

## Lesson-1: A Quick Refresher & General Knowledge

**Let us discuss the basics first!**

**What are the basic components of a computer?**

- CPU
- Primary Memory (RAM)
- Secondary Memory (HDD)
- Input Devices (Keyboard, Mouse, etc.)
- Output Devices (Monitor, Printer etc.)
- Communication Devices (NIC, Modem etc.)

**Give few examples of programming languages.**

- C, C++, Python, Java, C# etc.

**Give few examples of operating systems.**

- Apple's Mac OS, GNU/Linux, Microsoft Windows

**How long will it take to become a professional programmer?**

- Good Conceptual Knowledge
- Good At Logical Thinking
- Good At Debugging

**How long will it take to become a professional programmer?**

- Amount of practice, Quality of practice

**What is System Software and Application Software?**

- OS, Compilers, Interpreters (System Software)
- Amazon.com, MakeMyTrip.com (Application Software)

**Which is the best programming language?**

**What are the different domains in the IT industry?**

- Embedded, Telecom, Healthcare, E-Commerce, Finance, Banking, Automotive etc.

**How many programming languages to learn?**

**Exercise:**

- Watch the basics of digital electronics video completely.

## Lesson-2: Introducing Python Programming Language

**IMPORTANT NOTE:** Convince your mind, you may not be able to understand everything today – Don't Panic. Just move on.

### Credits & Appreciation:

- Guido Van Rossum
- In 1991

### What is the first step to learn Python?

OK! What is the first step to cook your favorite Hyderabadi Vegetable Biryani?

### How to install Python onto your computer?

<https://subhashprogrammingclasses.in/how-to-install-python-on-windows-linux-macintosh/>

<https://subhashprogrammingclasses.in/how-to-get-started-with-python-idle/>

### What is the difference between C, Java and Python?

#### 1. In Java:

```
public class Add
{
    public static void main( String [] args )
    {
        int a, b;
        a = b = 10;
        System.out.println("Sum = " + (a+b));
    }
}
```

#### 2. In C:

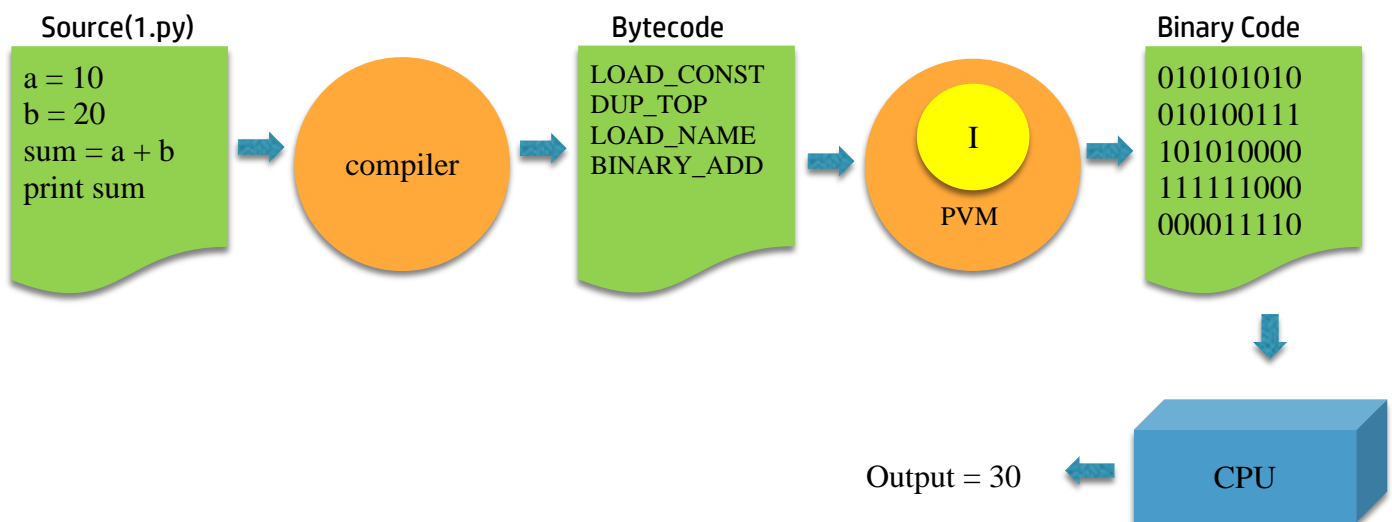
```
int main()
{
    int a, b;
    a = b = 10;
    printf("Sum = %d\n", (a+b));
}
```

#### 3. In Python:

```
a = b = 10
print( "Sum = ", (a+b))
```

**What are the main features of Python?**

- Simple
- Easy to learn
- Open Source
- High Level Language
- Dynamically Typed
- Platform Independent
- Portable
- Procedure Oriented & Object Oriented
- Huge Set Of Libraries (Batteries Included)
- Database connectivity

**How does Python program execute?****Try this:**

- `py -3 -m py_compile myfirstprogram.py` (Generates Bytecode)
- `cd __py__cache__`
- `py -3 myfirstprogram.cpython-39.pyc`

**OR**

- `py -3 myfirstprogram.py` (Gives you the output)

**Flavors Of Python:**

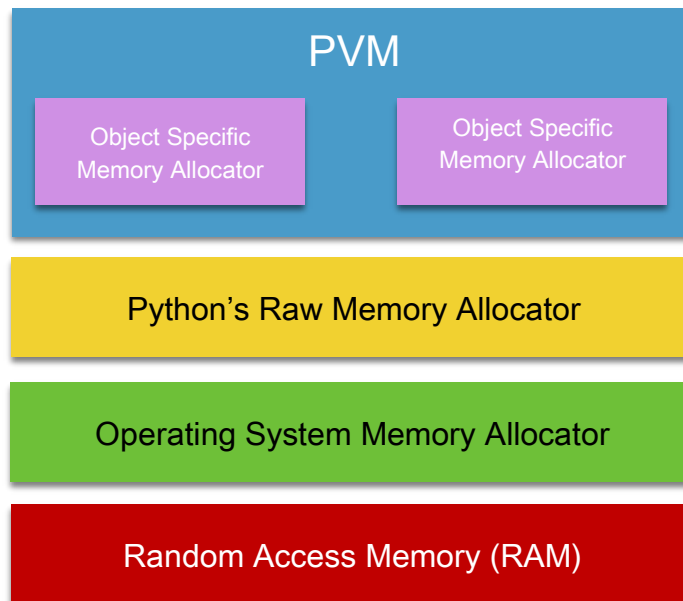
Cpython ([www.python.org](http://www.python.org))

Jython ([www.jython.org](http://www.jython.org))

IronPython ([www.ironpython.net](http://www.ironpython.net))

RubyPython ([www.rubygems.org/gems/rubypython/versions/0.6.3](http://www.rubygems.org/gems/rubypython/versions/0.6.3))

**Memory Management In Python:**



### Lesson-3: Basic I/O, Datatypes, Literals and Identifiers

- Literals are nothing but constants that can be stored in memory. All literals can be grouped into various categories called data types.
- In Python, data types are decided dynamically by the interpreter. This is called dynamic typing.
- In Python, anything and everything is considered as an object.
- In Python, unlike other programming languages, variable names are kind of tag attached to the objects stored on the heap.

#### **3 Types of Literals**

- Numeric Literals
  - Integer Literal
  - Float Literal
  - Hexadecimal Literal
  - Octal Literal
  - Binary Literal
  - Complex Literal
- String Literals
- Boolean Literals
  - True
  - False

#### **Built-in Data Types**

- None
- Numeric
  - int
  - float
  - complex
- bool
- Sequences (Ordered)
  - str
  - bytes
  - bytearray
  - list
  - tuple
  - range
- Set
  - set
  - frozenset
- Map
  - dict

**Program 1:****Output:**

```
#This is my first python program
```

```
"""This is my first python  
program"""
```

```
"""This is my first  
python program"""
```

```
a = 5  
b = +6
```

```
c = 1.0  
d = +2.0
```

```
e = 5 + 2j  
f = 2 + 3j
```

```
g = 0x1a  
h = 0x23
```

```
i = 0b1010  
j = 0b0001
```

```
k = 0o45  
l = 0o46
```

```
m = True;
```

```
n = 'Hello World\n'  
o = "Hello World\n"  
p = """Hello World\n"  
q = """Hello World\n"""
```

```
sum_one = a + b  
sum_two = c + d  
sum_three = e + f  
sum_four = g + h  
sum_five = i + j  
sum_six = k + l
```

```
print("The sum of ", a, "and ", b, "is ", sum_one )  
print("The sum of ", c, "and ", d, "is ", sum_two )  
print("The sum of ", e, "and ", f, "is ", sum_three )  
print("The sum of ", g, "and ", h, "is ", sum_four )  
print("The sum of ", i, "and ", j, "is ", sum_five )  
print("The sum of ", k, "and ", l, "is ", sum_six )  
print(m)  
print(n, o, p, q)
```

```
The sum of 5 and 6 is 11  
The sum of 1.0 and 2.0 is 3.0  
The sum of (5+2j) and (2+3j) is (7+5j)  
The sum of 26 and 35 is 61  
The sum of 10 and 1 is 11  
The sum of 37 and 38 is 75  
True  
Hello World  
Hello World  
Hello World  
Hello World
```

**User-defined Data Types**

- These are those data types created by programmers

**Program 2:**

#Taking input through keyboard

```
a = input("Enter first number\n")
b = input("Enter second number\n")
```

```
c = input("Enter third number\n")
d = input("Enter fourth number\n")
```

```
sum_one = a + b
sum_two = c + d
```

```
print("The sum of ", a, "and ", b, "is ", sum_one )
print("The sum of ", c, "and ", d, "is ", sum_two )
```

**Output:**

```
Enter first number
23
Enter second number
45
Enter third number
56
Enter fourth number
67
The sum of 23 and 45 is 2345
The sum of 56 and 67 is 5667
```

**Program 3:**

#Taking input through keyboard

```
a = input("Enter first number\n")
b = input("Enter second number\n")
```

```
a = int(a)
b = int(b)
```

```
c = input("Enter third number\n")
d = input("Enter fourth number\n")
```

```
c = float(c)
d = float(d)
```

```
complex_number_one = complex(input("Enter first complex number"));
complex_number_two = complex(input("Enter second complex number"));
```

```
sum_one = a + b
sum_two = c + d
sum_three = complex_number_one + complex_number_two;
```

```
print("The sum of ", a, "and ", b, "is ", sum_one )
print("The sum of ", c, "and ", d, "is ", sum_two )
print("The sum of ", complex_number_one, "and ", complex_number_two, "is ", sum_three )
```

**Output:**

```
Enter first number
1
Enter second number
2
Enter third number
3
Enter fourth number
4
Enter first complex number
5+6j
Enter second complex number
4+5j
The sum of 1 and 2 is 3
The sum of 3.0 and 4.0 is 7.0
The sum of (5+6j) and (4+5j) is
(9+11j)
```

**Guess The Output:**

```
>>> 10          >>> -10          >>> .10          >>> 1,0          >>> 1,024
```

```
>>> 1, 0x20      >>> 1, 0o45      >>> 1, 0b1010
```

```
>>> print("hello")          >>> print("hello", "hello")
```

```
>>> print("hello", "hello", sep='$')
```

```
>>> print("hello", "hello", sep='$', end='#')
```

```
>>> x = 15.6          >>> x = 15          >>> x = 0o17
>>> int(x)            >>> float(15)        >>> int(x)
```

```
>>> x = 0B1110010     >>> x = 0x1c2        >>> x = "17"
>>> int(x)            >>> int(x)          >>> int(x, 8)
```

**NOTE:** `int(x,8)` means convert the value of x which contains a value in octal format to integer.

```
>>> x = "1110010"     >>> x = "1c2"          >>> a = 10
>>> int(x,2)          >>> int(x,16)       >>> bin(a)
```

```
>>> b = 1010          >>> b = 10            >>> b = 10
>>> oct(b)            >>> hex(10)         >>> type(b)
```

**A BIG WARNING ABOUT FLOATING POINT REPRESENTATION: Range and Precision**

**Limits for floating Point Range** = (10 to the power -308) to (10 to the power +308)

```
>>> 1.5e200 * 2.0e210 => inf (Arithmetic Overflow)
>>> 1.0e-300 / 1.0e100 => 0.0 (Arithmetic underflow)
```

**Limits for floating Point Precision** = 16 to 17 digits

```
>>> 1/3
0.3333333333333333 (This is an approximation to 16 digits)
```

```
>>> 3 * (1/3) [Will not result in 0.9999999999999999]
1.0 (Rounded Off)
```



```
>>> 1.9999999999999999
2.0 (Rounded Off)
```

```
>>> 1.9999999999999998
1.9999999999999998 (Not rounded off)
```

```
>>> 1.9999999999999999 (1 fractional digit lesser than previous example)
1.9999999999999999 (Not rounded off)
```

### **More Examples:**

```
>>> 1/10          >>> 6 * (1/10)
0.1               0.6000000000000001
```

```
>>> 10 * (1/10)    >>> 1/10 + 1/10 + 1/10
1.0                0.30000000000000004
```

### **MORAL OF THE STORY:**

No matter how Python chooses to display calculated results, the value stored is limited in both the range of numbers that can be represented and the degree of precision. For most everyday applications, this slight loss in accuracy is of no practical concern. However, in scientific computing and other applications in which precise calculations are required, this is something that the programmer must be keenly aware of.

### **Program 4:**

### **Output:**

#### **#formatting function**

#formatting function

```
print(12/5 )
print(format(12/5, '0.2f'))
print(format(12/5, '0.3f'))
print(5/7)
print(format(5/7, '0.2f'))
print(format(11/12, '.3e'))
print(format(11/12, '.4e'))
print(format(11/12, '.2e'))
print(format(2**100, '.6e'))
```

```
2.4
2.40
2.400
0.7142857142857143
0.71
9.167e-01
9.1667e-01
9.17e-01
1.267651e+30
```

**Guess The Output:**

```
>>> print( 'Let's Go!' )
>>> print( 'Subhash' 'Python' 'Zebra' )
>>> print( 'Let's Go!' )
>>> print( "Lets Go!" )
>>> ord('A')           #give the ascii value of
>>> ord('6')
>>> ord('65')
>>> ord(65)
>>> chr(65)
>>> chr('48')
>>> format('Hello', '<20')
>>> format('Hello', '>20')
>>> format('Hello', '^20')
>>> format('Hello', '.^20')
>>> format(65, '>20')
>>> print(format('-', '<20'), 'Hello World', format('-', '->20'))
>>> format( 1/3, ">20" )
```

**More On Variables:**

```
>>> n = 10
>>> n = n + 1
>>> n
11
>>> k = n
>>> k
11
>>> id(n)
123456
>>> id(k)
123456
>>> n = "Hello"
>>> n
Hello
>>> n = 12.5
>>> n
12.5
```

**Guess The Output:**

```
>>> n = 10
>>> id(n)

>>> print ("1""2""3")

>>> k = n
>>> id(k)

>>> print (1,2,3)

>>> n = n + 1
>>> id(n)
>>> id(k)

>>> print("Hello""Hello""Hello")
```

```
>>> print("Hello","Hello","Hello")
```

```
>>> print ("1"2"3")
```

```
>>> print( "Hello,  
World")
```

```
>>> print( "Hello, \  
World" )
```

**Rules For Naming Variables:**

- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

**Identify The Wrong Variable Names (or Identifiers):**

a) iLoveIndia    b) 1LoveIndia    c)Love-India    d)Love India    e) \_LoveIndia

f) and g) print h) Break

**List Of Keywords In Python:**

and	as	assert	break	class	continue	def
del	elif	else	except	finally	for	from
global	if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try	while
with	yield	False	None	True		

There are many predefined identifiers that can be used as regular identifiers, but should not be. This includes float, int, print, exit, and quit. To check if an identifier is part of built-in identifiers, then you can use the following command.

```
>>> 'exit' in dir(__builtins__)
```

```
>>> 'subhash' in dir(__builtins__)
```

```
>>> 'print' in dir(__builtins__)
```

### **Assignment Program**

#### **Restaurant Bill Calculation Program:**

This program will calculate a restaurant bill for a couple with a gift coupon, with a restaurant tax of 18.0 %

Enter amount of the gift coupon:2000

Enter ordered items for person 1

Chicken Biryani : 120

Chicken Ghee Roast: 160

Coke Drink : 25

Vanilla Ice Cream : 35

Enter ordered items for person 2

Vegetable Biryani : 50

Gobi Manchurian : 30

Coke Drink : 25

Vanilla Ice Cream : 35

Bill without tax : 480.00

Total tax : 86.40

Total bill payable : 566.40

Amount balance in coupon : 1433.60

## Lesson-4: Operators & Expressions

### Types Of Operators:

#### Based on number of operands

- Unary
- Binary
- Ternary

#### Based on type of operations

- Arithmetic operators
- Assignment operators
- Unary Minus operator
- Relational operators
- Logical operators
- Bitwise operators
- Membership operators
- Identity operators

#### Precedence:

#### Associativity

**	(RL)
-	(LR)
*, /, //, %	(LR)
+, -	(LR)
<, >, <=, >=, !=, ==	(LR)
not	(LR)
and	(LR)
or	(LR)

#### Program: 1

```
a = 5
b = 3
c = 1

print(-a)
print(a + b)
print(a - b)
print(a * b)
print(a / b)
print(a % b)
print(a // b)
print(a ** b)
```

#### Output:

```
-5
8
2
15
1.6666666666666667
2
1
125
```

**Keep in mind:**

int / int = float  
 int / float = float  
 float / int = float  
 float / float = float

int // int = truncated int  
 int // float = truncated float  
 float // int = truncated float  
 float // float = truncated float

**Difference between “Coercion” and “Type Conversion”**

- 1) **Coercion:**  $2 + 4.5 \rightarrow 2.0 + 4.5 \rightarrow 6.5$  (Automatic Conversion)
- 2) **Conversion:**  $\text{float}(2) + 4.5 \rightarrow 2.0 + 4.5 \rightarrow 6.5$  (Forced Conversion)  
 $2 + \text{int}(4.5) \rightarrow 2 + 4 \rightarrow 6$  (Forced Conversion)

**Guess The Output:**

```
>>> print(7/5) >>> print( 7/5.0 ) >>> print( 7.0/5.0 )
```

```
>>> print( 7//5 ) >>> print( 7//5.0 ) >>> print( 7.0//5.0 )
```

```
>>> print( 0 % 5 )
```

```
>>> print( 3 * 2 + 1 ** 2 - 1 / 2 // 3 - 1 % 2 + 1 )
```

**Output: 7.0****Program: 2**

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

```
a = 1; b = 2
print(a, b)
```

```
a, b = 1, 2
print(a, b)
```

```
n = 10
print(-n)
```

**Output:**

```
5 5
1 2
1 2
-10
```

**Keep in mind:**

**x and y** ( if x is False, it returns x, otherwise it returns y )  
**x or y** ( if x is False, it returns y, otherwise it return x )  
**not x** ( if x is False, it returns True, otherwise False )

**NOTE:** All non-zero values are considered 'True' and zero value is considered 'False'

**Program: 3**

```
print( 100 and 200 )  
print( 0 and 200 )  
print( 200 and 0 )  
print( 100 or 200 )  
print( 0 or 200 )  
print( 200 or 0 )  
print( not 10 )  
print( not 0 )  
print(True and 100)
```

