
Clearing SET

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

SETTING PATH FOR WINDOWS

Checking If already Set

Setting Path for Windows



Copy Following Paths

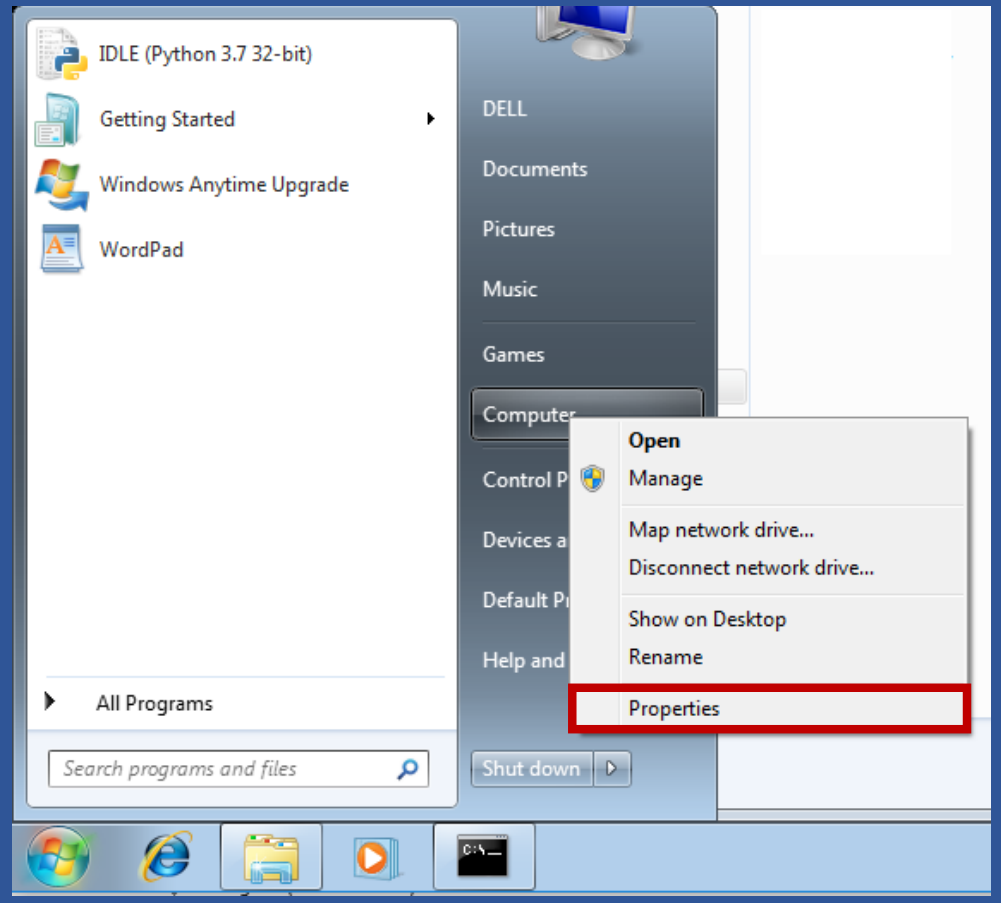
Setting Path for Windows

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

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

Setting Path for Windows

Go to Following Path

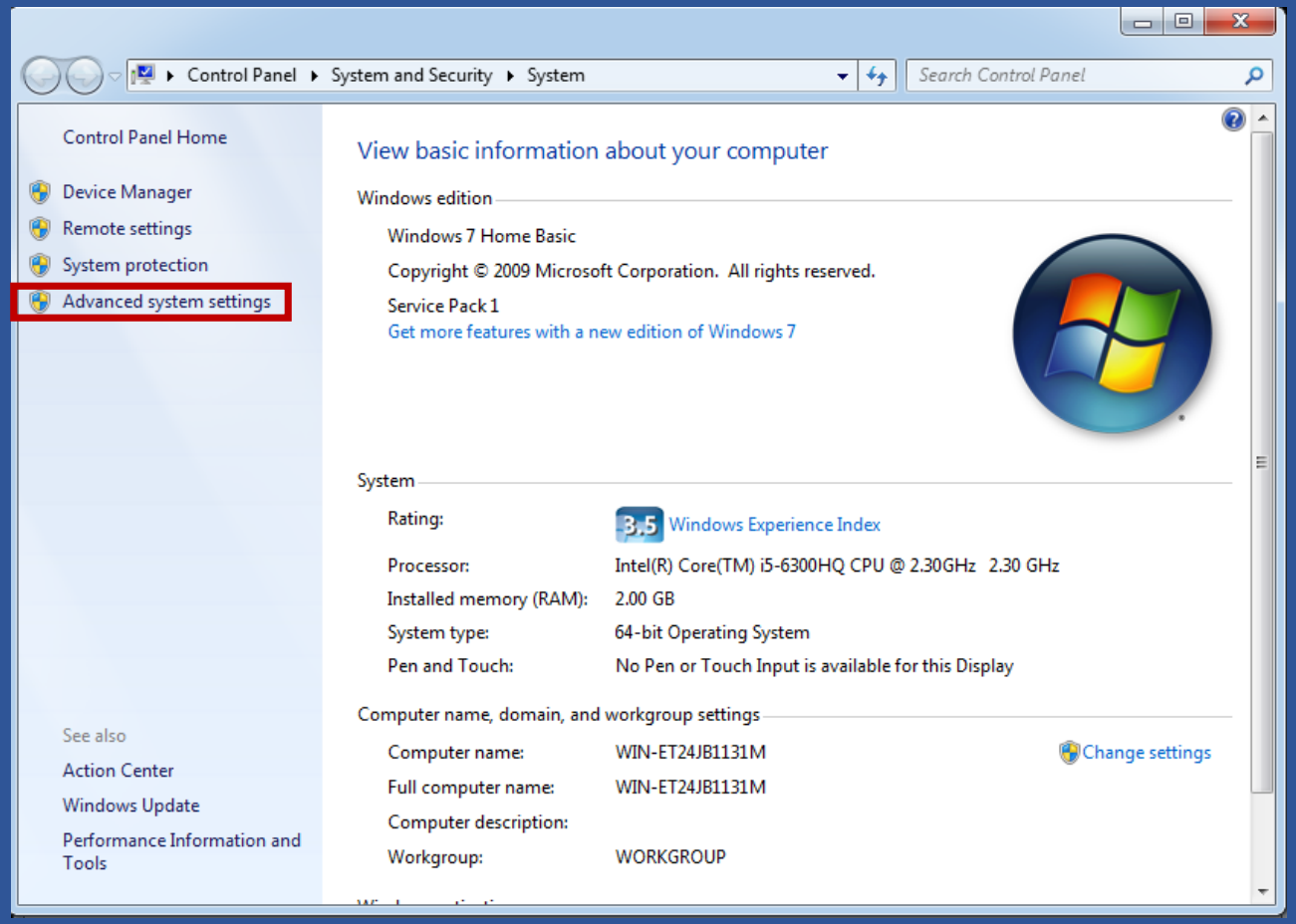


Axpino Technologies

Python Programing

Setting Path for Windows

Go to Following Path

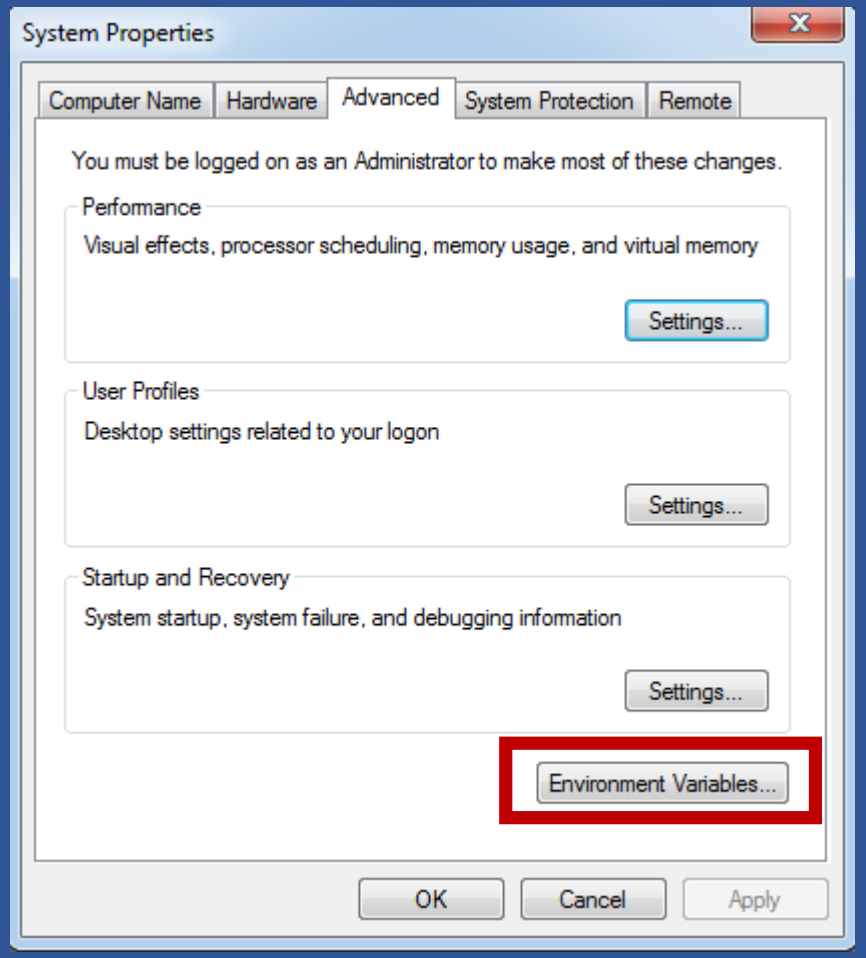


Axpino Technologies

Python Programing

