

Notes Python

Bases

```
#!/usr/bin/python3

# output
print ("Hello, Python!")
print(x, end=" ") # Appends a space instead of a newline in
Python 3

# input
x = input("something:")

# raise exception
raise IOError("file error") #this is the recommended syntax
in Python 3
```

```
# Quotation in Python
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Variable Types

```
# Simple Assignment
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
value   = 3e+26J       # A complex
name    = "John"      # A string

# Multiple Assignment
a = b = c = 1
a, b, c = 1, 2, "john"
```

```
# deletion
a, b, c = 1, 2, "john"
```

```
str = 'Hello World!'

print (str)           # Prints complete string
print (str[0])        # Prints first character of the string
print (str[2:5])      # Prints characters starting from 3rd
to 5th
print (str[2:])       # Prints string starting from 3rd
character
print (str * 2)       # Prints string two times
print (str + "TEST") # Prints concatenated string
```

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylis = [123, 'john']

print (list)          # Prints complete list
print (list[0])       # Prints first element of the list
print (list[1:3])     # Prints elements starting from 2nd
till 3rd
print (list[2:])      # Prints elements starting from 3rd
element
print (tinylis * 2)   # Prints list two times
print (list + tinylis) # Prints concatenated lists
```

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
tinytuple = (123, 'john')

print (tuple)         # Prints complete tuple
print (tuple[0])      # Prints first element of the tuple
print (tuple[1:3])    # Prints elements starting from 2nd
till 3rd
print (tuple[2:])     # Prints elements starting from 3rd
element
print (tinytuple * 2) # Prints tuple two times
print (tuple + tinytuple) # Prints concatenated tuple
```

Classes

```
#!/usr/bin/python
```

```
class Employee:
    'Common base class for all employees'
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print "Total Employee %d" % Employee.empCount

    def displayEmployee(self):
        print "Name : ", self.name, ", Salary: ", self.salary

"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print "Total Employee %d" % Employee.empCount

// Built-In Class Attributes

print "Employee.__doc__:", Employee.__doc__
print "Employee.__name__:", Employee.__name__
print "Employee.__module__:", Employee.__module__
print "Employee.__bases__:", Employee.__bases__
print "Employee.__dict__:", Employee.__dict__

// Destroying Objects (Garbage Collection)
del emp1

// Class Inheritance
#!/usr/bin/python
```

```

class Parent:          # define parent class
    parentAttr = 100
    def __init__(self):
        print "Calling parent constructor"

    def parentMethod(self):
        print 'Calling parent method'

    def setAttr(self, attr):
        Parent.parentAttr = attr

    def getAttr(self):
        print "Parent attribute :", Parent.parentAttr

class Child(Parent): # define child class
    def __init__(self):
        print "Calling child constructor"

    def childMethod(self):
        print 'Calling child method'

c = Child()           # instance of child
c.childMethod()       # child calls its method
c.parentMethod()      # calls parent's method
c.setAttr(200)        # again call parent's method
c.getAttr()           # again call parent's method


// Overriding Methods
#!/usr/bin/python

class Parent:          # define parent class
    def myMethod(self):
        print 'Calling parent method'

class Child(Parent): # define child class
    def myMethod(self):
        print 'Calling child method'

c = Child()           # instance of child
c.myMethod()          # child calls overridden method

```