**Ouput:**

```
200  
0  
0  
100  
200  
200  
False  
True  
100
```

**Program: 4**

```
print(1 < 2)  
print(1 < 2 < 3)  
print(4 > 2 >= 2 > 1)  
print(1 < 2 > 3 < 4)  
print(1 > 4 == 3 < 4 != 3)
```

**Output:**

```
True  
True  
True  
False  
False
```

**Program: 5**

```
a = 9  
b = 2  
c = 0  
print( a << b )  
print( a >> b )  
print( a & b )  
print( a | b )  
print( a ^ b )  
print( ~c )
```

**Output**

```
36  
2  
0  
11  
11  
-1
```

**Program: 6**

```
print(2 + 3 < 4 / 2 and 4 // 5 == 5 % 2 or 3 - 1)
```

**Output:**

```
2
```

**Program: 7**

```
a = 5
b = 3
c = 1

print( 2 + 3 * 5 )
print( (2 + 3) * 5 )
print( 2 * 3 * 5 )
print( 2 * (3 * 5) )
print( 2 ** 2 ** 3 )
print( 2 + 3 / 1 * 2 % 3 - 1 )
```

**Output:**

```
17
25
30
30
256
1.0
```

**Guess The Output:**

```
>>> int(10)
```

```
>>> int('10')
```

```
>>> int('10.8')
```

```
>>> float(10)
```

```
>>> float('10')
```

```
>>> float('10.8')
```

```
>>> eval(input("Enter an expression:\n"))
```

```
>>> int(0b1010)
```

```
>>> float(0b1010)
```

```
>>> int("0xA", 16)
```

```
>>> bin(10)
```

```
>>> hex(10)
```

```
>>> oct(10)
```

**Program 8:**

```
l = [ "Subhash", "Charan", "Amitabh" ]
print ("Subhash" in l)
print ("Subhash" not in l)
print ("Akshay" not in l)
```

**Output:**

```
True
False
True
```

**Program 9:**

```
a = 10
b = 10

print( a is b )
print( a is not b )

a = 11

print( a is b )
print( a is not b )
```

**Output:**

```
True
False
False
True
```



**Program 10:**

```
print( 1 / 0 )
```

**Output:**

```
Run-Time Error  
Zero Division Error
```

**Program 11:**

```
n = 10  
print( n != 0 or 1 / 0 )
```

**Output:**

```
True
```

**Guess The Output:**

```
>>> 10 == 20 >>> 10 != 20 >>> 10 <= 20 >>> '2' < '9'
```

```
>>> '12' > '9' >>> '1' > '000' >>> "99" > "11" >>> 'heaven' > 'heau'
```

**Programming Assignments:**

1. WAP to display the powers of 2, one per line. (2 power 1, 2 power 2 and so on till 2 power 10 ).
2. WAP to enter a number and print 2 raised the number you entered.
3. WAP to enter a base and a power and find the base raised to that power.
4. WAP that allows user to enter 4 binary digits and convert it into decimal number.
5. WAP to print ascii values of "Hello World".
6. WAP to convert a degree from fahrenheit to celcius.
7. WAP to enter two float point values and displays the result of the first number divided by the second, with exactly six decimal places displayed in scientific notation.
8. WAP to enter an uppercase letter or a lowercase letter and display the corresponding UNICODE encoding.
9. WAP that prompts the user for a certain number of cities for the Travelling Salesman problem, and displays the total number of possible routes that can be taken.

## Lesson-5: Control Structures

### 3 Types Of Control Structures:

- Sequential Control Structures,
  - Selection Based Control Structures
  - Iteration Based Control Structures
- 
- **What are Sequential Control Structures?**
    - Step-By-Step as it appears in the code
  - **What are Selection Control Structures?**
    - Based on a specific condition
    - if, if...else, if...elif...else
  - **What are Iterative Control Structures?**
    - Based on a specific condition in a repetitive manner
    - while, for
  - **Other interesting topics**
    - 'else' suite, break, continue, pass, assert

#### Program: 1

```
print("hello")  
print ("I Love India")
```

```
a = 10  
b = 20  
sum = 10 + 20
```

```
a = 10  
b = 10
```

```
print(id(a))  
print(id(b))
```

#### **Output:**

```
hello  
I Love India  
2280991517264  
2280991517264
```

#### Program: 2

```
if 1 == 1:  
    print("Hello")  
print("Hi")
```

#### **Output:**

```
Hello  
Hi
```

**Program: 3**

```
percentage = int(input("Enter Percentage"))
if percentage >= 70:
    print("Distinction")
else:
    print("First Class")
```

**Output:**

```
Enter Percentage
65
First Class
```

**Program: 4**

```
percentage = int(input("Enter Percentage\n"))
if percentage >= 70:
    print("Distinction\n")
else:
    if percentage >= 60:
        print("First Class\n")
    else:
        if percentage >= 50:
            print("Second Class\n")
        else:
            if percentage >= 40:
                print("Third Class\n")
            else:
                print("Fail\n")
```

**Output:****Program: 3**

```
percentage = int(input("Enter Percentage\n"))
if percentage >= 70:
    print("Distinction\n")
elif percentage >= 60:
    print("First Class\n")
elif percentage >= 50:
    print("Second Class\n")
elif percentage >= 35:
    print("Third Class\n")
else:
    print("Fail\n")
```

**Output:****Program: 4**

```
n = 5
while n != 0:
    print(n)
    n = n - 1
```

**Output:**

**Guess The Output:**

- 1)  

```
percentage = int(input("Enter Percentage"))
if percentage >= 70:
    print("Distinction")
else:
    print("First Class")
```
- 2)  

```
percentage = int(input("Enter Percentage"))
if percentage >= 70:
    print("Distinction")
```
- 3)  

```
percentage = int(input("Enter Percentage"))
if percentage >= 70:
    print("Distinction")
else:
    print("First Class")
```
- 4)  

```
n = 5
while(n != 0 )
    print(n)
```
- 5)  

```
n = 5
while(n):
    print(n)
```
- 6)  

```
n = 10
sum = 0
current = 1
while current <= n:
    sum = sum + current
print(sum)
```
- 7)  

```
x = int(input("Enter a number greater than 0: "))
assert x > 0, "Wrong input entered"
print("U entered: ", x)
```

**Program: 5**

```
i = 1
while i <= 5:
    print(i)
    i += 1
else:
    print("Done with printing 5 numbers\n")
```

**Output:****Program: 6**

```
i = 1
while i < 10:
    print(i)
    if i == 5:
        break;
    i += 1
else:
    print("I Love India")
```

**Output:****Program: 7**

```
i = 1
while i < 10:
    if i == 5:
        i += 1
        continue
    print(i)
    i += 1
```

**Output:****Program: 8**

```
i = 1
while i < 10:
    if i == 5:
        pass
    print(i)
    i += 1
```

**Output:****Program: 10**

```
n = 1 in (1,2,3)
print(n)

n = 4 in (1,2,3)
print(n)

n = 1 in [1,2,3]
print(n)

n = "Subhash" in [ "Amitabh", "Aamir", "Shahrukh" ]
print(n)
```

**Output:**

**Programming Assignments:**

1. WAP to find the area of a circle.
2. WAP to find whether given number is odd or even.
3. WAP to find whether given number is positive or negative.
4. WAP to find the biggest of 3 numbers.
5. WAP to find whether given year is leap year or not
6. WAP to input a month number 1 to 12 and print how many days in that month
7. WAP to convert from Fahrenheit to Celsius and vice-versa. Ask user for 'F' or 'C' and then carry out the operation. Ask user to re-input for input other than 'F' and 'C'.  
    **[Formula for F to C: celsius = (fahr - 32) \* 5.0/9.0]**  
    **[Formula for C to F: fahr = (9.0/5.0 \* celsius) + 32]**
8. WAP to find GCD.
9. WAP to print fibonacci series.
10. WAP to find whether a given number is prime or not.

**More Assignment Problems:**

1. Write a Python program in which the user enters either 'A', 'B', or 'C'. If 'A' is entered, the program should display the word 'APPLE'; if 'B' is entered, it displays 'BANANA'; and if 'C' is entered, it displays 'COCONUT'.
2. Write a program that sums a series of (positive) integers entered by the user, excluding all numbers that are greater than 100.
3. Write a program, in which the user can enter any number of positive and negative integer values, that displays the number of positive values entered, as well as the number of negative values.
4. Write a program containing a pair of nested while loops that displays the integer values 1-100, ten numbers per row, with the columns aligned neatly in an order.

**Lesson-6: Sequence Types (List & Tuple)**

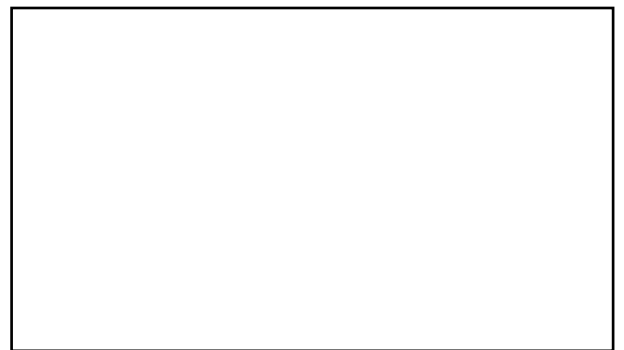
- List is a linear mutable data structure
- Tuple is a linear immutable data structure
- Sequences follow zero-based indexing

0	1	2	3	4
3	5	1	9	8
-5	-4	-3	-2	-1

**Program: 1**

```
ints      = [1, 2, 3]
strings   = ['abc','def','ghi']
floats    = [1.2, 2.3, 3.4]
mixed     = [1, 2, 'abc', 1.2]
empty     = [ ]
```

```
print( ints )
print( strings )
print( floats )
print( mixed )
print(empty)
```

**Output:****Draw The Diagram:**

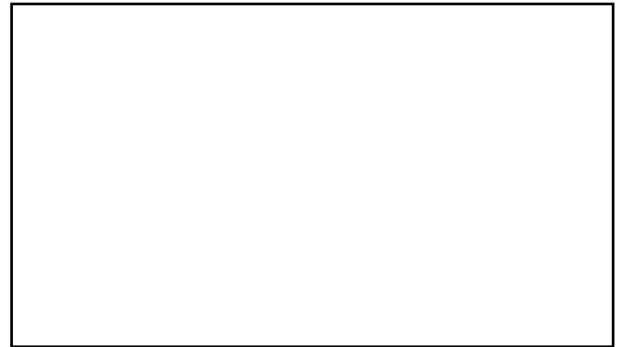
**Original:** l = [1, 2, 3]

1. l[2] = 5
2. del l[1]
3. l.insert(2,5)
4. l.append(6)
5. l.sort()
6. l.reverse()
7. l.sort(reverse=True)

**Program: 2**

```
ints      = (1, 2, 3)
strings   = ('abc','def','ghi')
floats    = (1.2, 2.3, 3.4)
mixed     = (1, 2, 'abc', 1.2)
empty     = ()
```

```
print( ints )
print( strings )
print( floats )
print( mixed )
print(empty)
```

**Output:****Guess The Output:**

**Original:** 1) l = (5, 2, 3)    2) s = "Subhash"

```
>>> l.sort()           >>> l.reverse()           >>> l.append(6)
```

```
>>> l.insert(2, 'Hello')  >>> del l[1]           >>> s.reverse()
```

```
>>> s.sort()
```

**Guess The Output:**

<pre>s = ( 1, 2, 3, 4 ) w = ( 5, 6 ) t = ( 1,3.5,3 ) u = ( 1, '3', 4 )</pre>	<pre>s = [ 1, 2, 3, 4 ] w = [ 5,6 ] t = [ 1,3.5,3 ] u = [ 1, '3', 4 ]</pre>	<pre>s = 'hello' w = '!'</pre>
<pre>print(len(s)) print(s[0]) print(s[1:4]) print(s[1:]) print(s.count('e')) print(s.count(4)) print(s.index('e')) //Error print(s.index(3)) print('h' in s) print(s + w) print(min(s)) print(max(s)) print(sum(s)) print(max(t)) //3.5 print(max(u)) //Error</pre>	<pre>print(len(s)) print(s[0]) print(s[1:4]) print(s[1:]) print(s.count('e')) print(s.count(4)) print(s.index('e')) //Error print(s.index(3)) print('h' in s) print(s + w) print(min(s)) print(max(s)) print(sum(s)) print(max(t)) //3.5 print(max(u)) //Error</pre>	<pre>print(len(s)) print(s[0]) print(s[1:4]) print(s[1:]) print(s.count('e')) print(s.count(4)) //Error print(s.count('4')) //0 print(s.index('e')) print(s.index(3)) //Error print('h' in s) print(s + w) print(min(s)) print(max(s)) print(sum(s)) //Error</pre>



**Guess The Output:**

```
>>> [1, 2, 3] + ( 4, 5, 6 )           >>> (1)           >>>(1,)
```

```
>>> 'coco' + 'nut'                     >>> s = [10, 30, 20, 10]
>>>                                     >>> s[1:3]
```

```
>>> s = "Subhash"                      >>> s = (20,30,50,40,90,100)
>>> s[4:]                              >>> s[1:5]
```

**Draw The Diagram:**

```
l = [ [2,3,4], [5,4,7], [2,5,6] ]
```

```
t = ( (2,3,4), (5,6,7), (2,5,6) )
```

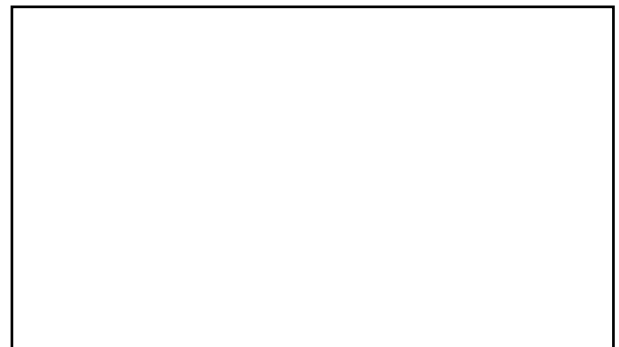
**Program: 3**

```
l = [ [2,3,4], [5,4,7], [2,5,6] ]
t = ( (2,3,4), (5,6,7), (2,5,6) )
```

```
print( l[1][1] )
print( t[1][1] )
```

```
print( l[2][0] )
print( t[2][0] )
```

```
print( l[0][2] )
print( t[0][2] )
```

**Output:****Program: 4**

```
#finding average score of the first subject of the class
```

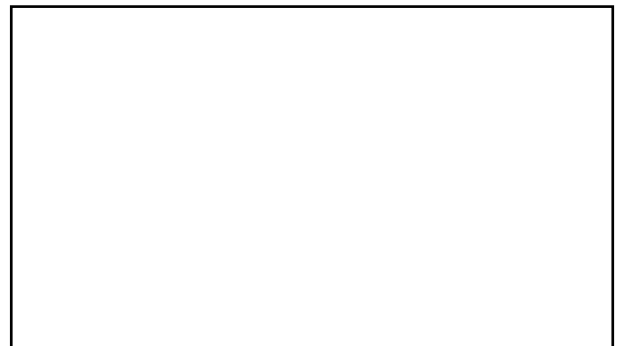
```
class_grades = [ [85,91,89], [78,81,86], [62,75,77] ]
```

```
k = 0
sum = 0
```

```
while (k < len(class_grades)):
    sum = sum + class_grades[k][0]
    k = k + 1
```

```
average = sum / len(class_grades)
```

```
print(average)
```

**Output:**

**Program: 5****Output:**

```
#finding average exam score for each student in class

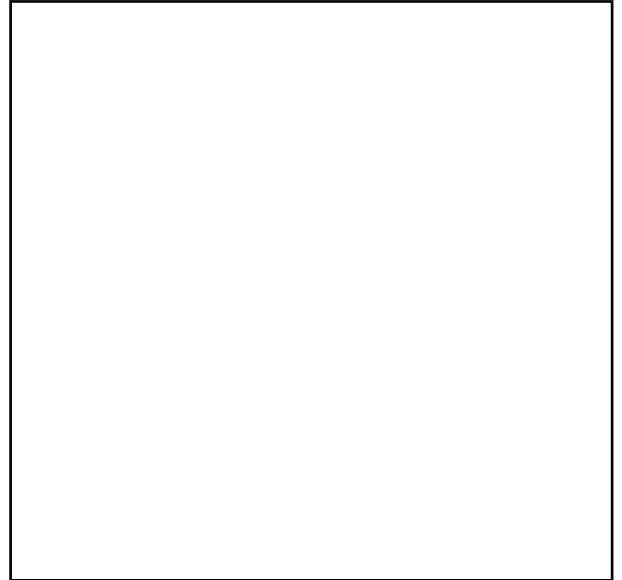
#finding average exam score for each student in class

class_grades = [ [85,91,89], [78,81,86], [62,75,77] ]

i = 0
j = 0
avg = []
sum = 0

while ( i < len(class_grades)):
    while(j < len(class_grades[i])):
        sum = sum + class_grades[i][j]
        j = j + 1

    sum = sum / len(class_grades[i])
    print( "The average of {0}th student is {1}".format(i,sum))
    i = i + 1
    j = 0
    sum = 0
```

**Program: 6****Output:**

```
nums = [ 1, 3, 4, 5 ]

for i in nums:
    print(i)

for i in (1, 2, 3, 4, 5):
    print(i)

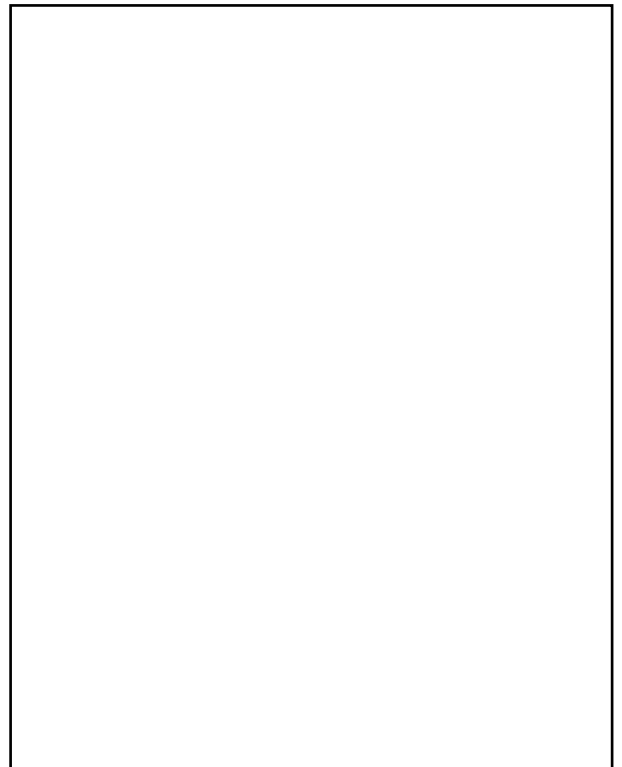
for i in "hello":
    print(i)

for i in range(5):
    print(i)

for i in range(3,5):
    print(i)

for i in range(0,10,2):
    print(i)

for i in range(10,0,-2):
    print(i)
```



**Guess The Output:**

```
>>> for i in range [0, 11]:
    print(i)
```

```
>>> for i in range(10, -1, -2):
    print(i)
```