Setting Path for Windows

Go to Following Path

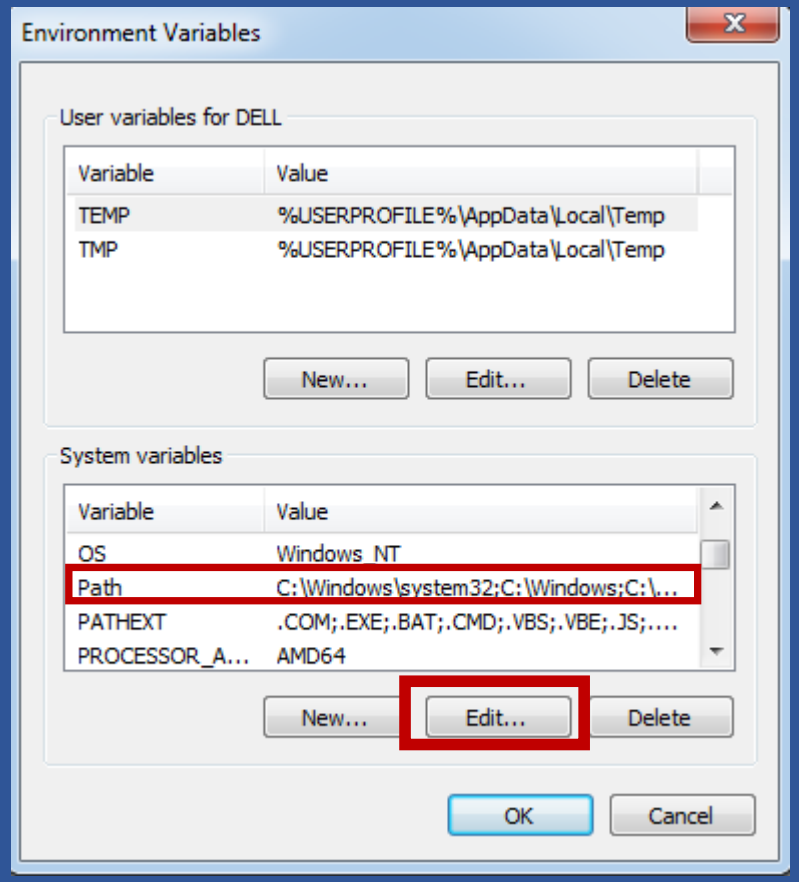


Axpino Technologies

Python Programing

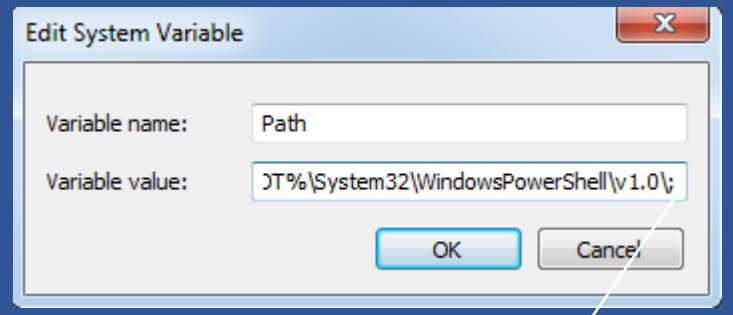
Setting Path for Windows

Go to Following Path



Setting Path for Windows

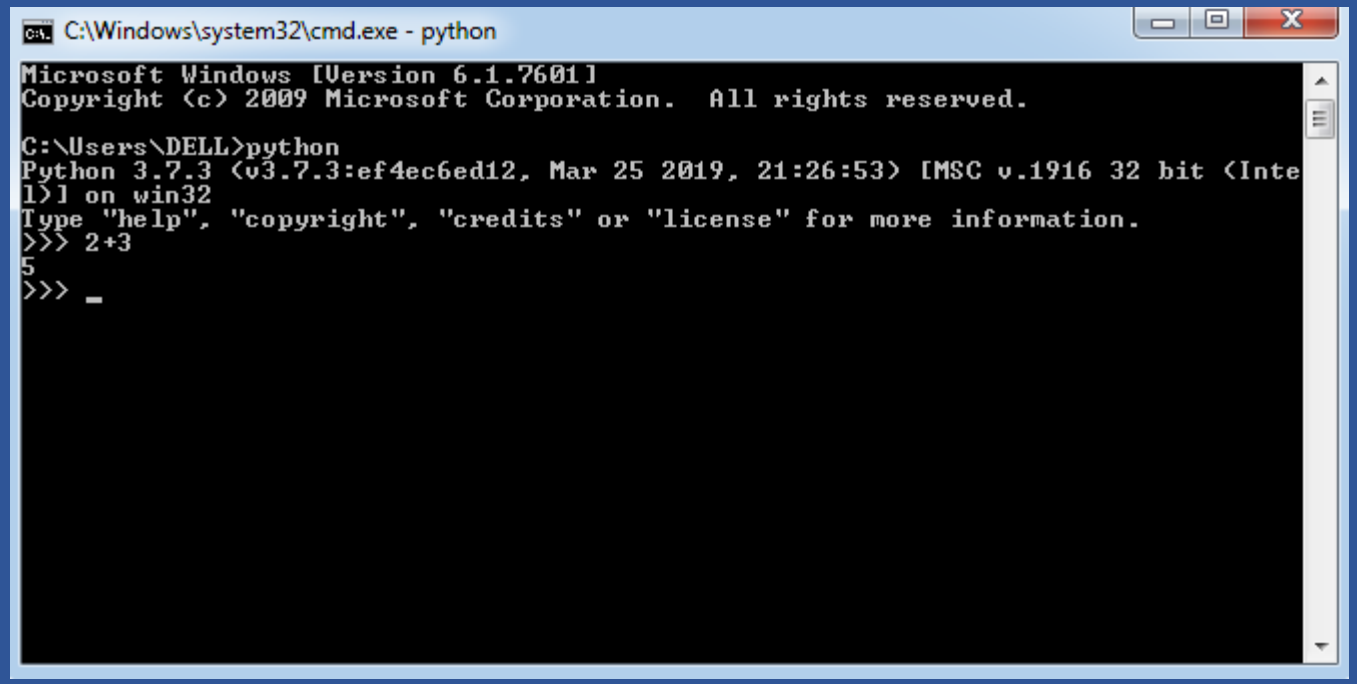
Go to Following Path



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

Verification

Setting Path for Windows



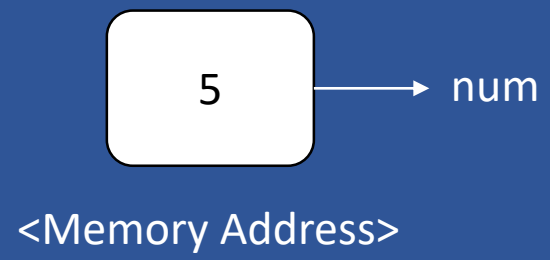
Variable Memory Concept

Variable

(Memory Concept)

Variable Storage

num=5

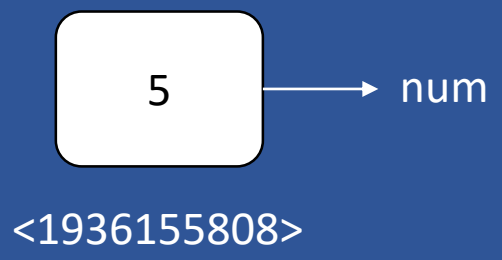


Variable

(Memory Concept)