**Write the 'while' equivalent of the following:****Original Code:**

```
nums = [ 1, 3, 4, 5 ]
```

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

**Equivalent 'while' code:****Program: 7**

#This is a self-observation program to analyse and appreciate python constructs

```
password_out = ""
case_changer = ord('a') - ord('A')
encryption_key = (('a','m'), ('b','h'), ('c','t'), ('d','f'), ('e','g'), ('f','k'), ('g','b'), ('h','p'), ('i','j'), ('j','w'), ('k','e'), ('l','r'),
('m','q'), ('n','s'), ('o','l'), ('p','n'), ('q','i'), ('r','u'), ('s','o'), ('t','x'), ('u','z'), ('v','y'), ('w','v'), ('x','d'), ('y','c'), ('z','a'))
```

```
print( "This program will encrypt and decrypt user passwords\n" )
```

```
which = input("Enter (e) to encrypt a password, and (d) to decrypt a password\n")
```

```
while which != 'e' and which != 'd':
    which = input("INVALID - Enter 'e' to encrypt, 'd' to decrypt\n")
```

```
encrypting = (which == 'e')
```

```
password_in = input("Enter password:")
```

```
if encrypting:
    from_index = 0
    to_index = 1
else:
    from_index = 1
    to_index = 0
```

```
case_changer = ord('a') - ord('A')
```

```
for ch in password_in:
    letter_found = False
```

```
    for t in encryption_key:
        if ('a' <= ch and ch <= 'z') and ch == t[from_index]:
            password_out = password_out + t[to_index]
            letter_found = True
        elif ('A' <= ch and ch <= 'Z') and chr(ord(ch) + 32) == t[from_index]:
            password_out = password_out + chr(ord(t[to_index]) - case_changer)
            letter_found = True
```

```
    if not letter_found:
```

```
password_out = password_out + ch

if encrypting:
    print( "Your encrypted password is: ", password_out)
else:
    print( "Your decrypted password is: ", password_out)
```

**Output:**

```
$ python3 encrpyt_decrypt_program.py
```

This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt a password

e

Enter password:subhash

Your encrypted password is: ozhpmop

```
$ python3 encrpyt_decrypt_program.py
```

This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt a password

d

Enter password:ozhpmop

Your decrypted password is: subhash

```
$ python3 encrpyt_decrypt_program.py
```

This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt a password

e

Enter password:8889

Your encrypted password is: 8889

**Program: 8****Output:**

```
nums_one = [ 1, 3, 4, 5 ]
num_two = nums_one
```

```
nums_two[2] = 8
print( nums_one )
print( nums_two )
```

```
print( id(nums_one) )
print( id(nums_two) )
```

```
l_one = [1,2,3]
l_two = list(l_one)
print(l_one)
print(l_two)
```

```
l_two[0] = 0
```

```
print(l_one)
print(l_two)

print( id(l_one) )
print( id(l_two) )
```

**Guess The Output:**

**Note:** These are called 'List Comprehensions'

```
>>> [ x for x in range(10) if (x % 2 == 0) ]

>>> [ x**2 for x in range(5) ]

>>>[ x for x in [-1, 1, -2, 2, -3, 3, -4, 4] if( x >= 0) ]

>>> [ord(x) for x in 'Hello']

>>> vowels = ('a', 'e', 'i', 'o', 'u' )
>>> w = 'Hello'
>>> [ch for ch in w if ch in vowels]

>>> temperatures = [88,94,97,89,101,98,102,95,100]
      [ (t - 32) * 5/9 for t in temperatures ]

>>> a, b = [int(x) for x in input("Enter 2 numbers").split()]

>>> eval("[1,2,3] + [1,2,3]")
```

**Program 9:**

```
l = range(0,9)
print(l)

l = list(range(0,9))
print(l)

lone = [1,2,3,4,5]
lone[1:2] = 10,11
print(lone)

print(lone * 2)

ltwo = lone
ltwo[0] = 23
```

**Output:**

```
lone[3] = 35
print(lone)
print(ltwo)

lthree = ltwo[:]
lthree[1] = 89
print(lthree)
print(ltwo)

lfour = lthree.copy()
lfour[1] = 98
print(lthree)
print(lfour)

nl = [9,7,6]
lfive = [1,2,3,nl]
print(lfive)

for i in lfive[3]:
    print(i, end = " ")
print()

print(lfive[3][0])
print(lfive[3][1])
print(lfive[3][2])

lsix = [[1,2,3],[4,5,6],[7,8,9]]

for lseven in lsix:
    print(lseven)

for lseven in lsix:
    for num in lseven:
        print(num, end = " ")
    print()

leight = [1,2,3]
lnine = [1,1,1]
lten = [i + j for i in leight for j in lnine ]
print(lten)

leleven = [ 4 * [0] for i in range(3) ]
print(leleven)

leleven = [ 4 * [i] for i in range(3) ]
print(leleven)
```

**Program 10:**

```
tone = 1,2,3
print(tone)

l = [11,22,33]
ttwo = tuple(l)
print(ttwo)

tthree = tuple(range(11,20))
print(tthree)

print(tthree[:2])

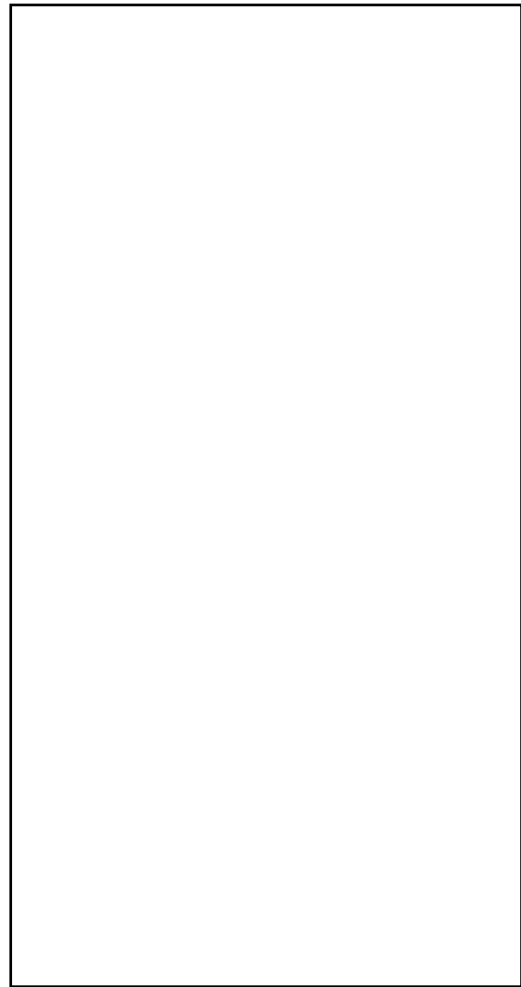
num1, num2 = tone[0:2]
print(num1)
print(num2)

tfour = tthree * 2
print(tfour)

tfive = (1,2,3,(4,5,6))

for t in tfive:
    print(t)

for t in tfive[3]:
    print(t)
```

**Output:****Few More Methods Of List – REMOVE, POP and EXTEND:**

```
>>> l = [1,2,3]
>>> l.remove(1)
>>> l
[2,3]
```

```
>>> l = [1,2,3]
>>> l.pop()
3
>>> l
[1,2]
>>> l.pop(0)
1
>>> l
[2]
```

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

**Program 11:**

Given two different sequences, write a program to find out how many elements are common in those sequences. **Hint:** Use **zip()** method.

```
count = 0
```

```
myname = ['s','u','b','h','a','s','h']  
yourname = ('S','u','B','H','a','S','h')
```

```
for m, y in zip(myname, yourname):  
    if m == y:  
        count = count + 1;
```

```
print(count, "values are equal")
```

**NOTE:** How **zip()** function works?

```
>>> l = [1,2]  
>>> t = (3, 4)  
>>> zip(l,t)  
>>> for pair in zip(l,t):  
    print(pair)
```

**Output:**

```
(1, 3)  
(2, 4)
```

**Program 12 (Same Program, Broken To One More Step):**

Given two different sequences, write a program to find out how many elements are common in those sequences. **Hint:** Use **zip()** method.

```
count = 0
```

```
myname = ['s','u','b','h','a','s','h']  
yourname = ('S','u','B','H','a','S','h')
```

```
for pair in zip(myname, yourname):  
    print(pair)  
    m,y = pair  
    if m == y:  
        count = count + 1;
```

```
print (count, "values are equal")
```

**Output:**

```
('s', 'S')  
(u', 'u')  
(b', 'B')  
(h', 'H')  
(a', 'a')  
(s', 'S')  
(h', 'h')  
3 values are equal
```



**Format Operator:**

```
print('sum of %d and %d is %d' % (1, 2, 3))  
print('My name is %s and I work as %s since %d years' % ('Subhash', 'Programmer', 15))  
print('My name is %s and I work as %s since %f years' % ('Subhash', 'Programmer', 10))
```

**Output:**

```
sum of 1 and 2 is 3  
My name is Subhash and I work as Programmer since 15 years  
My name is Subhash and I work as Programmer since 10.000000 years
```

**Programming Assignments:**

1. WAP to determine whether given number occurs in list 'nums'
2. WAP that prompts the user for a list of integers, stores in another list only those values between 1-100, and displays the resulting list
3. WAP that prompts the user for a list of integers, stores in another list only those values that are in tuple 'valid\_values', and displays the resulting list.
4. WAP that prompts the user for a list of integers and stores them in a list. For all values that are greater than 100, the string 'josh' should be stored instead. The program should display the resulting list.
5. WAP that prompts the user to enter a list of first names and stores them in a list. The program should display how many times the letter 'a' appears within the list.
6. WAP that prompts the user to enter integer values for each of two lists. It then should display the lists are of the same length, whether the elements in each list sum to the same value and whether there are any values that occur in both lists.
7. WAP to modify or replace an existing element of a tuple with a new element
8. WAP to find the first occurrence of an element in a tuple
9. WAP to accept elements in the form of a tuple and display their sum and average
10. WAP to add two matrices and display the sum matrix using lists
11. WAP to sort the list elements using bubble sort technique

## Lesson-6: Functions

### **Program: 1**

```
def got_called():  
    print("Hello")  
  
got_called()
```

**Output:**

### **Program: 2**

```
def got_called():  
    print("Hello")  
    return 5  
  
r = got_called()  
print(r)
```

**Output:**

### **Program: 3**

```
def got_called():  
    print("Hello")  
    return  
  
r = got_called()  
print(r)
```

**Output:**

### **Program: 4**

```
def got_called(a,b):  
    print("Hello")  
    return (a+b)  
  
a = 5  
b = 6  
r = got_called(a,b)  
print(r)
```

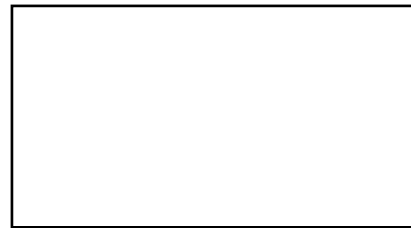
**Output:**

**Program: 5**

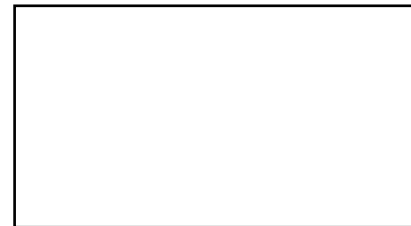
```
def got_called(b,a):  
    print("a = ", a)  
    print("b = ", b)  
    return (a+b)  
  
a = 5  
b = 6  
r = got_called(a,b)  
print(r)
```

**Output:****Program: 6**

```
def got_called(b,a):  
    print("a = ", a)  
    print("b = ", b)  
    return (a+b)  
  
r = got_called(a = 5,b = 6)  
print(r)
```

**Output:****Program: 7**

```
def got_called(b,a,c=3):  
    print("a = ", a)  
    print("b = ", b)  
    print("c = ", c)  
    return (a+b+c)  
  
r = got_called(5,6,7)  
print(r)  
r = got_called(5,6)  
print(r)
```

**Output:****Program: 8**

```
def max_min(l):  
    return (max(l), min(l))  
  
l = [1,2,3,4,5]  
max, min = max_min(l)  
print("max of ",l, "is ", max)  
print("min of ",l, "is ", min)
```

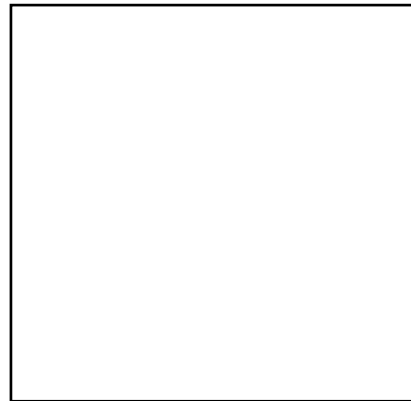
**Output:**

**Program: 9**

```
def ftwo():
    n = 10
    print("n = ", n)

def fone():
    n = 20
    print("n = ", n)
    ftwo()
    print("n = ", n)

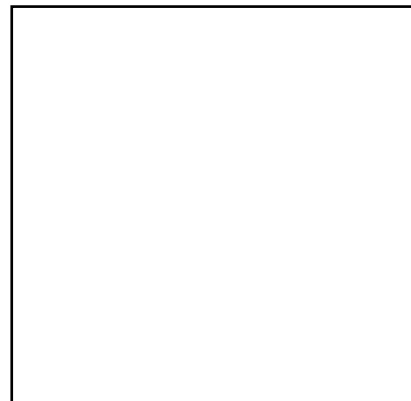
fone()
```

**Output:****Program: 10**

```
def ftwo():
    print("n = ", n)

def fone():
    n = 20
    print("n = ", n)
    ftwo()
    print("n = ", n)

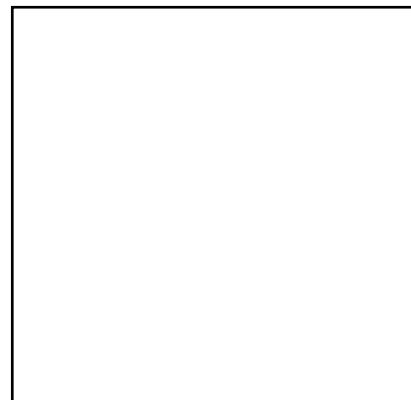
fone()
```

**Output:****Program: 11**

```
def fun_two():
    print("n = ", n)

def fun_one():
    n = 20
    print("n = ", n)
    fun_two()
    print("n = ", n)

n = 9
fun_one()
print(n)
```

**Output:**

**Program: 12**

```
def factorial(n):
    if( n == 0):
        return 1
    else:
        return n * factorial(n-1)

n = int(input("Enter a number to find factorial of it\n"))
f = factorial(n)
print("The factorial of " , n , " is " , f )
```

**Output:****Program: 13**

```
def fun(*args):
    for i in args:
        print(i, end=' ')
    print()

fun(1)
fun(1,2)
fun(1,2,3)
fun(1,2,3,4)
```

**Output:****Program: 14**

```
f = lambda x: x * x
x = f(5)
print(x)
```

**Output:****Program: 15**

```
print((lambda x: x * x)(5))
```

**Output:****Program: 16**

```
f = lambda x, y: x if x > y else y
print(f(5,6))

print((lambda x, y: x if x > y else y)(5,6))
```

**Output:**

**Program: 17**

```
l = [1,2,3,4,5,6,7,8,9,0]
f = lambda x: x % 2 == 0
res = filter(f,l)
updated_list = list(res)
print(updated_list)
```

**Output:**

```
[2, 4, 6, 8, 0]
```

**Program: 18**

```
l = [1,2,3,4,5,6]
f = lambda x: x * x
res = map(f,l)
updated_list = list(res)
print(updated_list)
```

**Output:**

```
[1, 4, 9, 16, 25, 36]
```

**Program: 19**

```
lone = [1,2,3,4,5,6,7,8,9,0]
ltwo = [1,2,3,4,5,6,7,8,9,0]
f = lambda x, y: x * y
res = map(f,lone,ltwo)
updated_list = list(res)
print(updated_list)
```

**Output:**

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 0]
```

**Program: 20**

```
from functools import *

l = [1,2,3,4,5,6,7,8,9]
f = lambda x, y: x * y
res = reduce(f,l,)
#res = reduce(f,l,0)
print(res)
```

**Output:**

```
362880
```

**Program: 21**

```
def decorator( ready_for_decoration ):
    def helping_to_decorate( ):
        x = ready_for_decoration( )
        return x + 1
    return helping_to_decorate
```

```
def getting_decorated( ):
    return 10
```

```
decorated = decorator(getting_decorated)
print(decorated( ))
```

**Output:**

```
11
```

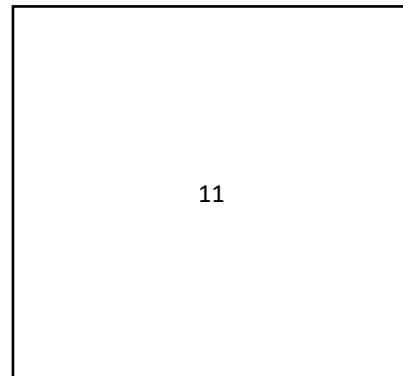
**Program: 22**

```
def decorator( ready_for_decoration ):
    def helping_to_decorate():
        x = ready_for_decoration()
        return x + 1
    return helping_to_decorate
```

**@decorator**

```
def getting_decorated():
    return 10
```

```
print(getting_decorated())
```

**Output:**

11

**Program: 23**

```
def mydecorator1( f ):
    def inner():
        x = f()
        return x + 1
    return inner
```

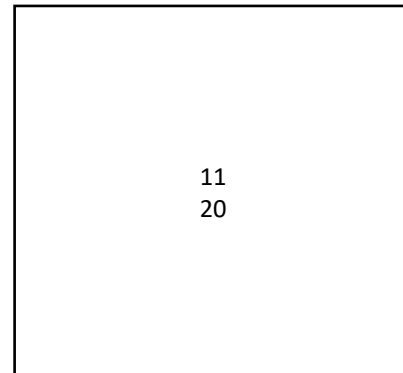
```
def mydecorator2( f ):
    def inner():
        x = f()
        return x * 2
```

```
    return inner
```

```
def myfunc():
    return 10
```

```
updated_func1 = mydecorator1(myfunc)
updated_func2 = mydecorator2(myfunc)
```

```
print(updated_func1())
print(updated_func2())
```

**Output:**

11  
20

**Program: 24**

```
def mydecorator1(f):  
    def inner():  
        x = f()  
        return x + 1  
    return inner
```

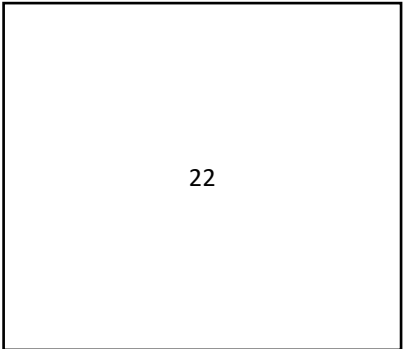
```
def mydecorator2(f):  
    def inner():  
        x = f()  
        return x * 2  
  
    return inner
```

```
@mydecorator2
```

```
@mydecorator1
```

```
def myfunc():  
    return 10
```

```
print(myfunc())
```

**Output:**

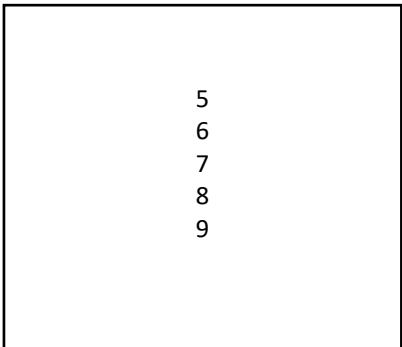
22

**Program: 25**

```
def my_generator(x):  
    for i in range(x):  
        r = 5 + i  
        yield r
```

```
g = my_generator(5)
```

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

**Output:**

5  
6  
7  
8  
9

**Guess The Output:**

1)

```
def avg(n1, n2, n3):  
    return (n1 + n2 + n3)/3.0
```

```
print(avg(10,25,40))  
print(avg(10,25,40) + 10)  
print(avg(avg(2,4,6),8,12))
```

2)

```
def avg(n1, n2, n3):  
    return (n1 + n2 + n3)/3.0
```



```
if avg(10, 25, -40) < 0:  
    print "Invalid Average"
```

3)

```
def maxmin(l):  
    return (max(l), min(l))  
  
l = [10,20,30]  
max_min = maxmin(l)  
print(max_min[0])  
print(max_min[1])
```

4)

```
def change( l ):  
    l[0] = 5  
  
def call( ):  
    l = [1,2,3]  
    change(l)  
    print(l)  
  
call( )
```

5)

```
def change( ):  
    ll[0] = 5  
  
def call( ):  
    ll = [1,2,3]  
    change( )  
    print(ll)  
  
ll = [1,2,3]  
call()  
print(ll)
```

6)

```
def call(z, a, b, c ):  
    print(a,b,c,z)  
    call(c = 6, b = 3, a = 1, 5 )
```

7)

```
def call( a = 10, b, c ):  
    print(a,b,c)  
  
call( 5,6 )
```

8)

```
print( "Hello" )
```

```
def hel():  
    print("Bello")
```

```
print("Kello")
```

9)

```
print( "Hello" )
```

```
def hel( ):  
    print("Bello")
```

```
print("Kello")  
hel()
```

10)

```
def hel( n ):  
    print(id(n))  
    n = n + 1  
    print(id(n))
```

```
n = 5  
print(id(n))  
hel(n)  
print(id(n))
```

11)

```
def hel( n ):  
    print(id(n))  
    n[0]=5  
    print(id(n))
```

```
n = [1,2,3]  
print(id(n))  
hel(n)  
print(id(n))
```

12)

```
def hel( ):  
    n = 6  
    print(id(n))  
    n = n + 1  
    print(id(n))  
n = 5  
print(id(n))
```

```
hel()  
print(id(n))
```

13)

```
def ret_example():  
    return 5, 6, 7  
  
r = ret_example()  
print(r)
```

14)

```
def fun():  
  
    def infun():  
        print("Hello")  
  
    infun()  
    return 5  
  
r = fun()  
print(r)
```

15)

```
def fun():  
  
    def infun():  
        print("Hello")  
  
    infun()  
    return infun  
  
f = fun()  
f()
```

16)

```
def fun():  
    lst = [1,2,3]  
    print(id(lst))  
  
lst = [1,2,3]  
fun()  
print(id(lst))
```

17)

```
def fun():  
    a = 2  
    print(a)
```

```
a = 3  
print(a)  
fun()  
print(a)
```

18)

```
def fun():  
    global a  
    a = 2  
    print(a)
```

```
a = 3  
print(a)  
fun()  
print(a)
```

19)

```
def fun():  
    x = globals()['a']  
    a = 2  
    print(a, x)
```

```
a = 3  
print(a)  
fun()  
print(a)
```

20)

```
def fun():  
    x = globals()['a']  
    y = globals()['b']  
    a = 2  
    print(a, x, y)
```

```
a = 3  
b = 5  
print(a)  
fun()  
print(a)
```

21)

```
lone = [1,2,3,4,5,6,7,8,9]  
f = lambda x: 0  
res = filter(f,lone)
```

```
print(list(res))
```

22)

```
lone = [1,2,3,4,5,6,7,8,9]
f = lambda x: 0
res = map(f,lone)
print(list(res))
```

23)

```
from functools import *
```

```
lone = [1,2,3,4,5,6,7,8,9]
f = lambda x,y: x + 0
res = reduce(f,lone)
print(res)
```

### **Programming Assignments:**

1. WAP to convert temperature from (Fahrenheit to Celcius) or (Celcius to Fahrenheit) based on user input for a specific range 'start' and 'end'  
[Formula: 'c = (f - 32) \* 5/9' and 'f = (5/9 \* c) + 32']
2. Write all previous programs using functions.
3. WAP with function names 'checkZero' that is given three integers, and returns 'True' if any of the integers is 0, otherwise it returns 'False'.
4. WAP with function named 'checkLowToHigh' that is passed three integers, and returns 'True' if the three integers are in order from smallest to largest, otherwise it returns 'False'.
5. WAP with function named 'isDivisible' that is given a positive integer, n, and a second positive integer, m <= n, and returns how many numbers between 1 and n are evenly divisible by m.
6. WAP with function named 'print\_name' that display "I love you <name>" for any given name passed to the function.

## Lesson-8: Modular Programming

### Modules:

- Module
- Interface
- Client

### Python Modules

#### Program: 1

**import math**

```
print( math.factorial(5) )
print( math.__doc__ )
print( )
print( math.factorial.__doc__ )
```

#### **Output:**

120

This module provides access to the mathematical functions defined by the C standard.

Find x!.

Raise a ValueError if x is negative or non-integral.

#### **This is my first module in python:**

**#save it as myfirstmodule.py**

```
""" This is my first module in python """
print("I am your first python module")
def sample_module():
    """ This is my first module function in python """
    print("Hello Module")
```

#### **Program: 2**

**import myfirstmodule**

```
myfirstmodule.sample_module()
print( myfirstmodule.__doc__ )
print( myfirstmodule.sample_module.__doc__ )
print( __name__ )
```

#### **Output:**

```
I am your first python module
Hello Module
This is my first module in
python
This is my first module function
in python
__main__
```

**#myfirstmoduleone.py**

```
def fun():  
    print("Hello fun - 1")
```

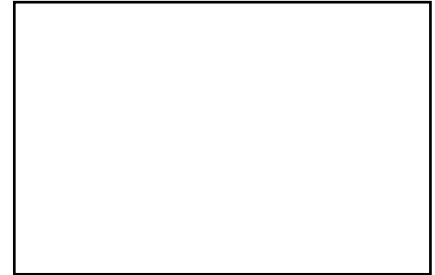
**#myfirstmoduletwo.py**

```
def fun():  
    print("Hello fun - 2")
```

**Program: 3**

```
import myfirstmoduleone, myfirstmoduletwo
```

```
fun()
```

**Output:****#moduleone.py**

```
def fun():  
    print("Hello fun - 1")
```

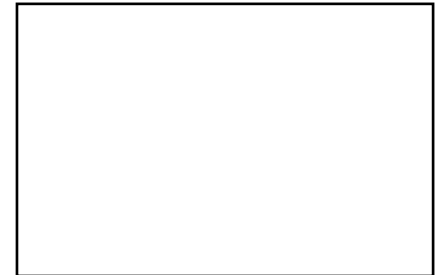
**#moduletwo.py**

```
def fun():  
    print("Hello fun - 2")
```

**Program: 4**

```
import moduleone, moduletwo
```

```
moduleone.fun()  
moduletwo.fun()
```

**Output:****Guess The Output:**

```
>>> factorial(5)
```

```
>>> math.factorial(5)
```

```
>>> import math
```

```
>>> factorial(5)
```

```
>>> math.factorial(5)
```

**#mymodule.py**

```
def fun_one():  
    print("Hello fun - 1")  
def fun_two():  
    print("Hello fun - 2")
```

**Program: 5**

```
from mymodule import fun_one  
  
fun_one()
```

**Output:****#mymodule.py**

```
def fun_one():  
    print("Hello fun - 1")  
def fun_two():  
    print("Hello fun - 2")
```

**Program: 6**

```
import mymodule  
from mymodule import fun_one  
  
fun_two()
```

**Output:****#mymodule.py**

```
def fun_one():  
    print("Hello fun - 1")  
def fun_two():  
    print("Hello fun - 2")
```

**Program: 7**

```
#check_module_program.py  
import mymodule  
from mymodule import fun_one  
  
mymodule.fun_two()
```

**Output:**



**#mymodule.py**

```
def fun_one():  
    print("Hello fun - 1")  
def fun_two():  
    print("Hello fun - 2")
```

**Program: 8****Output:**

```
import mymodule  
from mymodule import fun_one
```

```
def fun_one():  
    print("I am your function\n")
```

```
fun_one()  
mymodule.fun_two()
```

**#mymodule.py**

```
def fun_one():  
    print("Hello fun - 1")  
def fun_two():  
    print("Hello fun - 2")
```

**Program: 9****Output:**

```
import mymodule  
from mymodule import fun_one
```

```
fun_one()  
mymodule.fun_two()
```

```
def fun_one():  
    print("I am your function\n")
```

**#mymodule.py**

```
def fun_one():  
    print("Hello fun - 1")  
def fun_two():  
    print("Hello fun - 2")
```

**Program: 10**

```
import mymodule  
from mymodule import fun_one
```

```
def fun_one():  
    print("I am your function\n")
```

```
fun_one()  
mymodule.fun_two()
```

**Output:****Program: 11**

```
from math import factorial as f  
def factorial(n):  
    print(n)  
factorial(5)  
print(f(5))
```

**Output:****Program: 12**

```
from math import factorial as f  
def f(n):  
    print(n)  
f(5)
```

**Output:****#mymodule.py**

```
n = 20  
def fun_one():  
    print("Hello fun - 1")  
def fun_two():  
    print("Hello fun - 2")
```

**Program:13**

```
from mymodule import n  
  
print(n)
```

**Output:**

**#mymodule**

```
__n__ = 20
def fun_one():
    print("Hello fun - 1")
def fun_two():
    print("Hello fun - 2")
```

**Program: 14**

```
from mymodule import *

print(__n__)
```

**Output:**

**IMPORTANT NOTE:** When the **from module\_name import \*** form of import is used to import all the identifiers of a module's namespace, names beginning with double underscores are not imported. Thus, such entities become inaccessible from within the importing module.

**Guess The Output:**

1.

```
def sum(n1, n2, n3):
    total = n1 + n2 + n3
    return total
```

```
res = sum([1,2,3])
print(res)
```

2.

```
def sum(n1, n2, n3):
    total = n1 + n2 + n3
    return total
```

```
res = __builtins__.sum([1,2,3])
print(res)
```

**#grade\_calc module****#grade\_calc.py**

```
def max(grades):
    largest = 0

    for k in grades:
        if k > 100:
            largest = 100
        elif k > largest:
            largest = k
```

```
    return largest

def gradesHighLow(grades):
    return (min(grades), max(grades))
```

**Program: 15****#classgrades (main module)**

```
from grade_calc import *

class_grades = [86, 72, 94, 102, 89, 76, 96]

low_grade, high_grade = gradesHighLow(class_grades)
print('Highest adjusted grade on the exam was', high_grade)
print('Lowest grade on the exam was', low_grade)

print('The highest grade on exam was', max(class_grades))
print('Actual highest grade on exam was', __builtins__.max(class_grades))
```

**Program 16 (stack.py)****#stack module (LIFO)**

```
def getStack():
    """ Creates and returns an empty stack. """
    return [ ]

def isEmpty(s):
    """ Returns True if stack empty, otherwise returns False. """
    if s == [ ]:
        return True
    else:
        return False

def top(s):
    """ Returns value of the top item of stack, if stack not empty. Otherwise, returns None. """
    if isEmpty(s):
        return None
    else:
        return s[len(s) - 1]

def push(s, item):
    """ Pushes item on the top of stack. """
    s.append(item)

def pop(s):
    """ Returns top of stack if stack not empty. Otherwise, returns None. """
    if isEmpty(s):
```

```
        return None
    else:
        item = s[len(s) - 1]
        del s[len(s) - 1]
        return item
```

### **#main module - Using Stack - Client Program**

#### **import stack**

```
mystack = stack.getStack()
```

```
for item in range(1,5):
    stack.push(mystack, item)
    print('Pushing', item, 'on stack')
```

```
while not stack.isEmpty(mystack):
    item = stack.pop(mystack)
    print('Popping', item, 'from stack')
```

### **Program 17 (Palindrome or Not)**

#### **import stack**

#### **#welcome**

```
print("This program can determine if a given string is a palindrome\n")
print("(Enter return to exit)")
```

#### **#init**

```
char_stack = stack.getStack()
empty_string = ""
```

#### **#get string from user**

```
chars = input("Enter string to check: ")
```

```
while chars != empty_string:
    if len(chars) == 1:
        print('A one letter word is by definition a palindrome\n')
    else:
        #init
        is_palindrome = True
```

#### **#to handle strings of odd length**

```
compare_length = len(chars) // 2
```

#### **#push second half of input string on stack**

```
for k in range(compare_length, len(chars)):
    stack.push(char_stack, chars[k])
```

#### **#pop chars and compare to first half of string**

```
k = 0
while k < compare_length and is_palindrome:
    ch = stack.pop(char_stack)
    if chars[k].lower() != ch.lower():
        is_palindrome = False

    k = k + 1

#display results
if is_palindrome:
    print( chars, " is a palindrome\n")
else:
    print( chars, " is not a palindrome" )

#get next string from user
chars = input("Enter string to check: ")
```

**Output:**

This program can determine if a given string is a palindrome

(Enter return to exit)

Enter string to check: bool

bool is not a palindrome

Enter string to check: madam

madam is a palindrome

Enter string to check:

>>>

## Lesson – 9: File Management

### Difference b/w Text File & Binary File

A text file stores data in the form of alphabets, digits and other special symbols by storing their ASCII values and are in a human-readable format (.txt, .c, .py etc.). Whereas binary file contains a sequence or a collection of bytes which are not in a human-readable format (.png, .jpeg, .mp4 etc.)

#### IMPORTANT NOTES - 1:

In Python 3 - a string is a sequence of characters, i.e Unicode points; these are an abstract concept, and can't be directly stored on disk. A byte string is a sequence of, unsurprisingly, bytes - things that *can* be stored on disk. The mapping between them is an *encoding* - there are quite a lot of these (and infinitely many are possible) - and you need to know which applies in the particular case in order to do the conversion, since a different encoding may map the same bytes to a different string:

```
>>> b'\xcf\x84o\xcf\x81\xce\xbd\xcf\x82'.decode('utf-16')
```

```
'赫崙濂濂苏'
```

```
>>> b'\xcf\x84o\xcf\x81\xce\xbd\xcf\x82'.decode('utf-8')
```

```
'τορνος'
```

Once you know which one to use, you can use the **.decode()** method of the byte string to get the right character string from it as above. For completeness, the **.encode()** method of a character string goes the opposite way:

```
>>> 'τορνος'.encode('utf-8')  
b'\xcf\x84o\xcf\x81\xce\xbd\xcf\x82'
```

#### IMPORTANT NOTES - 2:

The only thing that can be stored onto a computer's memory are bytes.

To store anything onto a computer's memory, you must first encode it, i.e. convert it to bytes or binary format. For example:

- To store music, you must first encode it using MP3, WAV, etc.
- To store an image, you must first encode it using PNG, JPEG, etc.
- To store text, you must first encode it using ASCII, UTF-8, etc.

MP3, WAV, PNG, JPEG, ASCII and UTF-8 are examples of encodings.

An encoding is a format to represent audio, images, text, etc in bytes.

In Python, a byte string is just that: **a sequence of bytes**. It isn't human-readable. Under the hood, everything must be converted to a byte string before it can be stored onto a computer's memory.

On the other hand, a character string, often just called a "string", is **a sequence of characters**. It is human-readable. A character string can't be directly stored in a computer, it has to be encoded first

(converted into a byte string). There are multiple encodings through which a character string can be converted into a byte string, such as ASCII and UTF-8.

```
>>> 'My Name Is Subhash'.encode('ASCII')
```

The above Python code will encode the string 'My Name Is Subhash' using the encoding ASCII. The result of the above code will be a byte string. If you print it, Python will represent it as `b'My Name Is Subhash'`

Remember, however, that byte strings aren't human-readable, it's just that Python decodes them from ASCII when you print them. In Python, a byte string is represented by a `b`, followed by the byte string's ASCII representation.

A byte string can be decoded back into a character string, if you know the encoding that was used to encode it.