Getting Address

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

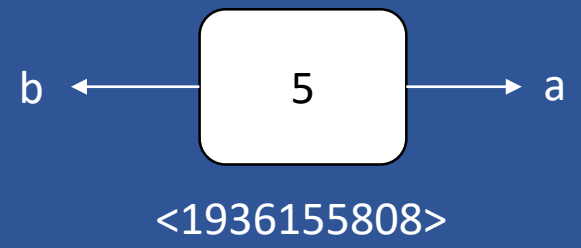


Variable

(Memory Concept)

Variables with Same value has same memory Address

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

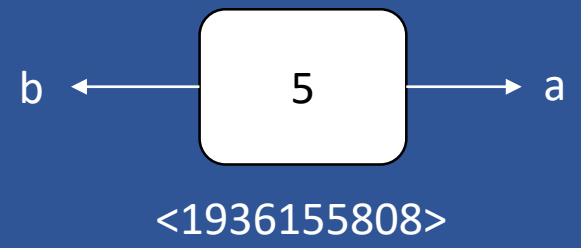


Variable

(Memory Concept)

Variables with Same value has same memory Address

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

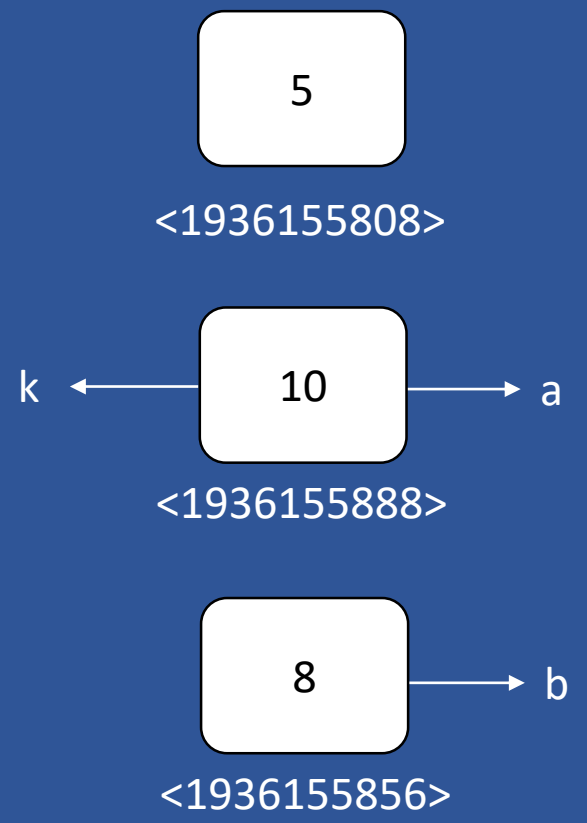


Concept of Garbage Value

Variable

(Memory Concept)

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



Variable

(Memory Concept)