```
b'My Name Is Subhash'.decode('ASCII')
```

The above code will return the original string 'I am a string'.

Encoding and decoding are inverse operations. Everything must be encoded before it can be written to disk, and it must be decoded before it can be read by a human

### **Program: 1**

```
import os
fp = open("test.txt", "w+")
print( fp.name )
print( fp.mode )
print( fp.closed )
fp.write("hello how are you\n");
fp.write("hello how are you\n");
fp.write("hello how are you\n");
print(fp.tell())
print(fp.seek(0))
print(fp.tell())
print(fp.read(2))
print(fp.readlines())
print(fp.tell())
fp.seek(0)
print(fp.read(5))
print(fp.tell( ))
fp.seek(0)
print(fp.readline())
os.rename("test.txt", "ps.txt")
os.remove("ps.txt")
os.mkdir("test")
os.chdir("test")
fp_new = open("input.txt", "w")
fp_new.write("Hello, Welcome to test directory" );
fp_new.close()
print(os.getcwd())
os.mkdir("testinside")
print(os.listdir( ))
```

### **Output:**

```
test.txt
w+
False
54
0
0
he
['llo how are you\n', 'hello how are you\n',
'hello how are you\n']
54
hello
5
hello how are you
/Users/subhash/PYTHON_LEARNING/files/tes
t
['input.txt', 'testinside']
```



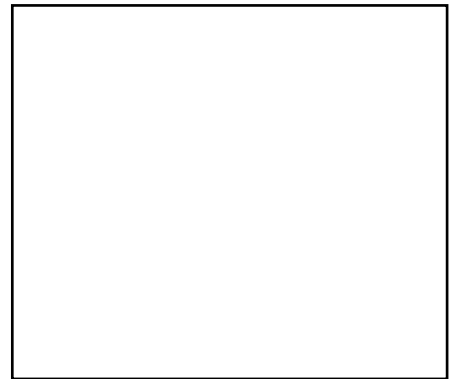
**Program: 2**

```
fp_to_read = open("file_one.txt", "r")
l = fp_to_read.read(1)
str = l

while l != "":
    l = fp_to_read.read(1)
    str = str + l

print(str)

fp_to_read.close()
```

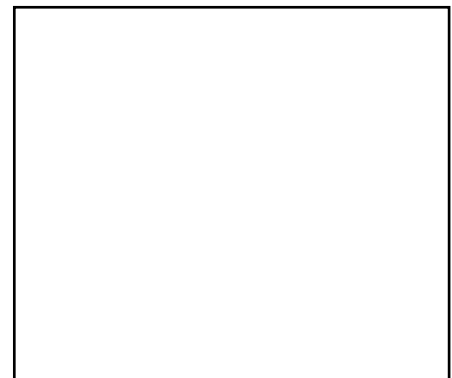
**Output:****Program : 3**

```
fp_to_read = open("file_one.txt", "r")
l = fp_to_read.read(1)
str = l

while l != "":
    l = fp_to_read.readline()
    str = str + l

print(str)

fp_to_read.close()
```

**Output:****Program: 4**

```
fp_to_read = open("file_one.txt", "r")
fp_to_write = open("file_two.txt", "w")

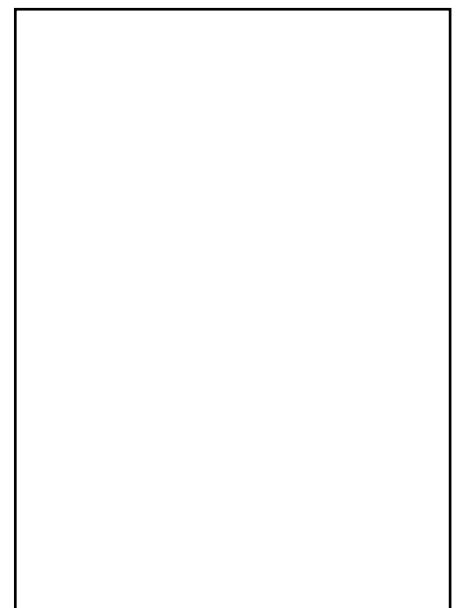
l = fp_to_read.read(1)

while l != "":
    fp_to_write.write(l)
    l = fp_to_read.read(1)

fp_to_read.close()
fp_to_write.close()

fp_to_read = open("file_two.txt", "r")
l = fp_to_read.read(1)
str = l
while l != "":
    l = fp_to_read.read(1)
    str = str + l

print(str)
fp_to_read.close()
```

**Output:**

**Program: 5**

```
fp_to_read = open("file_one.txt", "r")
fp_to_write = open("file_two.txt", "w")

l = fp_to_read.read(1)

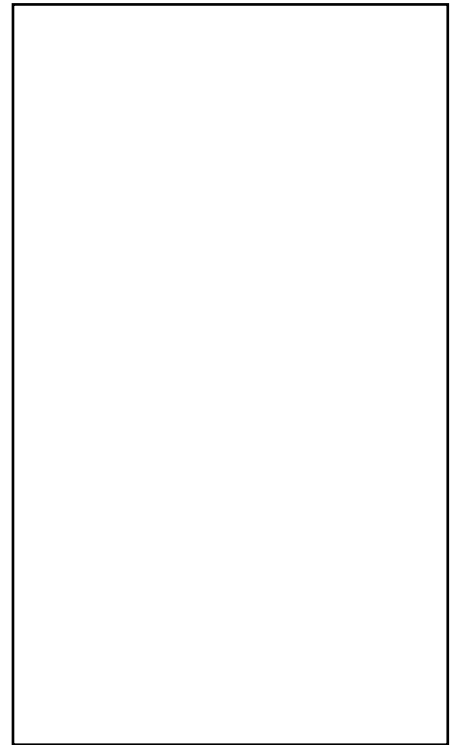
while l != "":
    fp_to_write.write(l)
    l = fp_to_read.readline()

fp_to_read.close()
fp_to_write.close()

fp_to_read = open("file_two.txt", "r")
l = fp_to_read.readline()
str = l
while l != "":
    l = fp_to_read.readline()
    str = str + l

print(str)

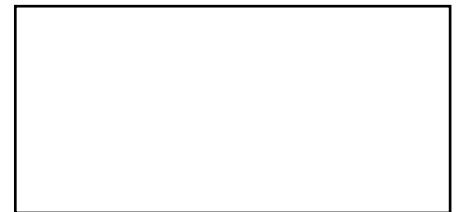
fp_to_read.close()
```

**Output:****Program: 6**

```
fp_to_read = open("file_one.txt", "r")

for line in fp_to_read:
    print(line)

fp_to_read.close()
```

**Output:****Program: 7**

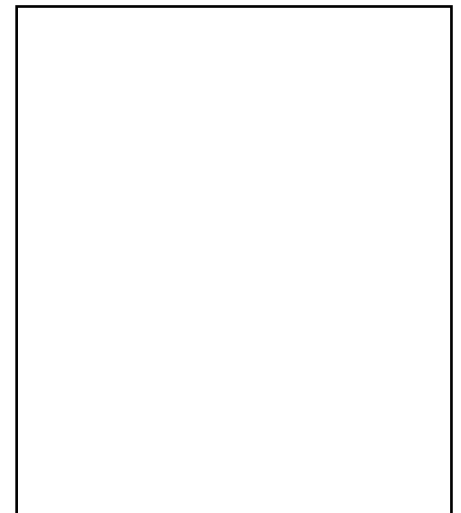
```
fp_one = open("file_one.txt", "r")

line = ""
line = fp_one.readline()

space = 0

while line != "":
    for i in range(0, len(line)):
        if (line[i] == ' '):
            space = space + 1
    line = fp_one.readline()

print("Number of spaces = ", space)
```

**Output:**

**Program: 8**

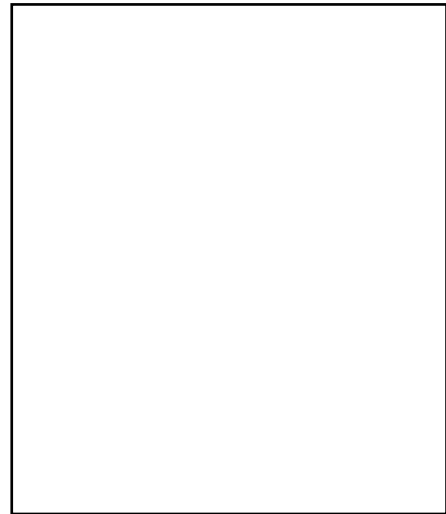
```
fp_one = open("file_one.txt", "r")

line = ""
line = fp_one.readline()
spaces = 0

while line != "":
    for ch in line:
        if( ch == ' '):
            spaces = spaces + 1

    line = fp_one.readline()

print( "Spaces = ", spaces )
```

**Output:****Assignments To Practically Try Out:****Program: 9**

```
import os
fp = open("PPP.png", "r" )
print(fp.read( ))
```

**Program: 10**

```
import os
fp = open("PPP.png", "rb" )
print(fp.read( ))
```

**Program: 11**

```
import os
fp = open("f.bin", "wb")
fp.write(b"Hello\nHello\nHello\n")
fp.close()
fp = open("f.bin", "rb")
print(fp.read( ))
```

**Program: 12**

```
import os
fp = open("f.bin", "wb")
fp.write(b"Hello\nHello\nHello\n")
fp.close()
fp = open("f.bin", "rb")
print(fp.read( ))
```

**Guess The Output:**

1.

```
f = open("fileone.bin", "w+b")
f.write("Hello")
f.close()
```

2.

```
f = open("fileone.bin", "w+b")
f.write(b"Hello")
f.close()
```

3.

```
f = open("fileone.bin", "w+b")
f.write("Hello".encode( ))
f.close()
```

4.

```
f = open("fileone.bin", "w+b")
f.write("Hello".encode())
f.seek(0)
str = f.read( )
print(str)
f.close()
```

5.

```
f = open("fileone.bin", "w+b")
f.write("Hello".encode( ))
f.seek(0)
str = f.read()
print(str.decode())
f.close()
```

6.

```
>>> '하롱칼 濔芬'.encode('utf-8')
```

7.

```
>>> 'τορνος'.encode('utf-8')
```

8.

```
>>> b'\xcf\x84o\xcf\x81\xce\xbd\xcf\x82'.decode('utf-16')
```

9.

```
>>> b'\xcf\x84o\xcf\x81\xce\xbd\xcf\x82'.decode('utf-8')
```

**Let's explore built-in support for file operations:**

```
>>> import os
>>> cwd = os.getcwd()
>>> cwd
'C:\\Users\\Subhash K U\\AppData\\Local\\Programs\\Python\\Python39'
>>> os.path.abspath('equal.py')
'C:\\Users\\Subhash K U\\AppData\\Local\\Programs\\Python\\Python39\\equal.py'
>>> os.path.exists('equal.py')
True
>>> os.path.isdir('C:\\Users\\Subhash K U')
True
>>> os.path.isdir('C:\\Users\\Subhash K
U\\AppData\\Local\\Programs\\Python\\Python39\\equal.py')
False
>>> os.listdir(cwd)
['DLLs', 'Doc', 'equal.py', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.dll',
'python39.dll', 'pythonw.exe', 'Scripts', 'tcl', 'Tools', 'vcruntime140.dll', 'vcruntime140_1.dll']
>>>
```

**Program 13:** Program to recursively walk through a directory, prints the names of all the files, and calls itself recursively on all the directories:

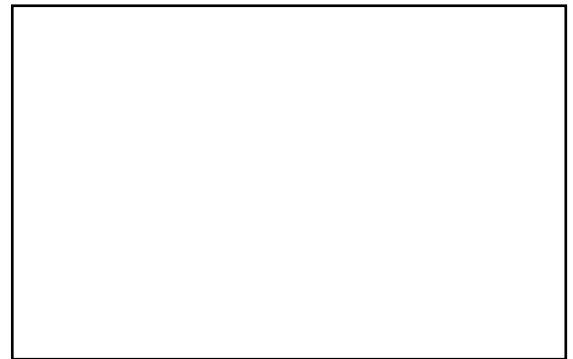
**Output:**

```
import os

def walk(dirname):
    for name in os.listdir(dirname):
        path = os.path.join(dirname, name)

        if os.path.isfile(path):
            print(path)
        else:
            walk(path)

walk('C:\\Users\\Subhash K U\\OneDrive\\Desktop\\Sample')
```



**Serialization and Deserialization:(Pickling and Unpickling)**

#First - create a class and make it a module (call - obj.py )

```
class object:
    def __init__(self):
        self.name = input("Enter name:")
        self.age = input("Enter age:")

    def display(self):
        print("Name = " + self.name )
        print("Age = " + self.age )
```

**Program: 13**

#Second - Create a program to serialize objects (Pickling)  
#pickling.py

```
import obj, pickle

fone = open("jum.bin", "wb")

choice = "yes"

while choice == "yes":
    o = obj.object()
    pickle.dump(o,fone)
    choice = input("Enter one more object? 'yes' or 'no' ?")
```

**Output:**

```
Enter name:subhash
Enter age:25
Enter one more object? 'yes' or 'no' ?yes
Enter name:charan
Enter age:12
Enter one more object? 'yes' or 'no' ?yes
Enter name:archita
Enter age:18
Enter one more object? 'yes' or 'no' ?no
```

**Program: 14****Output:**

#Third - Create a program to de-serialize objects (Unpickling)  
#unpickling.py

```
import pickle, obj

fone = open("jum.bin", "rb")

while(True):
    try:
        obj = pickle.load(fone)
        obj.display()
    except EOFError:
        print("End of file reached")
        break
```

```
Name = subhash
Age = 25
Name = charan
Age = 12
Name = archita
Age = 18
End of file reached
```

## Lesson-10: Working On Strings

### Guess The Output:

**Original String:** s = "Hello Mr.Bean"

```
>>> len(s)                >>> s.count('o')                >>> s + "!"

>>> s[6]                  >>> s.index('b')                >>> min(s)

>>> s[6:10]               >>> 'a' in s                >>> max(s)

>>> 'H' not in s          >>> "Hello".isalpha( )                >>> "Hello!".isalpha( )

>>> "1234".isdigit( )     >>> "1AB2".isdigit()                >>> "Hello".isupper()

>>> "he".islower()        >>> "HELLO".lower()                >>> "hello".upper()

>>> "Hell".find('l')       >>> "Hell".find('g')                >>> "He".replace('H','t')

>>> " Hello ".strip(' ')   >>> "\nHello\n".strip('\n')

>>> "Shah, Rukh, Khan".split(',')

>>> l = [ "Subhash", "Programming", "classes" ]
>>> str = "-".join(l)
>>> str

>>> sone = "Subhash is a programmer"
>>> print(sone[0].isupper())
>>> print(sone[7].isspace())

>>> s = "SuBhAsH"
>>> print(s.swapcase())

>>> sl = [ "Subhash", "Programming", "Classes" ]
>>> str = "+".join(sl)
>>> str
```



**Programming Assignments:**

1. WAP to separate username and domain from a giving email id.
2. WAP to remove the occurrences of letter 'e' from text of a file and rewrite the changed version to another file.
3. Assume a file contains list of names written one below the other. WAP to read the names and update the file by reversing the order of the names.
4. WAP to reverse the words in a string. **Example:** "Subhash Loves India" must become "India Loves Subhash".
5. WAP to emulate inserting a sub-string into a particular position within another string.
6. WAP to count the number of characters and words in a string.
7. WAP to read strings from keyword and store those group of newline terminated strings in a file.
8. WAP to count number of lines, words and characters in a text file.
9. WAP to open a .jpeg file and copy its contents to another file.

## Lesson-12: Classes & Objects

### Classes and Objects

#### Program 1:

```
class classy:

    def setValue(self):
        self.a = 6
        self.b = 7

    def getValue(self):
        return (self.a, self.b)

print("Hello")
c = classy()
c.setValue()
print(c.getValue())
```

#### **Output:**



Hello  
(6, 7)

#### Program 2:

```
class classy:

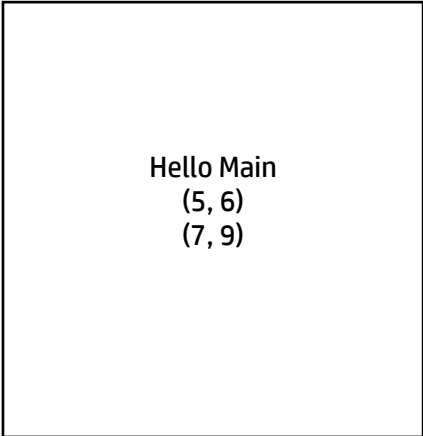
    def setValue(self, a, b):
        self.a = a
        self.b = b

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy()
cobjtwo = classy()
cobjone.setValue(5,6)
cobjtwo.setValue(7,9)
print(cobjone.getValue())
print(cobjtwo.getValue())
```

#### **Output:**



Hello Main  
(5, 6)  
(7, 9)

**Program 3:**

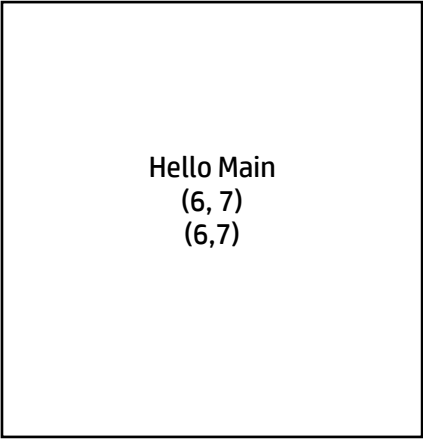
```
class classy:

    def __init__(self):
        self.a = 6
        self.b = 7

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy()
cobjtwo = classy()
print(cobjone.getValue())
print(cobjtwo.getValue())
```

**Output:**

```
Hello Main
(6, 7)
(6,7)
```

**Program 4:**

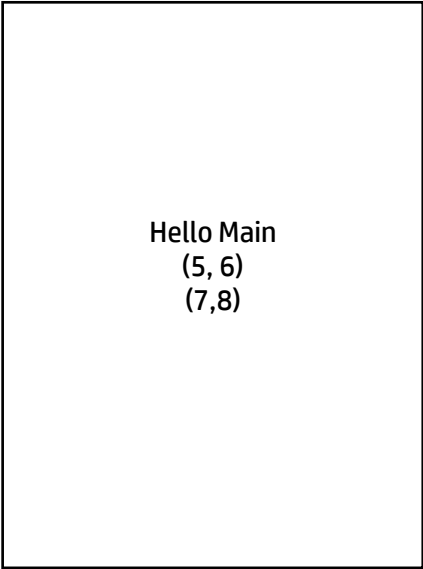
```
class classy:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy(5,6)
cobjtwo = classy(7,8)
print(cobjone.getValue())
print(cobjtwo.getValue())
```

**Output:**

```
Hello Main
(5, 6)
(7,8)
```

**Program 5:**

```
class classy:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy(5,6)
cobjtwo = classy(7,8)

print( cobjone.a, cobjone.b )
print( cobjtwo.a, cobjtwo.b )
```

**Output:**

```
Hello Main
(5, 6)
(7,8)
```

**Program 6:**

```
class classy:

    def __init__(self, a, b):
        self.__a = a
        self.__b = b

print("Hello Main")

cobjone = classy(5,6)

print( cobjone.__a, cobjone.__b )
```

**Output:**

```
Hello Main
Traceback (most recent call last):
  File "classexamplefour.py", line
20, in <module>
    print( cobjone.__a,
cobjone.__b )
AttributeError: 'classy' object has
no attribute '__a'
```

**Program 7:**

```
class classy:

    def __init__(self, a, b):
        self.__a = a
        self.__b = b
        self.same()

    def same(self):
        print("a = ", self.__a)
        print("b = ", self.__b)

print("Hello Main")

cobjone = classy(5,6)
```

**Output:**

```
Hello Main
a = 5
b = 6
```

**Program 8:**

```
class classy:

    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def getValue(self):
        return (self.a, self.b)

print("Hello Main")

cobjone = classy(5,6)

print( cobjone._classy__a, cobjone._classy__b )
```

**Output:**

```
Hello Main
5 6
```

**Program 9:**

```
class classy:

print("Hello main")
cobj = classy( )
print(cobj)
```

**Output:**

```
File "classexampleseven.py", line
4
  print("Hello main")
  ^
IndentationError: expected an
indented block
```

**Program 10:**

```
class classy:
    pass

print("Hello main")
cobj = classy( )
print(cobj)
```

**Output:**

```
Hello main
<__main__.classy object at
0x10d34ed60>
```

**Program 11:**

```
class classy:
    def __str__(self):
        return "I am a __str__ function"

    def __repr__(self):
        return "I am a __repr__ function"

print("Hello main")
cobj = classy()
print(cobj)
```

**Output:**

```
Hello main
I am a __str__ function
```

**NOTE:** The default version of `__str__` method calls the `__repr__` method.

**Program 12:**

```
class classy:

    def __repr__(self):
        return "I am a __repr__ function"

print("Hello main")
cobj = classy()
print(cobj)
```

**Output:**

```
Hello main
I am a __repr__ function
```

**Just remember:**

- The result of `__str__` should be readable.
- The result of `__repr__` should be unambiguous.
- Always add a `__repr__` to your classes. The default implementation for `__str__` just calls `__repr__`, so you get the best of both worlds.

The official Python documentation says `__repr__` is used to find the “official” string representation of an object and `__str__` is used to find the “informal” string representation of an object. The `print` statement and `str()` built-in function uses `__str__` to display the string representation of the object while the `repr()` built-in function uses `__repr__` to display the object. Let us take an example to understand what the two methods actually do.

```
>>> import datetime
>>> today = datetime.datetime.now()
>>> str(today)           #internally calls __str__
'2018-01-12 09:21:58.130922'
>>> repr(today)
'datetime.datetime(2018, 1, 12, 9, 21, 58, 130922)'
```

**Program 13:**

```
class Person:
```

```
    def __init__(self, person_name, person_age):
        self.name = person_name
        self.age = person_age

    def __str__(self):
        return f'Person name is {self.name} and age is {self.age}'

    def __repr__(self):
        return f'Person(name={self.name}, age={self.age})'
```

```
p = Person('Subhash', 35)
```

```
print(p.__str__())
print(p.__repr__())
```

**Output:**

```
Person name is Subhash and age is
35
Person(name=Pankaj, age=35)
```

**Program 14:**

```
class classy:
```

```
    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def __str__(self):
        pass
        return "In __str__ " + str(self.__a) + " and " + str(self.__b)

    def __repr__(self):
        return "In __repr__ " + str(self.__a) + " and " + str(self.__b)
```

```
print("Hello main")
cobj = classy(5,6)
print(cobj)
```

**Output:**

```
Hello main
In __str__ 5 and 6
```

**Program 15:**

```
class Angel:

    a = 10

    def imethod(self):
        self.a = 11

    def get(self):
        print(self.a)

a1 = Angel()
a2 = Angel()

a1.imethod()
a1.get()

a2.get()
```

**Output:**

```
11
10
```

**Program 16:**

```
class Angel:

    a = 10

    @classmethod
    def cmethod(cls):
        cls.a = 11

    def get(self):
        print(self.a)

a1 = Angel()
a2 = Angel()

a1.cmethod()
a1.get()

a2.get()
```

**Output:**

```
11
11
```

**Output:**

```
11
11
```



**Program 17:**

```
class Angel:
```

```
    a = 10
```

```
    @staticmethod
    def smethod(x):
        x = x * 4
        return x
```

```
a1 = Angel()
```

```
a2 = Angel()
```

```
res = a1.smethod(4)
print(res)
```

```
res = Angel.smethod(5)
print(res)
```

**Output:**

```
16
20
```

**Program 18:**

```
class Student:
```

```
    class dob:
```

```
        def __init__(self,date, month, year):
            self.date = date
            self.month = month
            self.year = year
```

```
        def __str__(self):
            return str(self.date) + "/" + str(self.month) + "/" + str(self.year)
```

```
    def __init__(self, name, age, d, m, y ):
        self.name = name
        self.age = age
        self.d_o_b = self.dob(d, m, y)
```

```
    def print_all(self):
        print(self.name)
        print(self.age)
        print(self.d_o_b)
```

```
s1 = Student("subhash", 34, 7, 6, 1985)
s1.print_all()
```

```
s2 = Student("charan", 21, 4, 5, 1995)
s2.print_all()
```

**Output:**

```
subhash
34
7/6/1985
charan
21
4/5/1995
```

**Programming Assignments:**

1. Implement Circle class and find out the area, perimeter and circumference of the circle.
2. Implement a TV class with volume up and down, channel up and down, power on and off, switch to a specific channel etc.

## Lesson-13: Inheritance & Polymorphism

### Inheritance & Polymorphism

#### Program 1:

```
class Teacher:

    def teach(self):
        print("I randomly teach any subject")

    def schoolname(self):
        print("I work in Subhash Programming Classes")

class CTeacher(Teacher):
    pass

class CppTeacher(Teacher):
    pass

c = CTeacher()
c.teach()
c.schoolname()

cpp = CppTeacher()
cpp.teach()
cpp.schoolname()
```

#### **Output:**

```
I randomly teach any subject
I work in Subhash Programming Classes
I randomly teach any subject
I work in Subhash Programming Classes
```

#### Program 2:

```
class Teacher:

    def teach(self):
        print("I randomly teach any subject")

    def schoolname(self):
        print("I work in Subhash Programming Classes")

class CTeacher(Teacher):
    def teach(self):
        print("I teach C Programming")

class CppTeacher(Teacher):
    def teach(self):
        print("I teach C++ Programming")
```

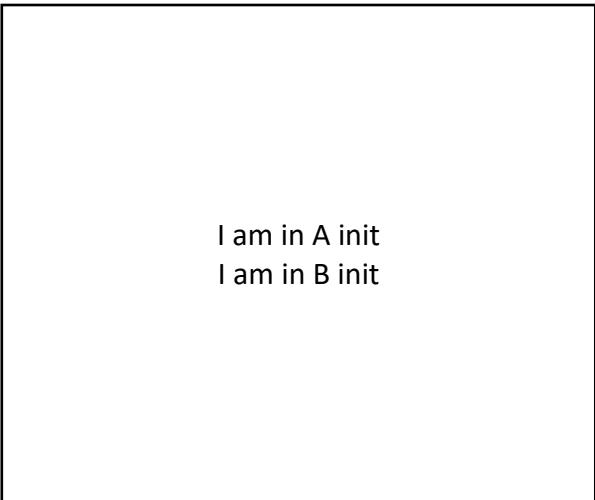
#### **Output:**

```
I teach C programming
I work in Subhash Programming Classes
I teach C++ programming
I work in Subhash Programming Classes
```

```
c = CTeacher()  
c.teach()  
c.schoolname()  
  
cpp = CppTeacher()  
cpp.teach()  
cpp.schoolname()
```

**Program 3:**

```
class A:  
    def __init__(self):  
        print("I am in A init")  
  
class B(A):  
    def __init__(self):  
        print("I am in B init")  
  
a = A()  
b = B()
```

**Output:**

I am in A init  
I am in B init

**Program 4:**

```
class A:  
    def __init__(self):  
        print("I am in A init")  
  
class B(A):  
    def __init__(self):  
        print("I am in B init")  
  
b = B()
```

**Output:**

I am in B init

**Program 5:**

```
class A:
    def __init__(self):
        print("I am in A init")

class B(A):
    def __init__(self):
        print("I am in B init")
        super().__init__()

b = B()
```

**Output:**

```
I am in B init
I am in A init
```

**Program 6:**

```
class A:
    a = 50
    def __init__(self):
        print("I am in A init")
        self.a = 10
        self.b = 20

class B(A):
    b = 30
    def __init__(self):
        print("I am in B init")
        super().__init__()

    def getValue(self):
        print("My value is = ", self.a)
        print("My value is = ", self.b)
        print("My value is = ", B.b)

b = B()
b.getValue()
```

**Output:**

```
I am in B init
I am in A init
My value is = 10
My value is = 20
My value is = 30
```

**Program 7:**

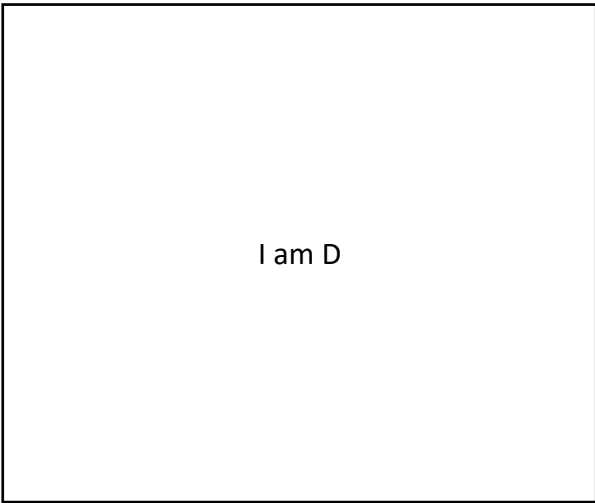
```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    def call(self):
        print("I am D")

d = D()
d.call()
```

**Output:**

I am D

**Program 8:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    pass

d = D()
d.call()
```

**Output:**

I am B

**Program 9:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

class C(A):
    def call(self):
        print("I am C")

class D(C,B):
    pass

d = D()
d.call()
```

**Output:**

I am C

**Program 10:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    pass

class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    pass

d = D()
d.call()
```

**Output:**

I am C

**Program 11:**

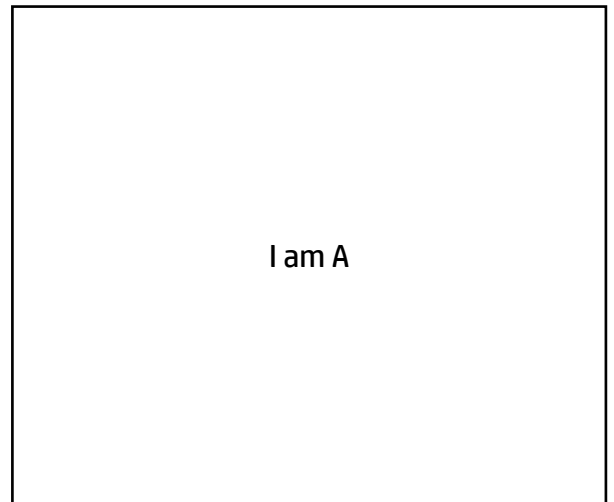
```
class A:
    def call(self):
        print("I am A")

class B(A):
    pass

class C(A):
    pass

class D(B,C):
    pass

d = D()
d.call()
```

**Output:**

I am A

**Program 12:**

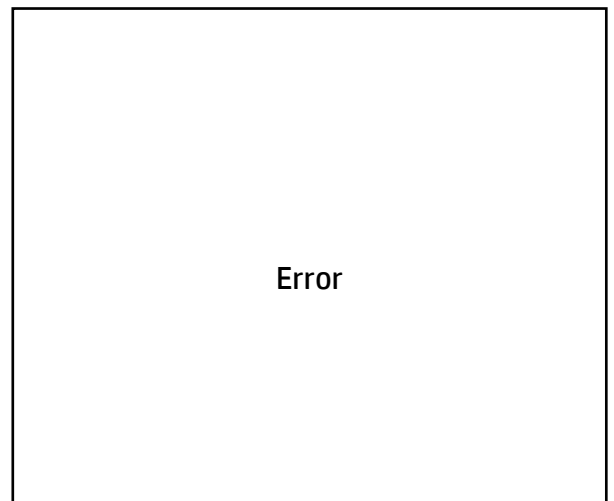
```
class A:
    pass:

class B(A):
    pass

class C(A):
    pass

class D(B,C):
    pass

d = D()
d.call()
```

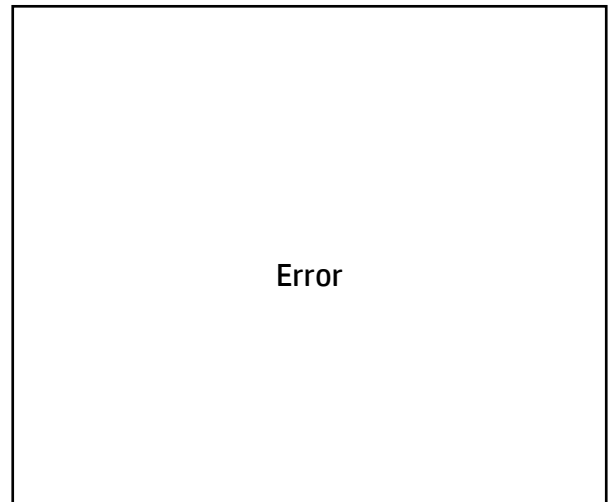
**Output:**

Error



**Program 13:**

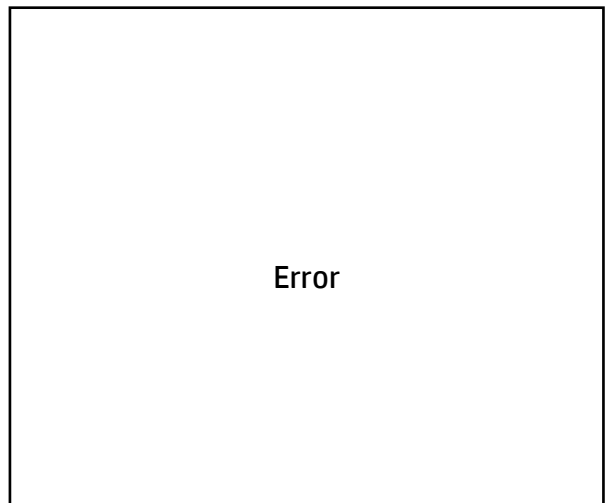
```
class A:  
    pass  
  
class B(A):  
    pass  
  
class C(A):  
    pass  
  
class D(B,C):  
    pass  
  
class E(A):  
    def call(self):  
        print("I am E")  
  
d = D()  
d.call()
```

**Output:**

Error

**Program 14:**

```
class A:  
    pass  
  
class B(A):  
    pass  
  
class C(A):  
    pass  
  
class D(B,C):  
    pass  
  
class E(A):  
    def call(self):  
        print("I am E")  
  
d = D()  
d.call()
```

**Output:**

Error

**Program 15:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    pass

class C(A):
    pass

class E(A):
    def call(self):
        print("I am E")

class D(B,C,E):
    pass

d = D()
d.call()
```

**Output:**

I am E

**Program 16:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

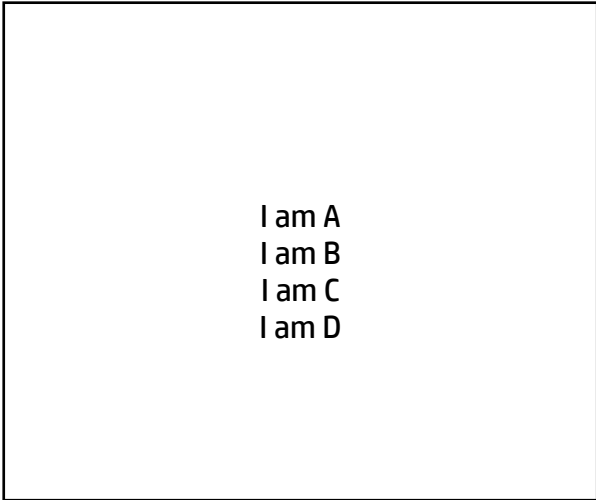
class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    def call(self):
        print("I am D")

def jam(x):
    x.call()

a = A()
b = B()
c = C()
d = D()

jam(a)
jam(b)
jam(c)
jam(d)
```

**Output:**

I am A  
I am B  
I am C  
I am D

**Abstract Classes & Abstract Methods****Program 17:**

```
from abc import *

class Teacher(ABC):

    def schoolname(self):
        print("I teach at Subhash Programming Classes")

    @abstractmethod
    def teach(self):
        pass

class CTeacher(Teacher):

    def teach(self):
        print("I teach C")

class PythonTeacher(Teacher):

    def teach(self):
        print("I teach Python")

t = CTeacher()
t.schoolname()
t.teach()

print()

t = PythonTeacher()
t.schoolname()
t.teach()
```

**Output:**

```
I teach at Subhash Programming Classes
I teach C
```

```
I teach at Subhash Programming Classes
I teach Python
```

**Program 18:**

```
from abc import *  
  
class Teacher(ABC):  
    def schoolname(self):  
        print("I teach at Subhash Programming Classes")  
  
    @abstractmethod  
    def teach(self):  
        pass  
  
class CTeacher(Teacher):  
    def teach(self):  
        print("I teach C")  
  
class pythonTeacher(Teacher):  
    def teach(self):  
        print("I teach Python")  
  
t = Teacher()  
t.schoolname()
```

**Output:**

Error

**Program 19:**

```
from abc import *  
  
class Teacher(ABC):  
    def schoolname(self):  
        print("I teach at Subhash Programming Classes")  
  
    @abstractmethod  
    def teach(self):  
        pass  
  
class CTeacher(Teacher):  
    pass  
  
class PythonTeacher(Teacher):  
    pass  
  
t = CTeacher()  
t = PythonTeacher()
```

**Output:**

Error

**Demonstrating Polymorphism:****Program 20:**

```
from abc import *

class Shape(ABC):

    @abstractmethod
    def draw():
        pass

class Square(Shape):
    def draw(self):
        print("I am a square")

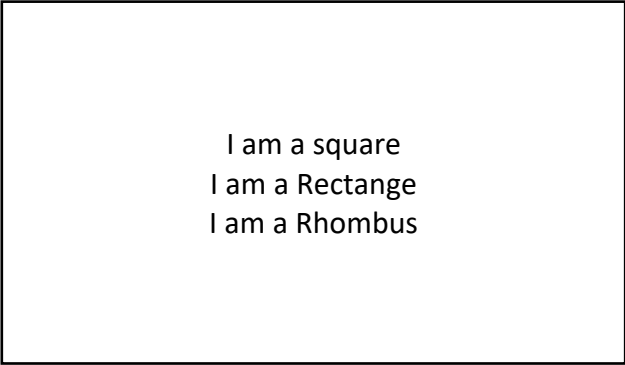
class Rectangle(Shape):
    def draw(self):
        print("I am a Rectangle")

class Rhombus(Shape):
    def draw(self):
        print("I am a Rhombus")

s = Square()
r = Rectangle()
rh = Rhombus()

shapes = [ s, r, rh ]

for i in shapes:
    i.draw()
```

**Output:**

```
I am a square
I am a Rectangle
I am a Rhombus
```

## Lesson-14: Miscellaneous Concepts In Python

### Achieving Method Overloading:

**NOTE:** There is no method overloading in python

#### Program 1:

**#This is a very famous interview question.**

```
class MyMath:
    def sum(self, a=None, b=None, c=None):
        if(a != None and b != None and c != None):
            return (a + b + c)

        elif(a != None and b != None ):
            return (a + b)

m = MyMath()
print(m.sum(1,2,3))
print(m.sum(1.5, 2.5))
```

**Output:**

6  
4.0

### Operator Overloading

#### Program 2:

```
class classy:

    def __init__(self, a, b ):
        self.__a = a
        self.__b = b

    def __add__(self, obj ):
        ta = self.__a + obj.__a
        tb = self.__b + obj.__b
        cc = classy(ta, tb)
        return cc

    def __str__(self):
        return "a = " + str(self.__a) + " b = " + str(self.__b)

cobjone = classy(1,2)
cobjtwo = classy(1,2)
cobjthree = cobjone + cobjtwo
print(cobjthree)
```

**Output:**

a = 2  
b = 4

**Program 3:**

```
class classy:
```

```
    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def __sub__(self, obj):
        return classy((self.__a - obj.__a), (self.__b - obj.__b))

    def __str__(self):
        return "a = " + str(self.__a) + " b = " + str(self.__b)
```

```
cobjone = classy(1,2)
cobjtwo = classy(1,2)
cobjthree = cobjone - cobjtwo
print(cobjthree)
```

**Output:**

```
a = 0
b = 0
```

**Program 4:**

```
class classy:
```

```
    def __init__(self, a, b):
        self.__a = a
        self.__b = b

    def __mul__(self, obj):
        return classy((self.__a * obj.__a), (self.__b * self.__b))

    def __str__(self):
        return "a = " + str(self.__a) + " b = " + str(self.__b)
```

```
cobjone = classy(1,2)
cobjtwo = classy(1,2)
cobjthree = cobjone * cobjtwo
print(cobjthree)
```

**Output:**

```
a = 1
b = 4
```

**Program 5:**

```
class classy:
```

```
    def __init__(self, a, b ):
        self.__a = a
        self.__b = b

    def __neg__(self ):
        return classy(-(self.__a ), -(self.__b))

    def __str__(self):
        return "a = " + str(self.__a) + " b = " + str(self.__b)
```

```
cobjone = classy(1,2)
cobjtwo = -cobjone
print(cobjtwo)
```

**Output:**

```
a = -1
b = -2
```

**Program 6:**

```
class classy:
```

```
    def __init__(self, a ):
        self.__a = a

    def __lt__(self, obj):
        return self.__a < obj.__a

    def __le__(self, obj):
        return self.__a <= obj.__a

    def __ge__(self, obj):
        return self.__a >= obj.__a

    def __gt__(self, obj):
        return self.__a > obj.__a

    def __ne__(self,obj):
        return self.__a != obj.__a

    def __eq__(self,obj):
        return self.__a == obj.__a
```

```
cobjone = classy(1)
cobjtwo = classy(2)
```

```
print(cobjone < cobjtwo)
print(cobjone <= cobjtwo)
```

**Output:**

```
True
True
False
False
False
True
```



```
print(cobjone >= cobjtwo)
print(cobjone > cobjtwo)
cobjthree = classy(0)
cobjfour = classy(0)
print(cobjthree != cobjfour)
print(cobjthree == cobjfour)
```

### **Command Line Arguments:**

#### **Program 7:**

```
import sys

print("The length of CLA is = ", len(sys.argv))
print("The values of CLA are:")

for value in sys.argv:
    print(value)
```

#### **Output:**

```
C:\> py -3 clacla.py subhash loves india
The length of CLA is = 4
The values of CLA are:
clacla.py
subhash
loves
india
C:\>
```

#### **Program 8:**

```
import sys

print("Adding two numbers through CLA")
print(int(sys.argv[1]) + int(sys.argv[2]))
```

#### **Output:**

```
C:\> py -3 clacla.py 2 3
Adding two numbers through CLA
5
C:\>
```

### **Assignments:**

Write a program to accept two files names as input through command line and do a copy operation.