Type of a Variable

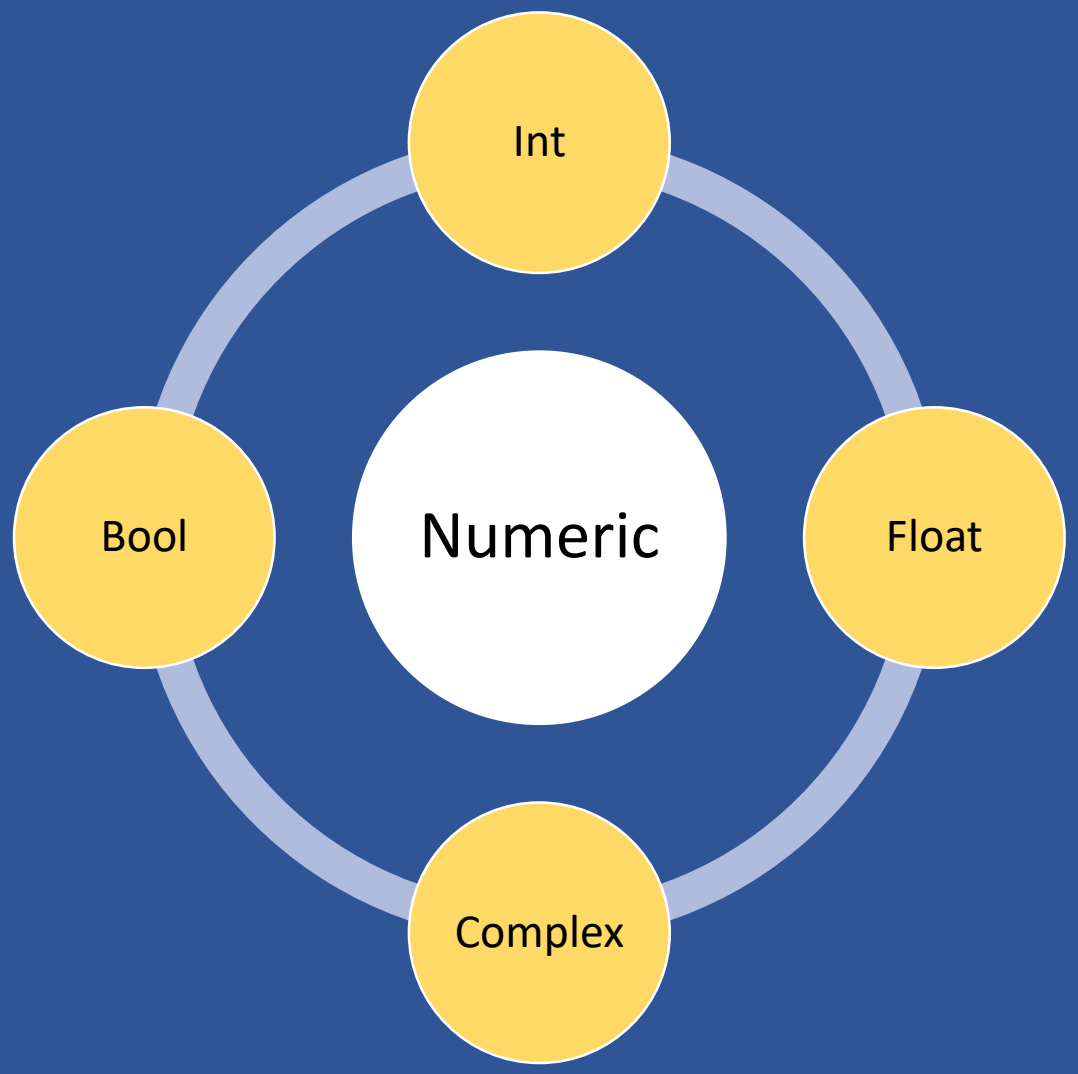
```
>>a=5
>>type(a)
<class ‘int’>
>>b=4.6
<class
‘float’>
```

Data Types

Data Types

- None
- Numeric
- Sequence
- Dictionary

Data Types



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
```

Data Types

Data Types Conversions

INT → FLOAT

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

FLOAT → INT

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

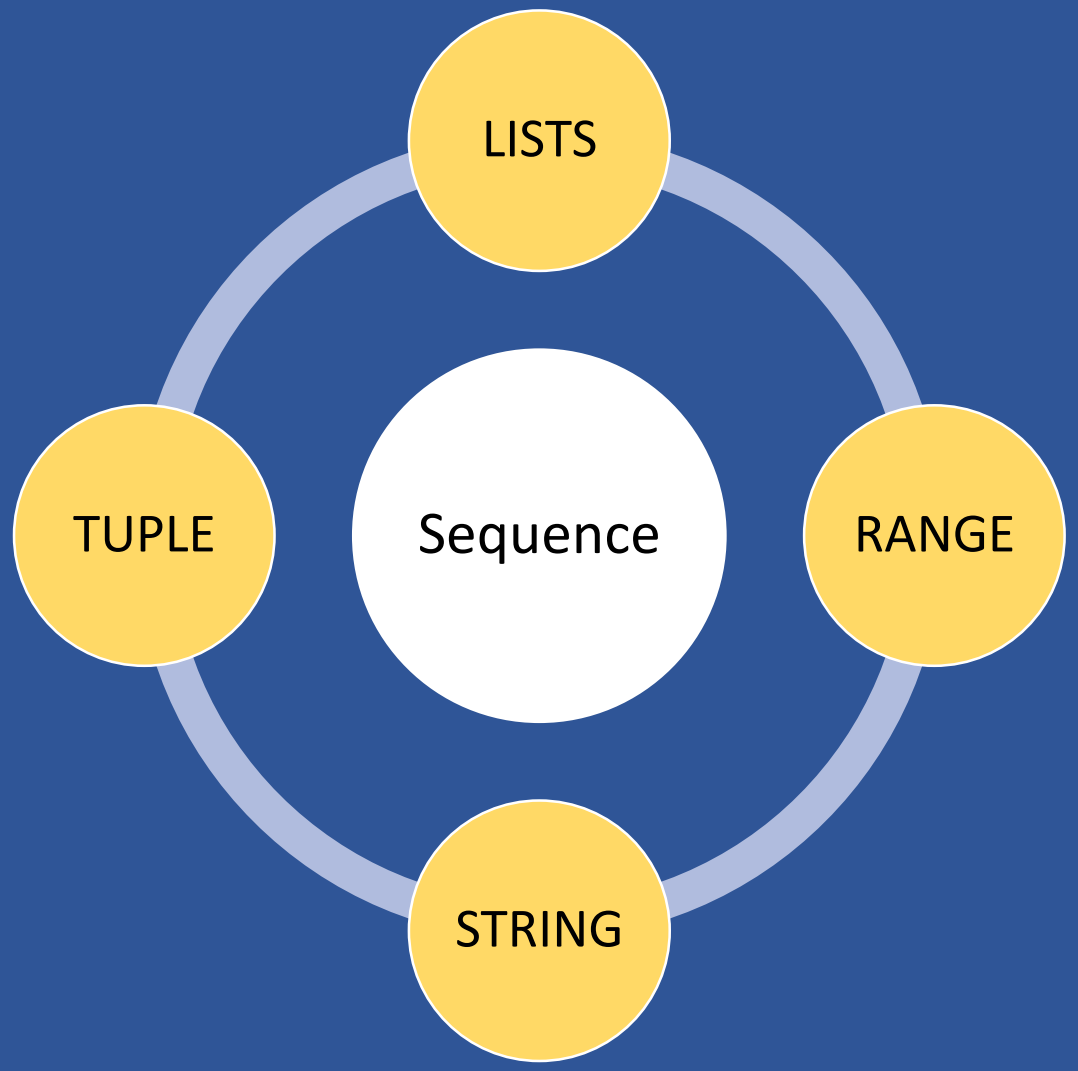
INT → COMPLEX

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

BOOL → INT

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

Data Types



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'>
```