**Program 9:**

```
import argparse

parser = argparse.ArgumentParser(description="This program will add two floats")
parser.add_argument("numone", type=float, help = "Enter a float type number")
parser.add_argument("numtwo", type=float, help = "Enter a float type number")
args = parser.parse_args( )

result = args.numone + args.numtwo

print(result)
```

**Output:**

```
C:\> py -3 clacla.py 1 2
3
C:\>
```

**Program 10:**

```
import argparse

parser = argparse.ArgumentParser(description="This program will add three numbers")
parser.add_argument("num", nargs=3, type=int, help="Pass two values")
args = parser.parse_args( )

result = int(args.num[0]) + int(args.num[1]) + int(args.num[2])

print(result)
```

**Output:**

```
C:\> py -3 clacla.py 1 2 3
6
C:\>
```

**Program 11:**

```
import argparse

parser = argparse.ArgumentParser(description="This program will accept all arguments")
```

```
parser.add_argument("values", nargs = '*', help="Enter anything")
args = parser.parse_args()
```

```
for val in args.values:
    print(val)
```

**Output:**

```
C:\> py -3 clacla.py Subhash 10 2.3 [1,2,3]
subhash
10
2.3
[1,2,3]
C:\>
```

**Program 12:**

```
import argparse
```

```
parser = argparse.ArgumentParser(description="This program will accept 1 or more arguments")
parser.add_argument("values", nargs='+', help="Enter one or more arguments")
args = parser.parse_args()
```

```
for val in args.values:
    print(val)
```

**Output:**

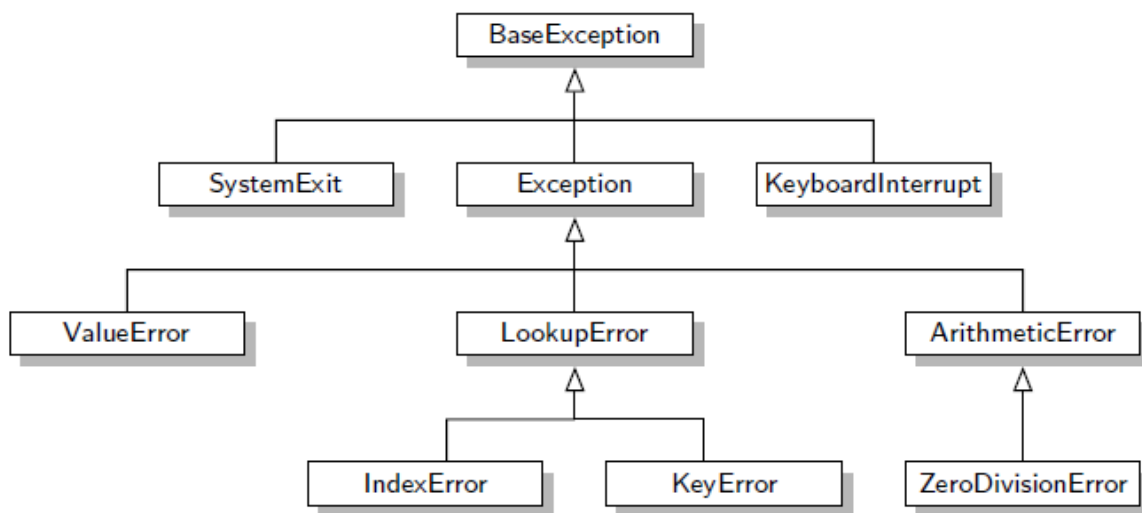
```
C:\> py -3 clacla.py 1 subhash [1,2,3]
1
subhash
[1,2,3]
C:\>
```

## Lesson-15: Exception Handling

**What is the difference b/w Compile-Time Error, Logical Error, Run-Time Error:**

**NOTE:**

- Run-time errors are called exceptions. The process of handling run-time errors is called **Exception Handling**.
- The advantage of Exception Handling is the separation of exception detection code from the exception handling code.



**Notice The Output:**

```
1)
>>> n, x = 5, 6, "Jack"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack (expected 2)
>>>
```

```
2)
>>> l = [1,2,3]
>>> l[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>>
```

3)

&gt;&gt;&gt; b = 10

&gt;&gt;&gt; a

Traceback (most recent call last):

File "&lt;stdin&gt;", line 1, in &lt;module&gt;

**NameError:** name 'a' is not defined

&gt;&gt;&gt;

4)

&gt;&gt;&gt; import subhash

Traceback (most recent call last):

File "&lt;stdin&gt;", line 1, in &lt;module&gt;

**ModuleNotFoundError:** No module named 'subhash'

&gt;&gt;&gt;

5)

&gt;&gt;&gt; 2 + '3'

Traceback (most recent call last):

File "&lt;stdin&gt;", line 1, in &lt;module&gt;

**TypeError:** unsupported operand type(s) for +: 'int' and 'str'

&gt;&gt;&gt;

**Program 1:**

import math

#Enter negative number to check what happens

num = int(input('Enter a number to compute factorial of:'))

print(math.factorial(num))

**Output:**

C:\&gt; py -3 Handling\_Exception.py

Enter a number to compute factorial of: -5

Traceback (most recent call last):

File "Handling\_Exception.py", line 6, in

&lt;module&gt;

print(math.factorial(num))

**ValueError:** factorial() not defined for negative values

C:\&gt;

**Program 2:**

import math

#Enter negative number to check what happens

num = int(input('Enter number to compute factorial of:'))

**try:**

print(math.factorial(num))

**except ValueError:**

print('Cannot compute the factorial of negative numbers')

**Output:**

C:\&gt; py -3 Handling\_Exception.py

Enter number to compute factorial of:-5

Cannot compute the factorial of negative numbers

C:\&gt;

**Program 3:****Output:****import math**

```
#Enter negative number to check what happens
num = int(input('Enter number to compute factorial of:'))
valid_input = False

while not valid_input:
    try:
        print(math.factorial(num))
        valid_input = True
    except ValueError:
        print('Cannot compute the factorial of negative numbers')
        num = int(input('Please re-enter: '))
```

```
C:\> py -3 Handling_Exception.py
Enter number to compute factorial of:-5
Cannot compute the factorial of negative
numbers
Please re-enter: -5
Cannot compute the factorial of negative
numbers
Please re-enter: 5
120
C:\>
```

**Program 4:**

```
def getMonth( ):
    month = int(input("Enter current month(1-12):"))

    if month < 1 or month > 12:
        raise ValueError

    return month

valid = False
month_name = ("January","Februry", "March", "April", "May", "June", "July", "August", "September",
"October", "November", "December")
while not valid:
    try:
        month = getMonth( )
        print("The month you entered is ", month_name[month - 1])
        valid = True
    except ValueError:
        print('Invalid Month Entry\n')
```

**Output:**

```
C:\> py -3 Handling_Exception.py
Enter current month(1-12):13
Invalid Month Entry

Enter current month(1-12):1a
Invalid Month Entry

Enter current month(1-12):2
The month you entered is  Februry
C:\>
```

**Program 5:**

```
def getMonth():
    month = int(input("Enter current month(1-12):"))

    if month < 1 or month > 12:
        raise ValueError("Invalid Month Value")
    return month

valid = False
month_name = ("January", "February", "March", "April", "May", "June", "July", "August", "September",
"October", "November", "December")
while not valid:
    try:
        month = getMonth()
        print("The month you entered is ", month_name[month - 1])
        valid = True
    except ValueError as err_msg:
        print(err_msg)
```

**Output:**

```
C:\> py -3 Handling_Exception.py
Enter current month(1-12):1
The month you entered is January
C:\> py -3 Handling_Exception.py
Enter current month(1-12):23
Invalid Month Value
Enter current month(1-12):1a
invalid literal for int() with base 10: '1a'
Enter current month(1-12):1
The month you entered is January
C:\>
```

**Program 6:**

#This is a program for self-analysis. Reading and understanding this code slowly and steadily is worth 10 classes of someone explaining this code to you. So try to go through this code and analyze it line by line. Make use of pen and a paper.

```
def getFile():
    input_file_opened = False
    while not input_file_opened:
        try:
            file_name = input("Enter input file name (with extension): ")
            input_file = open(file_name, "r")
            input_file_opened = True
        except IOError:
            print("Input file not found - please re-renter\n")

    return (file_name, input_file)

def countWords(input_file, search_words):
```

```
space = ''
num_occurrences = 0
word_delimiters = (space, "'", ";", ":", ".", "\n", ",", "(", ")")

search_word_len = len(search_word)

for line in input_file:
    end_of_line = False
    #convert line read to all lower case chars
    line = line.lower()

    #scan line until end of line reached
    while not end_of_line:
        try :
            #search for word in current line
            index = line.index(search_word)

            #if word at start of line followed by a delimiter
            if index == 0 and line[search_word_len] in word_delimiters:
                found_search_word = True
            #if search word within line, check chars before/after
            elif line[index-1] in word_delimiters and line[index + search_word_len] in word_delimiters:
                found_search_word = True

            #if found within other letters, then not search word
            else:
                found_search_word = False

            #if search word found, increment count
            if found_search_word:
                num_occurrences = num_occurrences + 1

            #reset line to rest of line following search word
            line = line[index + search_word_len:len(line)]

        except ValueError:
            end_of_line = True

    return num_occurrences

#start program

print("This program will display the number of occurrences of a specified word within a given text file\n")

#open file to search

file_name, input_file = getFile()

#get search word

search_word = input("Enter word to search: ")
search_word = search_word.lower()

#count all occurrences of search word

num_occurrences = countWords(input_file, search_word)

#display results

if num_occurrences == 0:
```



```
print("No occurrences of word", "" + search_word + "", "found in file", file_name)
else:
    print("The word", "" + search_word + "", "occurs", num_occurrences, "times in file", file_name)
```

**Output:**

```
C:\> py -3 Handling_Exception.py
This program will display the number of occurrences of a specified word within a given text file

Enter input file name (with extension): hello.txt
Enter word to search: the
The word 'the' occurs 3 times in file hello.txt
C:\>
```

**Program 7:**

```
class UserDefinedException(Exception): #mandatory to be inherited
    def __init__(self, message):
        self.message = message

try:
    s = input("Enter name:")
    if(s == " "):
        raise UserDefinedException("No Empty String Allowed")
    else:
        print(s)

except UserDefinedException as msg:
    print("Error Message:", msg)
```

**Output:**

```
C:\> py -3 Handling_Exception.py
Enter name:
Error Message: No Empty String Allowed
C:\>
```

**Guess The Output:**

1.

```
try:
    a = 5
    b = 5 / 0
except ZeroDivisionError:
    print("ZDE")
except BaseException:
    print("BE")
except Exception:
```

```
print("E")
```

2.

```
try:
```

```
    a = 5
```

```
    b = 5 / 0
```

```
except BaseException:
```

```
    print("BE")
```

```
except Exception:
```

```
    print("E")
```

```
except ZeroDivisionError:
```

```
    print("ZDE")
```

3.

```
try:
```

```
    a = 5
```

```
    b = 5 / 0
```

```
except ZeroDivisionError:
```

```
    print("ZDE")
```

```
except BaseException:
```

```
    print("BE")
```

```
except Exception:
```

```
    print("E")
```

```
else:
```

```
    print("ELSE")
```

4.

```
try:
```

```
    a = 5
```

```
    b = 5 / 5
```

```
except ZeroDivisionError:
```

```
    print("ZDE")
```

```
except BaseException:
```

```
    print("BE")
```

```
except Exception:
```

```
    print("E")
```

```
else:
```

```
    print("ELSE")
```

5.

```
try:
    a = 5
    b = 5 / 0
except (BaseException, Exception, ZeroDivisionError):
    print("An Exception Occured")
```

6.

```
try:
    a = 5
    b = 5 / 5
except:
    print("Hi")
else:
    print("Hello")
```

7.

```
try:
    a = 5
    b = 5 / 5
else:
    print("Hello")
```

8.

```
try:
    a = 5
    b = 5 / 0
except:
    print("Hi")
else:
    print("Hello")
finally:
    print("Namaste")
```

9.

```
try:
    a = 5
    b = 5 / 5
except:
    print("Hi")
else:
    print("Hello")
finally:
    print("Namaste")
```

10.

try:

    a = 5

    b = 5 / 0

finally:

    print("Namaste")

## Lesson-16: Threads

### Threads:

What is a thread?

What is Multithreading?

What is the difference between a 'process' and a 'thread'?

What are the uses of Threads?

### IMPORTANT NOTE:

The Thread class represents an activity that is run in a separate thread of control. There are two ways to specify the activity: **by passing a callable object to the constructor**, or **by overriding the run( ) method in a subclass**. No other methods (except for the constructor) should be overridden in a subclass. In other words, *only* override the `__init__( )` and `run( )` methods of this class.

### Program 1:

#### #thread creation method - 1

```
import threading

print(threading.current_thread( ))
print(threading.main_thread( ))
print(threading.current_thread( ).getName( ))

def thread_fun(str):
    for i in range(5):
        print(str)

    print(threading.current_thread( ))
    print(threading.current_thread( ).getName())
    threading.current_thread( ).setName("My First Thread")
    print(threading.current_thread().getName( ))

t = threading.Thread(target=thread_fun, args = ("Hello Subhash",))

t.start( )    #internally calls Thread.run( )
```

### Output:

```
<_MainThread(MainThread, started
140735275934464)>
<_MainThread(MainThread, started
140735275934464)>
MainThread
Hello Subhash
Hello Subhash
Hello Subhash
Hello Subhash
Hello Subhash
<Thread(Thread-1, started
4447309824)>
Thread-1
My First Thread
```

**Program 2:****#thread creation method - 2**

```
import threading

class MyThreadClass(threading.Thread):
    def __init__(self,str):
        super( ).__init__( )
        self.str = str

    def run(self):
        print("Entering into:", end=' ')
        threading.current_thread( ).setName("My Second Thread")
        print(threading.current_thread().getName( ))
        self.my_thread( )

    def my_thread(self):
        for i in range(5):
            print(self.str)

print("Here is my ", threading.current_thread( ).getName( ))

thread_one = MyThreadClass("Hello Subhash")
thread_one.start( )
```

**Output:**

```
Here is my MainThread
Entering into: My Second Thread
Hello Subhash
Hello Subhash
Hello Subhash
Hello Subhash
Hello Subhash
```

**Program 3:****#thread creation method - 3**

```
import threading

#thread creation method - 3

class MyThreadClass:
    def __init__(self,str):
        self.str = str

    def my_thread(self, str):
        print("Entering into:", end=' ')
        threading.current_thread().setName("My Second Thread")
        print(threading.current_thread().getName())
        for i in range(5):
            print(self.str, ", ", str)

print("Here is my ", threading.current_thread().getName())
```

**Output:**

```
Here is my MainThread
Entering into: My Second Thread
Hello Subhash , How Are You?
Hello Subhash , How Are You?
Hello Subhash , How Are You?
Hello Subhash , How Are You?
Hello Subhash , How Are You?
```

```
thread_obj = MyThreadClass("Hello Subhash")
```

```
thread = threading.Thread(target=thread_obj.my_thread, args=("How Are You?"),)
thread.start( )
```

**Program 4:**

```
import threading
```

**#Creating multiple threads**

```
class MyThreadClass:
```

```
    def my_thread(self, str):
        print("Entering into:", end=' ')
        threading.current_thread().setName(str[1])
        print(threading.current_thread().getName())
        for i in range(5):
            print(str[0])
```

```
print("Here is my ", threading.current_thread().getName())
```

```
thread_obj_one = MyThreadClass()
```

```
thread_obj_two = MyThreadClass()
```

```
thread_one = threading.Thread(target=thread_obj_one.my_thread, args=(["Hello Subhash, How Are You?", "Second Thread"],))
```

```
thread_two = threading.Thread(target=thread_obj_two.my_thread, args=(["I Am Awesome, Thank You", "Third Thread"],))
```

```
thread_one.start( )
```

```
thread_two.start( )
```

**Output:**

```
Here is my MainThread
Entering into: Second Thread
Hello Subhash, How Are You?
Hello Subhash, How Are You?
Hello Subhash, How Are You?
Hello Subhash, How Are You?
Hello Subhash, How Are You?
Entering into: Third Thread
I Am Awesome, Thank You
I Am Awesome, Thank You
I Am Awesome, Thank You
I Am Awesome, Thank You
I Am Awesome, Thank You
```

**Program 5:**

```
import threading
import time
#Problem With Multithreading

class TakeMyPieceOfCake:

    def __init__(self,piece_available):
        self.piece_available = piece_available

    def take_my_cake(self, piece_needed):

        threading.current_thread().setName(piece_needed[1])
        if self.piece_available >= piece_needed[0]:
            print( "{0} piece given to {1}".format(piece_needed[0],
threading.current_thread().getName()))
            time.sleep(3)
            self.piece_available = self.piece_available - piece_needed[0]
        else:
            print("No more cake pieces available")

print("Here is my ", threading.current_thread().getName())

give_to_person = TakeMyPieceOfCake(1)

thread_one = threading.Thread(target=give_to_person.take_my_cake, args=([1,"Second Thread"],))
thread_two = threading.Thread(target=give_to_person.take_my_cake, args=([1,"Third Thread"],))

thread_one.start( )
thread_two.start( )
```

**Output:**

Here is my MainThread  
1 piece given to Second Thread  
1 piece given to Third Thread



**Program 6:**

```
import threading
import time
#Solving Multithreading With 'Lock'

class TakeMyPieceOfCake:

    def __init__(self,piece_available):
        self.piece_available = piece_available
        self.lock_it = threading.Lock( )

    def take_my_cake(self, piece_needed):
        self.lock_it.acquire( )
        threading.current_thread( ).setName(piece_needed[1])
        if self.piece_available >= piece_needed[0]:
            print( "{0} piece given to {1}".format(piece_needed[0], threading.current_thread(
).getName( )))
            time.sleep(3)
            self.piece_available = self.piece_available - piece_needed[0]
        else:
            print("No more cake pieces available")

        self.lock_it.release( )

print("Here is my ", threading.current_thread().getName())

give_to_person = TakeMyPieceOfCake(1)

thread_one = threading.Thread(target=give_to_person.take_my_cake, args=([1,"Second Thread"],))
thread_two = threading.Thread(target=give_to_person.take_my_cake, args=([1,"Third Thread"],))

thread_one.start( )
thread_two.start( )
```

**Output:**

Here is my MainThread  
1 piece given to Second Thread  
No more cake pieces available

**Program 7:**

```
# Demonstrating 'Deadlock' problem

import threading

#Let us take two locks

lock_one = threading.Lock( )
lock_two = threading.Lock( )

def TakeBook( ):
    lock_one.acquire( )
    print("Locked Library Database")
    print("Checking for book availability")
    lock_two.acquire()
    print("Checking for number of books available")
    lock_two.release()
    lock_one.release()
    print("Book Issued")

def ReturnBook( ):
    lock_two.acquire()
    print("Locked Books Counter")
    print("Updating number of books available")
    lock_one.acquire()
    print("Checking for book availability")
    lock_one.release()
    lock_two.release()
    print("Book Returned")

#Creating two threads

person_taking_book = threading.Thread(target=TakeBook)
person_returning_book = threading.Thread(target=ReturnBook)
person_taking_book.start( )
person_returning_book.start( )
```

**Output:**

Locked Library Database  
Checking for book availability  
Locked Books Counter  
Updating number of books  
available

(Continues to wait - Deadlock)

**Program 8:**

```
# Demonstrating possible solution for 'Deadlock' problem

import threading

#Let us take two locks

lock_one = threading.Lock()
lock_two = threading.Lock()

def TakeBook():
    lock_one.acquire()
    print("Locked Library Database")
    print("Checking for book availability")
    lock_two.acquire()
    print("Checking for number of books available")
    lock_two.release()
    lock_one.release()
    print("Book Issued")

def ReturnBook():
    lock_one.acquire()
    print("Locked Books Counter")
    print("Updating number of books available")
    lock_two.acquire()
    print("Checking for book availability")
    lock_two.release()
    lock_one.release()
    print("Book Returned")

#Creating two threads

person_taking_book = threading.Thread(target=TakeBook)
person_returning_book = threading.Thread(target=ReturnBook)
person_taking_book.start()
person_returning_book.start()
```

**Output:**

```
Locked Library Database
Checking for book availability
Checking for number of books
available
Book Issued
Locked Books Counter
Updating number of books
available
Checking for book availability
Book Returned
```

**Program 9:****#Producer Consumer - Bad Method**

```
import threading
import time
class Producer:

    def __init__(self):
        self.l = [ ]
        self.production_done = False
        self.i = 0

    def produce(self):
        while True:
            if self.production_done == False:

                for self.i in range(100):
                    self.l.append(self.i)
                    print("Another item produced...")
                    self.production_done = True
            if( self.i == 100 ):
                break;

class Consumer:

    def __init__(self, producer_handle):
        self.producer_handle = producer_handle

    def consume(self):
        while True:
            if self.producer_handle.production_done == True:
                print("Item Consumed")
                print(self.producer_handle.l)
                self.producer_handle.production_done = False;
            if(self.producer_handle.i == 100):
                break;

producer_obj = Producer()
consumer_obj = Consumer(producer_obj)

tone = threading.Thread(target=producer_obj.produce)
ttwo = threading.Thread(target=consumer_obj.consume)

tone.start( )
ttwo.start( )
```

**Output:**

```
Another item produced...
Item Consumed
[0]
Another item produced...
Item Consumed
[0, 54]
Another item produced...
Item Consumed
[0, 54, 48]
Another item produced...
Item Consumed
[0, 54, 48, 54]
Another item produced...
Item Consumed
[0, 54, 48, 54, 5]
Another item produced...
Item Consumed
[0, 54, 48, 54, 5, 85]
Another item produced...
Item Consumed
[0, 54, 48, 54, 5, 85, 66]
Another item produced...
Item Consumed
[0, 54, 48, 54, 5, 85, 66, 71]
```

**Program 10:****Output:****#Producer Consumer - Better Than Bad Method - Yet, Bad Method**

```
import threading
import time
class Producer:

    def __init__(self):
        self.l = [ ]
        self.production_done = False
        self.i = 0

    def produce(self):
        for self.i in range(11):
            time.sleep(1)
            if self.production_done == False:
                self.l.append(self.i)
                print("Another item produced...")
                self.production_done = True
            if(self.i == 10):
                time.sleep(1)
                self.i = 11

class Consumer:

    def __init__(self, producer_handle):
        self.producer_handle = producer_handle

    def consume(self):
        while self.producer_handle.i <= 10:
            if self.producer_handle.production_done == True:
                print("Item Consumed")
                print(self.producer_handle.l)
                self.producer_handle.production_done = False;
                time.sleep(1)

producer_obj = Producer()
consumer_obj = Consumer(producer_obj)

tone = threading.Thread(target=producer_obj.produce)
ttwo = threading.Thread(target=consumer_obj.consume)

tone.start()
ttwo.start()
```

```
Another item produced...
Item Consumed
[0]
Another item produced...
Item Consumed
[0, 1]
Another item produced...
Item Consumed
[0, 1, 2]
Another item produced...
Item Consumed
[0, 1, 2, 3]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7, 8]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

**Program 11:****Output:****#Producer Consumer - Good Method, But, Not Great Method**

```

import threading
import time
class Producer:

    def __init__(self):
        self.l = [ ]
        self.condition_variable = threading.Condition( )

    def produce(self):

        for i in range(11):
            self.condition_variable.acquire( )
            self.l.append(i)
            print("Another item produced...")
            self.condition_variable.notify()
            self.condition_variable.release()
            time.sleep(2)

class Consumer:

    def __init__(self, producer_handle):
        self.producer_handle = producer_handle

    def consume(self):

        for i in range(11):
            time.sleep(1)
            self.producer_handle.condition_variable.acquire( )
            self.producer_handle.condition_variable.wait(timeout=0)
            print("Item Consumed")
            print(self.producer_handle.l)
            self.producer_handle.condition_variable.release( )
            time.sleep(1)

producer_obj = Producer()
consumer_obj = Consumer(producer_obj)

tone = threading.Thread(target=producer_obj.produce)
ttwo = threading.Thread(target=consumer_obj.consume)

tone.start()
ttwo.start()

```

```

Another item produced...
Item Consumed
[0]
Another item produced...
Item Consumed
[0, 1]
Another item produced...
Item Consumed
[0, 1, 2]
Another item produced...
Item Consumed
[0, 1, 2, 3]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7, 8]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Another item produced...
Item Consumed
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```

**Program 12:****#Producer Consumer - Great Method, Can Be Improved**

```
import threading
import time
import queue

class Producer:

    def __init__(self):
        self.q = queue.Queue( )

    def produce(self):

        for i in range(11):
            self.q.put(i)
            print("Another item produced...")
            time.sleep(1)

class Consumer:

    def __init__(self, producer_handle):
        self.producer_handle = producer_handle

    def consume(self):

        for i in range(11):
            print("Item Consumed: ", self.producer_handle.q.get(i))

producer_obj = Producer( )
consumer_obj = Consumer(producer_obj)

tone = threading.Thread(target=producer_obj.produce)
ttwo = threading.Thread(target=consumer_obj.consume)

tone.start( )
ttwo.start( )
```

**Output:**

```
Another item produced...
Item Consumed: 0
Another item produced...
Item Consumed: 1
Another item produced...
Item Consumed: 2
Another item produced...
Item Consumed: 3
Another item produced...
Item Consumed: 4
Another item produced...
Item Consumed: 5
Another item produced...
Item Consumed: 6
Another item produced...
Item Consumed: 7
Another item produced...
Item Consumed: 8
Another item produced...
Item Consumed: 9
Another item produced...
Item Consumed: 10
```

## Lesson-17: Client-Server Programming In Python

- What do you mean by 'protocol' in networking (Ex: HTTP, TCP, IP, DHCP etc.)?
- What is an OSI Layer?
- What is the difference between TCP/IP and UDP?
- What is a 'Host' and 'Software Port' In Networking?
- What is address family IPV4 and IPV6?
- What is loopback address?

### A TCP/IP SERVER:

#### Program tcp-server.py:

```
import socket

#server name and port number

host = '127.0.0.1'
port = 2000

#creating a socket as a communication medium
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#let the clients know who and where the server is bound to (connect to)
s.bind((host,port))

#how many clients is this server ready to respond
s.listen(1)

c, addr = s.accept()

print(type(c))
print(type(addr))
print("My Client Is From : " , str(addr))

#send message to client
c.send(b"Hello Buddy! Thanks For Connecting ! Bye !")

c.close()
```

#### **Output:**

```
<class 'socket.socket'>
<class 'tuple'>
My Client Is From : ('127.0.0.1',
49305)
```



**A TCP/IP CLIENT:****Program tcp-client.py****Output:**

```
import socket

#need server name and port number to connect
host = '127.0.0.1'
port = 2000

#creating a socket as a communication medium
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#connect it to the server who is waiting to accept
s.connect((host,port))

message = s.recv(1024)

while message:
    print("Message From Server :", message.decode( ))
    message = s.recv(1024)

s.close( )
```

Message From Server : Hello  
Buddy! Thanks For Connecting !  
Bye !

**A UDP SERVER:****Program udp-server.py**

```
import socket
```

```
localIP  = "127.0.0.1"
```

```
localPort = 20001
```

```
bufferSize = 1024
```

```
msgToClient = b"Hello UDP Client"
```

```
# Create a datagram socket
```

```
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
```

```
# Bind to address and ip
```

```
UDPServerSocket.bind((localIP, localPort))
```

```
print("UDP server up and listening")
```

```
# Listen for incoming datagrams
```

```
while(True):
```

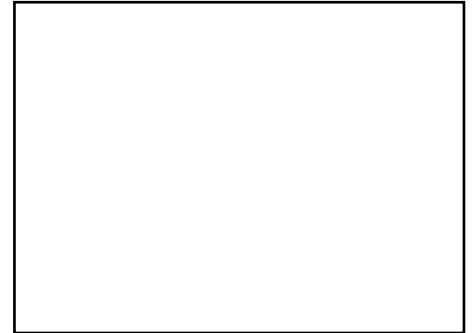
```
    message,address = UDPServerSocket.recvfrom(bufferSize)
```

```
    print(message.decode())
```

```
    print(address)
```

```
    # Sending a reply to client
```

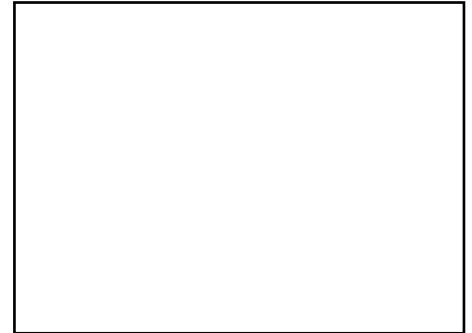
```
    UDPServerSocket.sendto(msgToClient, address)
```

**Output:**

**A UDP CLIENT:****Program udp-client.py**

```
import socket
```

```
msgToServer      = b"Hello UDP Server"
serverAddressPort = ("127.0.0.1", 20001)
bufferSize       = 1024
```

**Output:**

```
# Create a UDP socket at client side
```

```
UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
```

```
# Send to server using created UDP socket
```

```
UDPClientSocket.sendto(msgToServer, serverAddressPort)
```

```
serverMessage,serverAddress = UDPClientSocket.recvfrom(bufferSize)
```

```
print(serverMessage.decode())
```

```
print(serverAddress)
```

**A FILE SERVER:****Program file-server.py**

```
import socket

host = '127.0.0.1'
port = 2000

s = socket.socket()

t = (host,port)
s.bind(t)
s.listen(1)

c,addr = s.accept()

file_name = c.recv(1024)

fname = str(file_name.decode())
print("File Name received from client :", file_name)
try:
    f = open(file_name, "rb")
    data = f.read()
    c.send(data)
    print("File Data Sent To Client")
    f.close( )
except FileNotFoundError:
    c.send(b'File Does Not Exist')

c.close( )
```

**Output:**

File Name received from client :  
b'network\_file.txt'  
File Data Sent To Client

**A FILE CLIENT:****Program file-client.py**

```
import socket

host = '127.0.0.1'
port = 2000

s = socket.socket()

t = (host,port)
```

**Output:**

Enter the file name:  
network\_file.txt  
  
My name is Subhash. I am a  
programmer.

```
s.connect(t)

file_name = input("Enter the file name: ")
s.send(file_name.encode())

file_data = s.recv(1024)
print(file_data.decode())
s.close()
```

### **Two-Way Communication (Server Code):**

#### **Program server-code.py**

```
import socket

host = '127.0.0.1'
port = 2000

s = socket.socket()
s.bind((host,port))
s.listen(1)
c,addr = s.accept()

while True:
    message_from_client = c.recv(1024)

    if not message_from_client:
        break

    print("Client Says :", str(message_from_client.decode()))

    message_to_client = input("What do you wish to say to client? ")

    c.send(message_to_client.encode())

c.close()
```

#### **Output:**

```
Client Says : hi
What do you wish to say to
client? hello
Client Says : what are you doing?
What do you wish to say to
client? Waiting for your message
Client Says : ok bye
What do you wish to say to
client? hve
```

### **Two-Way Communication (Client Code):**

#### **Program client-code.py**

```
import socket
host = '127.0.0.1'
port = 2000

s = socket.socket()
t = (host,port)
s.connect(t)
```

#### **Output:**

```
Enter your message to server: hi
Message From Server : hello
Enter your message to server:
what are you doing?
Message From Server : Waiting
for your message
Enter your message to server: ok
bye
Message From Server : hve
```

```
message_to_server = input("Enter your message to server: ")

while message_to_server != 'stop':
    s.send(message_to_server.encode())

    message_from_server = s.recv(1024)
    message_from_server = message_from_server.decode()
    print("Message From Server : ", message_from_server )

    message_to_server = input("Enter your message to server: ")

s.close()
```

**Program findipaddress.py:**

#Knowing IP Address Of A Website

```
import socket
```

```
host = "www.google.com"
```

```
try:
```

```
    addr = socket.gethostbyname(host)
```

```
    print("IP Address = " + addr)
```

```
except socket.gaierror:
```

```
    print("The website does not exist")
```

## Lesson-18: Regular Expressions

### IMPORTANT NOTE:

\w (word character) matches any single letter, number or underscore (same as [a-zA-Z0-9\_]). The uppercase counterpart \W (non-word-character) matches any single character that doesn't match by \w (same as [^a-zA-Z0-9\_]).

\*: The preceding item will be matched zero or more times, i.e., 0+

+: The preceding item will be matched one or more times, i.e., 1+

#\d - Represents any digit (0 to 9) characters

#\D - Represents any non-digit characters

#\s - Represents white space characters

#\S - Represents non-white space characters

#\w - Represents any alphanumeric characters

#\W - Represents any non-alphanumeric characters

#\b - Represents a space around words

#\A - Matches only at the start of the string

#\Z - Matches only at the end of the string

# \* - 0 or more repetitions of the preceding regex

# + - 1 or more repetitions of the preceding regex

### Program 1:

```
import re

str = "I love you sub and India"
v = re.compile(r"s\w\w")
result = v.search(str)
if result:
    print(result.group( ))
```

### Output:

sub

### Program 2:

```
import re

str = "I love you sub and india"
res = re.search( r"s\w\w",str)
if result:
    print(result.group( ))
```

### Output:

sub

**Program 3:**

```
import re

str = "python is a great programming language and python is a snake"
res = re.findall( r"p\w\w\w\w\w",str)
if res:
    print(res)
else:
    print("Not Found")
```

**Output:**

```
['python', 'progra', 'python']
```

**Program 4:**

```
import re

strone = "C is a programming language and python is a snake"
strtwo = "python is a great programming language and python is a snake"

#match the starting word
resone = re.match( r"p\w\w\w\w\w",strone)
restwo = re.match( r"p\w\w\w\w\w",strtwo)
if resone:
    print("****")
    print(resone.group( ))

if restwo:
    print("----")
    print(restwo.group( ))
```

**Output:**

```
---
python
```

**Program 5:**

```
import re

s = "Subhash+- Programming*****Classes"
result = re.split(r"\W+", s)
print(result)
```

**Output:**

```
['Subhash', 'Programming',
'Classes']
```



**Program 6:**

```
import re

s = "Subhash Loves Cooking"
result = re.sub(r"Cooking", "Programming", s)
print(result)
```

**Output:**

Subhash Loves Programming

**Program 7**

```
import re

s = "subhash loves 3 hashing in programming and praying as hash at 5pm or 6"
d = "Subhash celebrates his birthday on 07-06-1985 and marriage anniversary on 28-08-2014"
a = "Subhash: 80808080 Charan: 9090909090 Archita: 1010101010"
b = "Hello World"

res = re.findall(r"h\w\w\w", s)
print(res)

res = re.findall(r"h[\w]*", s)
print(res)

res = re.findall(r"\bh[\w]*\b", s)
print(res)

res = re.findall(r"d\w", s)
print(res)

res = re.findall(r"d[\w]*", s)
print(res)

res = re.findall(r"d[\w]+", s)
print(res)

res = re.findall(r"\b\w{7}\b", s)
print(res)

res = re.findall(r"\b\w{5,}\b", s)
print(res)

res = re.findall(r"\b\w{2,3}\b", s)
print(res)

res = re.findall(r"\b\d\b", s)
```

```
print(res)

res = re.findall(r"[\w]\Z", s)
print(res)

res = re.findall(r"A[\w]*", s)
print(res)

res = re.findall(r"Ap[\w]*", s)
print(res)

res = re.findall(r"As[\w]*",s)
print(res)

res = re.findall(r"\w{1}", s)
print(res)

res = re.findall(r"\b\w{1}\b", s)
print(res)

res = re.findall(r"D+", s)
print(res)

res = re.findall(r"d{1,2}-d{1,2}-d{1,4}", d)
print(res)

res = re.findall(r"[A-Z][a-z]*", a)
print(res)

res = re.findall(r"d[0-9]*", a)
print(res)

res = re.search(r"^hello", b, re.IGNORECASE)
if(res):
    print("string starts with hello")
else:
    print("string does not start with hello")

res = re.search(r"world$", b, re.IGNORECASE)
if(res):
    print("string ends with world")
else:
    print("string does not end with world")
```

**Output:**

```
['hash', 'hash', 'hash']
['hash', 'hashing', 'hash']
['hashing', 'hash']
['5p']
['3', '5pm', '6']
['5pm']
['subhash', 'hashing', 'praying']
['subhash', 'loves', 'hashing', 'programming', 'praying']
['in', 'and', 'as', 'at', '5pm', 'or']
['3', '6']
['6']
['subhash']
[]
['subhash']
['s', 'u', 'b', 'h', 'a', 's', 'h', 'l', 'o', 'v', 'e', 's', '3', 'h', 'a', 's', 'h', 'i', 'n', 'g', 'i', 'n', 'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g', 'a', 'n', 'd', 'p', 'r', 'a', 'y', 'i', 'n', 'g', 'a', 's', 'h', 'a', 's', 'h', 'a', 't', '5', 'p', 'm', 'o', 'r', '6']
['3', '6']
['subhash loves ', ' hashing in programming and praying as hash at ', 'pm or ']
['07-06-1985', '28-08-2014']
['Subhash', 'Charan', 'Archita']
['80808080', '9090909090', '1010101010']
string starts with hello
string ends with world
```