Dictionary

Data Types

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

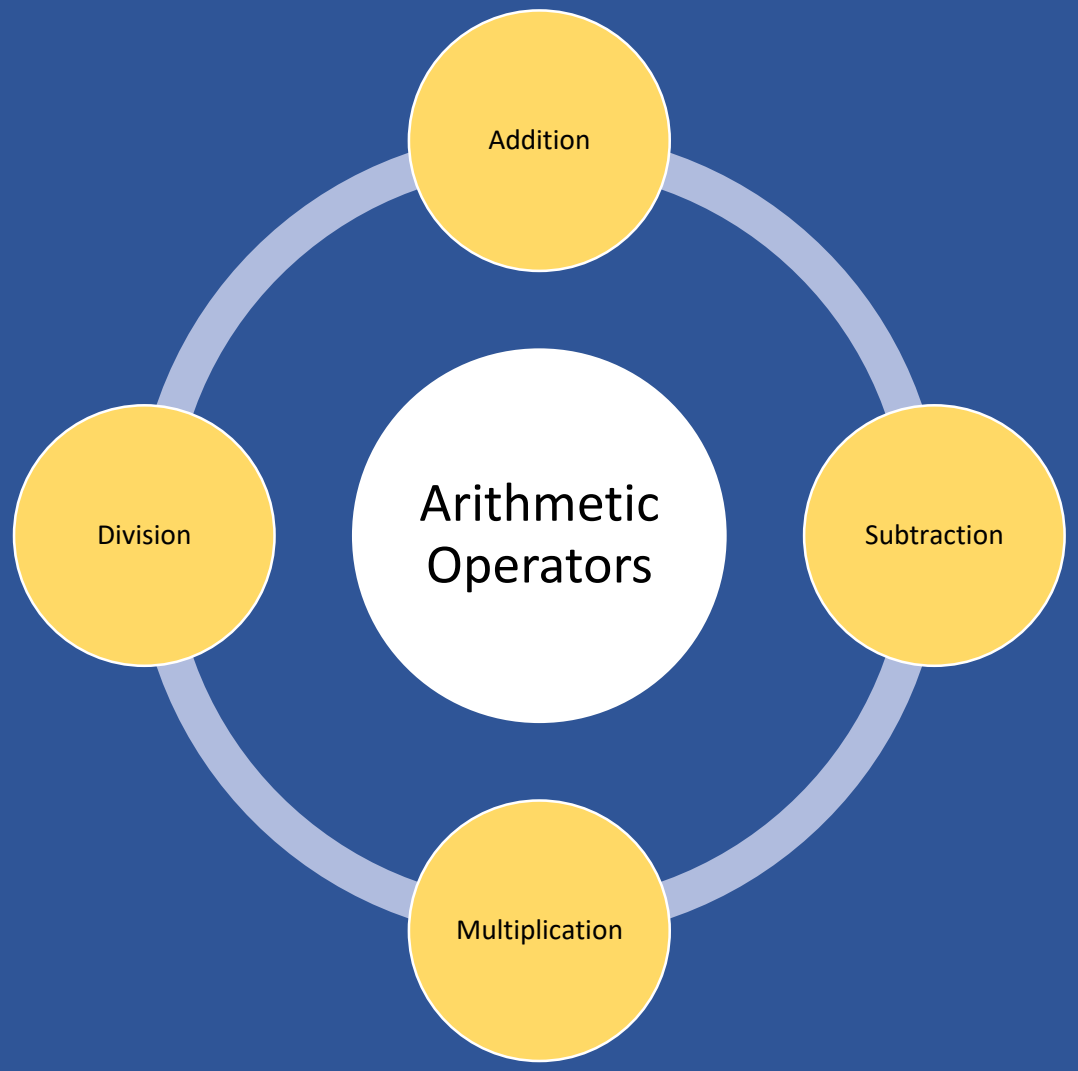
```
>>a= {‘name’:’Harminder’,’class’:’1st’}
>>a[‘class’]
‘1st’
```

OPERATORS

Operators

- Arithmetic Operators
- Assignment Operators
- Relational Operators
- Logical Operators

Operators


Addition
Division
Arithmetic Operators
Subtraction
Multiplication

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
```

Operators


Addition Assignment
Division Assignment
Assignment Operators
Subtraction Assignment
Multiplication Assignment

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
```

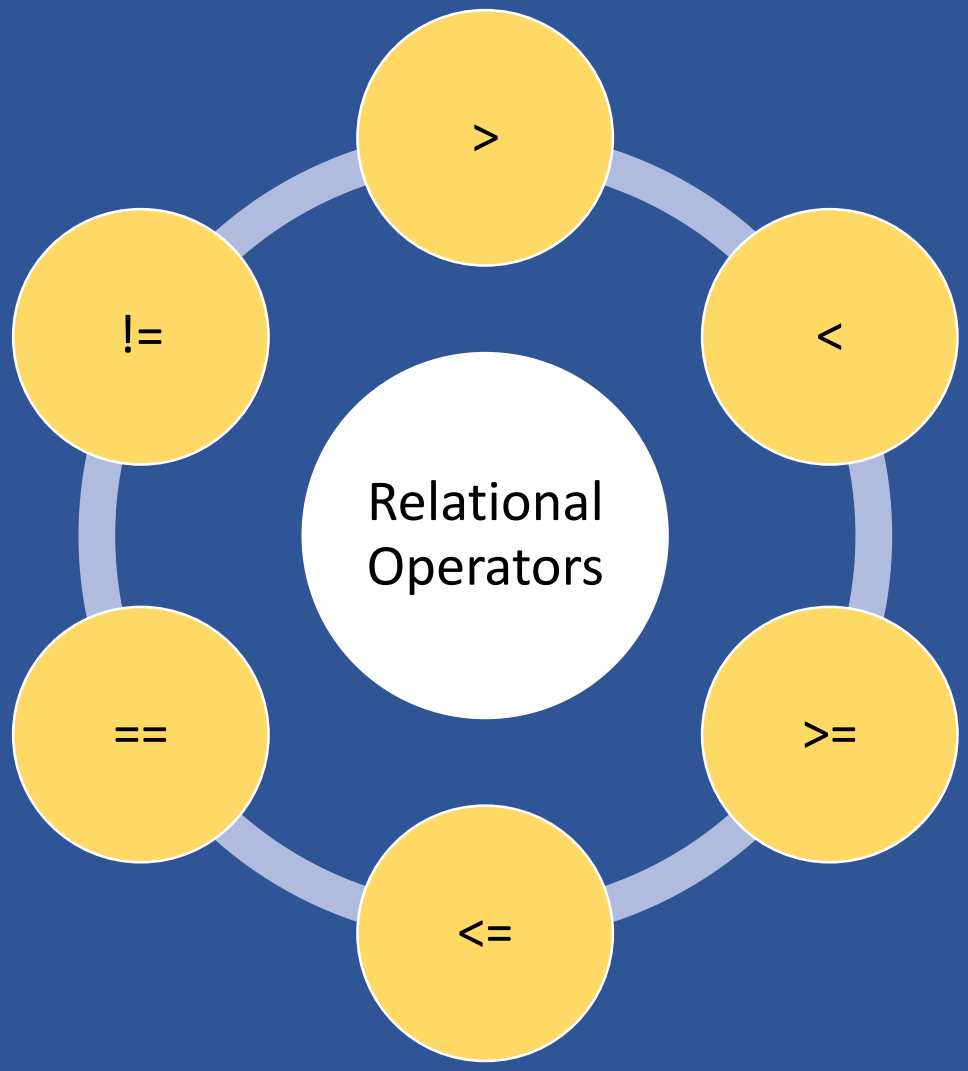
Operators

Assignment Operators

Assigning Multiple Variables at once

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

Operators



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
```

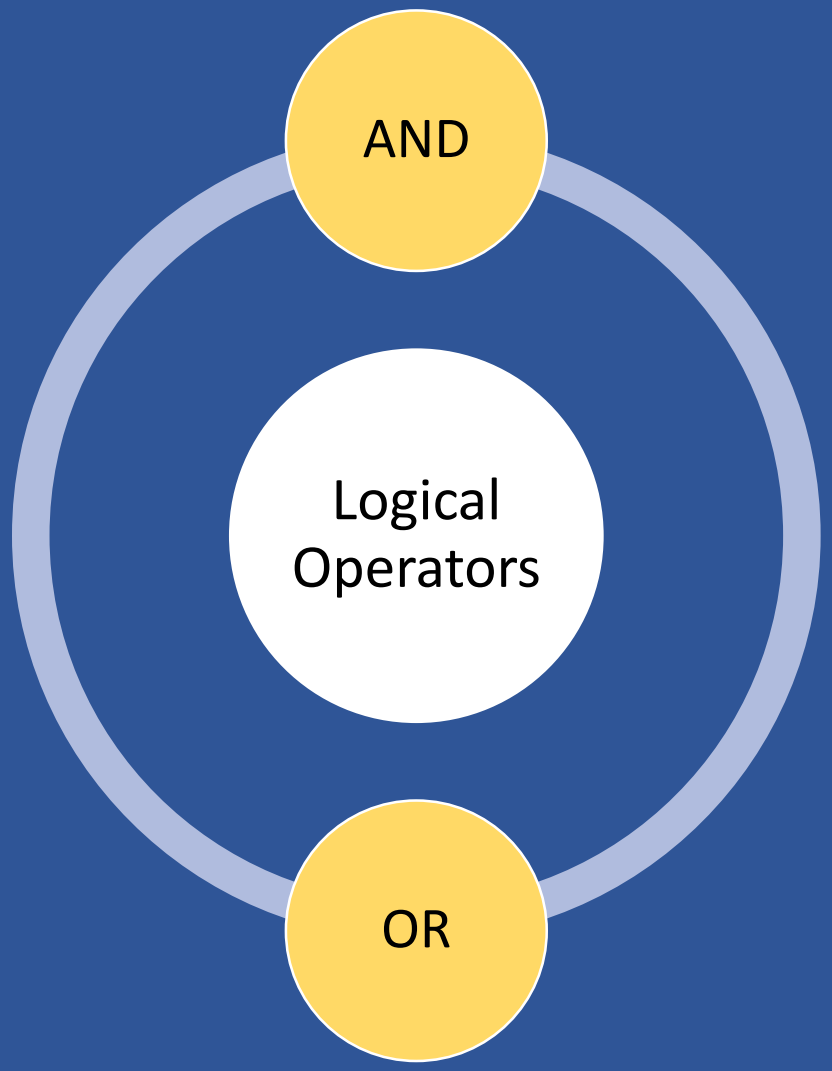
Operators

Relational Operators

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

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

Operators



Logical Operators

Operators

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

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

BITWISE OPERATOR

Bitwise Operators

- AND (&)
- OR (|)
- XOR (^)
- Left Shift (<<)
- Right Shift (>>)

Bitwise Operators

Decimal to Binary Conversion

12 → 1100

2	12	
2	6	0
2	3	0
	1	1

Bitwise Operators

Binary to Decimal Conversion

1100 → 12

1 1 0 0

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

8+4=12

Bitwise Operators

Bitwise (AND)

12 & 13 = 12

00001100 -> 12
00001101 -> 13

00001100 -> 12

Bitwise Operators

Bitwise (OR)

12 | 13 = 13

00001100 -> 12
00001101 -> 13

00001101 -> 13

Bitwise Operators

Bitwise (XOR)

12 ^ 13 = 1

00001100 -> 12
00001101 -> 13

00000001 -> 1

Bitwise Operators

Left Shift (<<)

$$10 << 2 = 40$$

00001010.000 -> 10
0000101000.0 -> 40

Bitwise Operators

Right Shift (>>)

10 >> 2 = 2

00001010.000 -> 10
00001010.10000 -> 2

Math Module

Axpino Technologies

Python Programing

Math Module

```
>>Import math
```

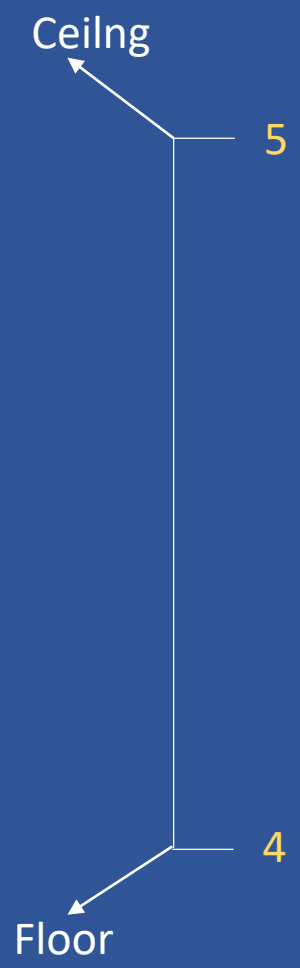
Finding Square Root

Math Module

```
>>Import math
>>x=math.sqrt(25)
>>x
5
```

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
```

Math Module

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

Importing Specific functions of Math Module

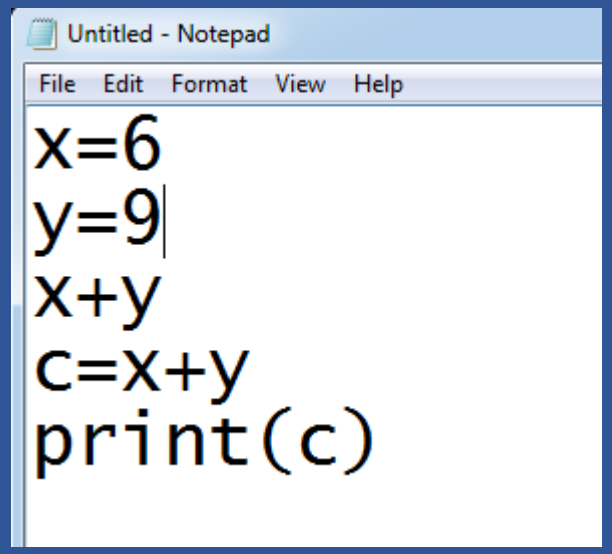
Math Module

```
>>from math import sqrt
```

Creating & Running Python Files

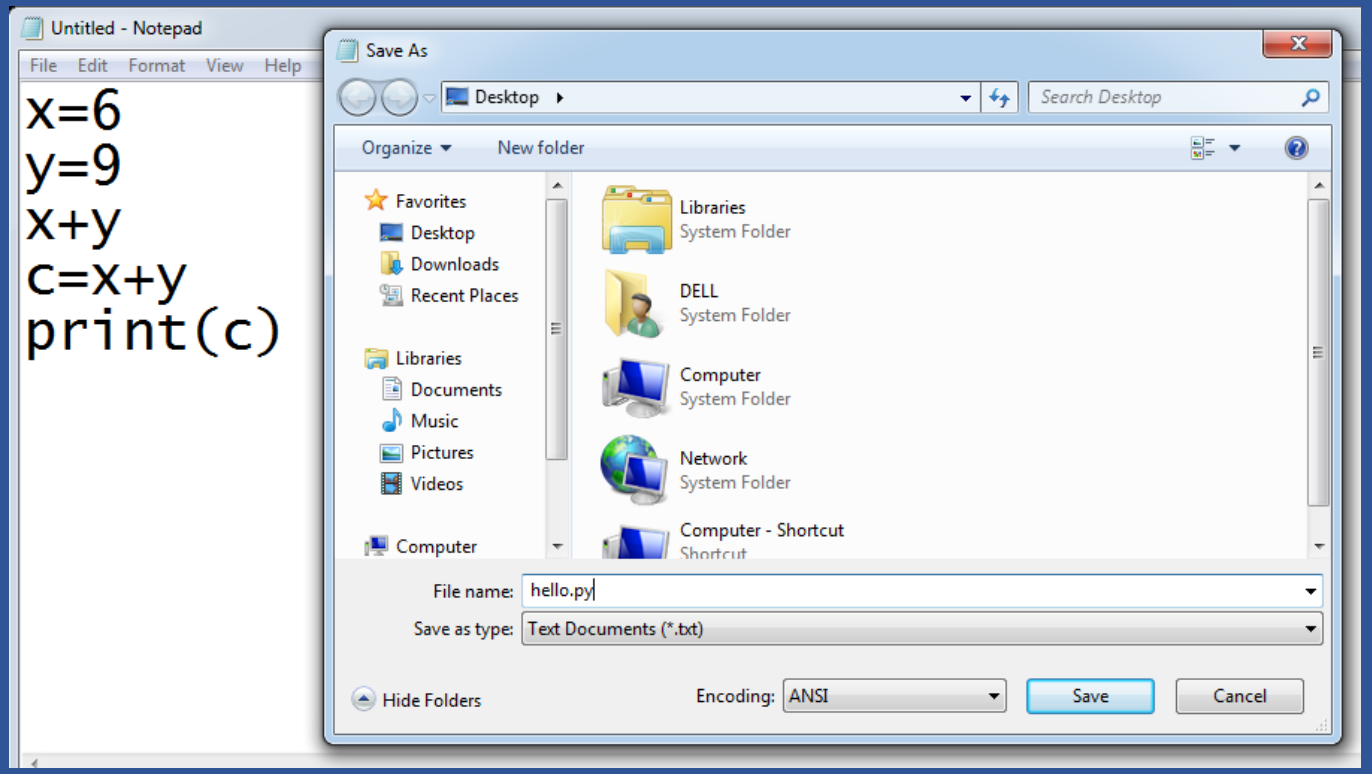
Write a Program on Notepad/IDE

Creating & Running Py Files



Save file with PY Extension

Creating & Running Py Files

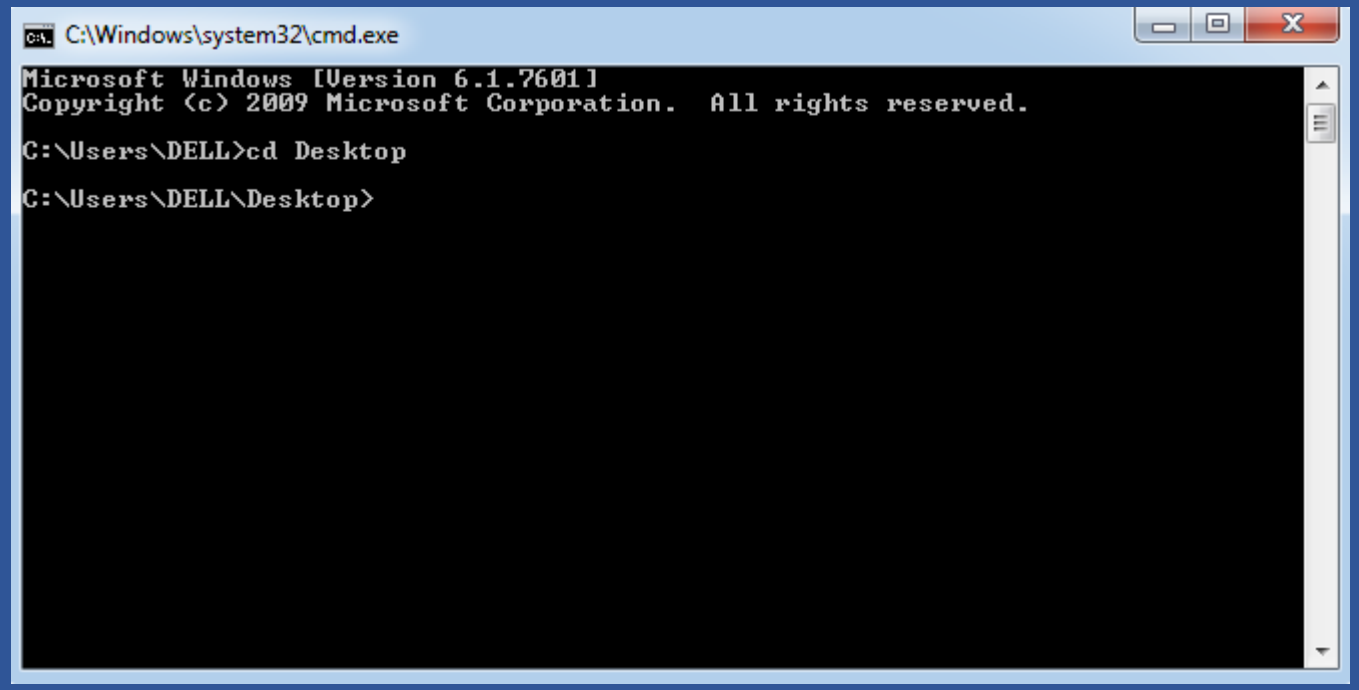


Axpino Technologies

Python Programing

Creating & Running Py Files

Open CMD and Change path to file's location

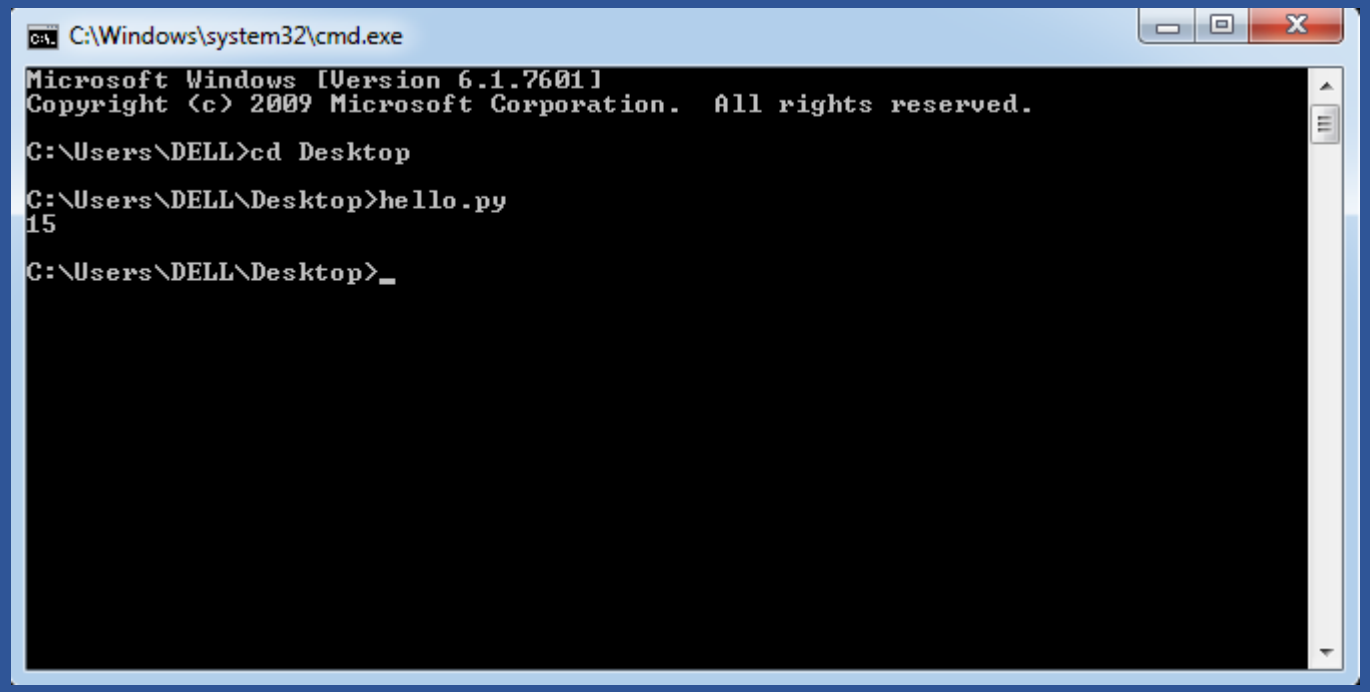


Axpino Technologies

Python Programing

Call the python file

Creating & Running Py Files



User Input

Input Function

User Input

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

Input Function only accept strings

User Input

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

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

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
```

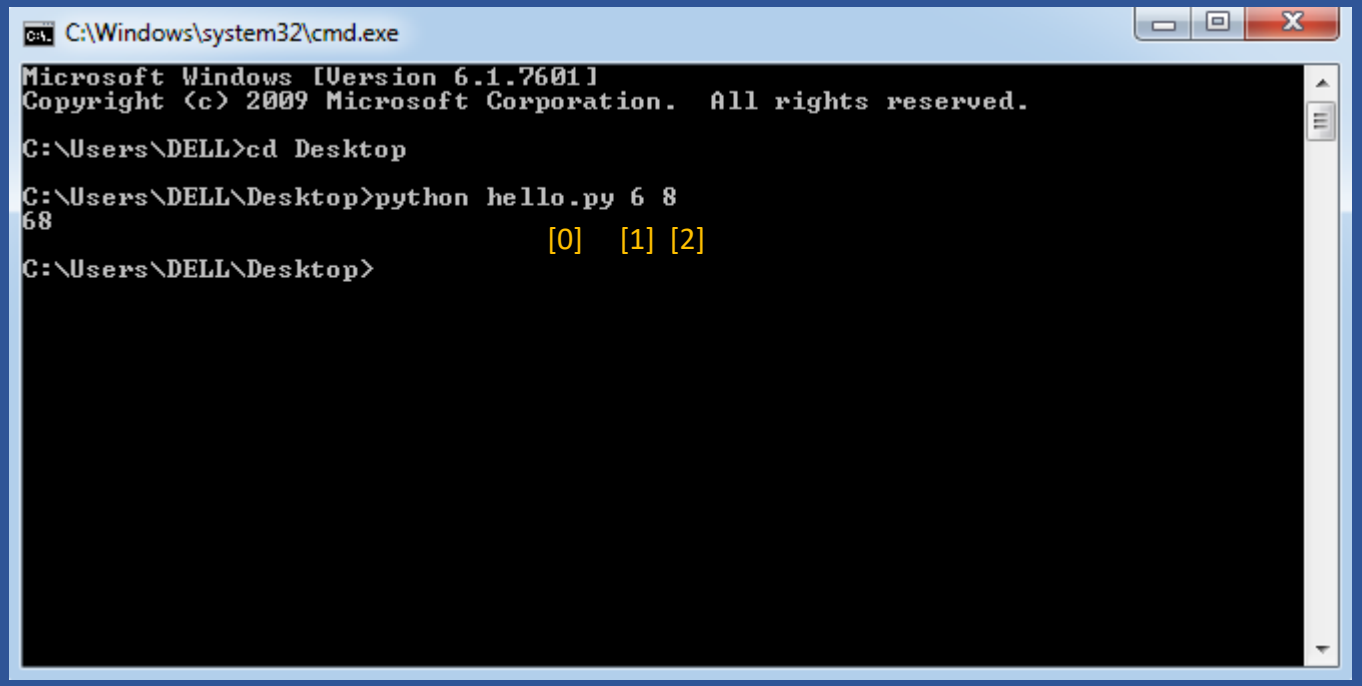
Passing Argument Input in CMD

User Input

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

Passing Argument Input in CMD

User Input



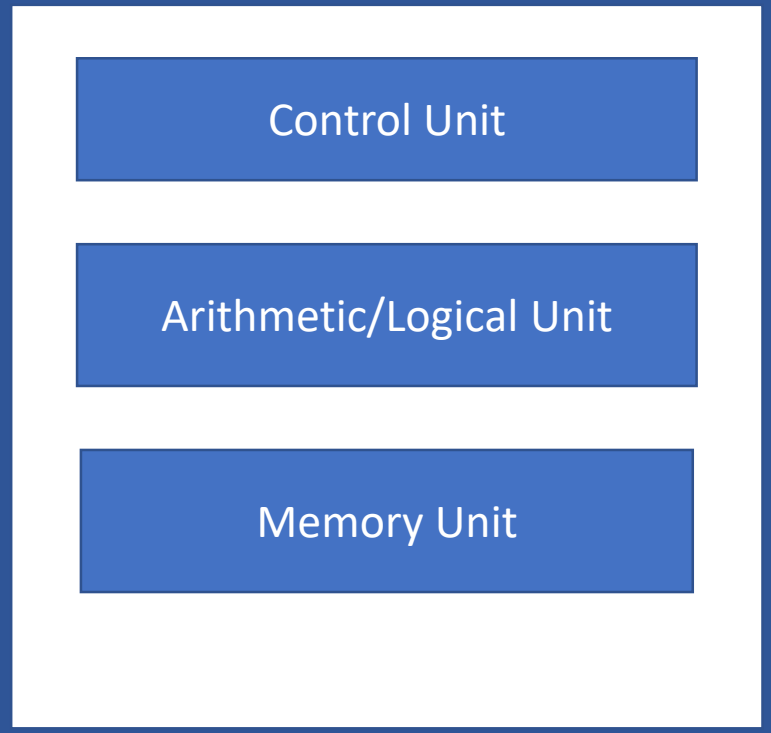
Control Flow Statements

Axpino Technologies

Python Programing

Control Flow Statements

Central Processing Unit



Control Flow Statements

IF Statement

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

Suite

IF Statement Needs Indentation

Control Flow Statements

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

Else Statement

Control Flow Statements

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

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")
```

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")
```

Loops

Loops

While Loop

```
x=0
while x<=5:
    print(x)
    x=x+1
```

Initialization

Condition

Increment

Loops

While Loop (Reverse)

```
x=5
while x>=0:
    print(x)
    x=x-1
```

Initialization

Condition

Increment

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()
```

Loops

For Loop with List

```
a = [“Harminder”,1,”Surender”]

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

Loops

For Loop with String

```
a = “Harminder”

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

Loops

For Loop with Tuple

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

Loops

For Loop with Sets

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

Loops

For Loop with Range

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

Loops

For Loop with Range

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

Loops

For Loop with Range

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

Loops

Nested For Loop

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

Loops

Break Statement

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

Loops

Continue Statement

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

Loops

Pass Statement

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