Sir Syed University of Engineering & Technology (SSUET)

Software Engineering Department
Sir Syed University of Engineering &Technology, Karachi
Main University Road, Karachi 75300
http://www.ssuet.edu.pk

Course: Programming Fundamental

Name: Abdul Moiz Chishti

Roll Number: 2020F-BSE-022

Section: A

Semester: I

Lab File

Lab Teachers:

Mr.Adnan Afroz

Mrs.fizzah

LAB # 01

INTRODUCTION

EXERCISE

A. Create a file named lab1.py. Write the following code in the file. Execute it and show the output. (You can use the Snipping Tool to take a snapshot of your output window).

1. Code:

```
#Abdul Moiz Chishti
#BSE-20F-022
```

```
# My first program
print("Welcome in the world of programming! \n")
```

Output:

```
Python 3.7.7 (bundled)
>>> %Run 'Task 1 lab 1.py'

  Welcome in the world of programming!
```

2. Code:

```
#Abdul Moiz Chishti
#BSE-20F-022
#My second program
print("Welcome in the ")  print("world
of programming! \n")
```

Output:

```
>>> %Run 'task 2 lab 1.py'

  Welcome in the
  world of programming!
```

B. Write a program in Python language that prints your bio-data on the screen. Show your program and its output.

Code:

```
#Abdul Moiz Chishti
#BSE-20F-022
#Bio-data print(‘ Bio Data’)
print(“ Name : Abdul Moiz Chishti”)
print(“ Father Name : Haroon
Chishti”)
print(“ Roll No. : SE20F-022”)
print(‘ Technology : Software Engineering’)
```

Output:

```
>>> %Run 'Biodata Task3.py'

    Bio Data
    Name : Abdul Moiz Chishti
    Father Name : Haroon Chishti
    Roll No. : SE20F-022
    Technology : Software Engineering
```

LAB # 02

VARIABLES AND OPERATORS

OBJECTIVE

Implement different type of data types, variables and operators used in Python.

THEORY

Variable

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Rules for constructing variable names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python 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)

Example

```
x= 5 y=
"John"
print(x)
print (y)
```

Output:

```
>>> %Run task1.py
5
John
```

Variables do not need to be declared with any particular type and can even change type after they have been set.

Example:

```
x= 4 x=
"Sally"
print(x)
```

Output:

```
>>> %Run task2.py
Sally
```

Assign Value to Multiple Variables

Python allows you to assign values to multiple variables in one line

Example:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x) print(y) print(z)
```

Output:

```
>>> %Run task3.py
Orange
Banana
Cherry
```

To combine both text and a variable, Python uses the + character

Example:

```
x= "awesome"
print("Python is " , x)
```

Output:

```
>>> %Run task4.py Python
is awesome
```

Python Keywords

Keywords are the words whose meaning have already been explained to the Python compiler. The keywords cannot be used as variable names,function name or any identifier because if we do so we are trying to assign a new meaning to the keyword, which is not allowed by the computer. Keywords are also called 'Reserved words'. Some keywords are as follows:

false	class	finally	is	return	none	continue	for	try	break
true	def	for	from	while	and	del	not	with	as
elif	if	or	except	in	raise	yield			

Data Types

Data types specify how we enter data into our programs and what type of data we enter. Python Data Types are used to define the type of a variable.

Python has five standard data types −

- Numbers (int, float)
- String
- List
- Tuple
- Dictionary

You can get the data type of any object by using the "type( )" function.

Operators

Operators are special symbols in Python that carry out arithmetic or logical computation.

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	**Name**	**Example**
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Python Relational Operators

Comparison operators are used to compare two values:

Operator	Name	Example
= =	Equal	x = = y

!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Name	Example
and	Return True if both statements are true	x < 5 and x < 10
or	Return True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not (x<5 and x< 10)

EXERCISE

A. Point out the errors, if any, in the following Python statements.

1.
```
x=5:
  print(x)
```

In this program (:) is causing the error in the first line.

2.
```
1TEXT = "SSUET"
NUMBER = 1
print(NUMBER+ TEXT)
```

In this program (1) is causing the error in the first line and in the third line (+) is causing the error.

3.
```
a = b = 3 = 4
```

In this program (=) is causing error b/w a and b , 3 and 4

B. Evaluate the operation in each of the following statements, and show the resultant value after each statement is executed.

1.
```
a = 2 % 2 + 2 * 2 - 2 / 2;
```

```
>>> %Run 'Lab 2 1.py'
  3.0
>>>
```

2. `b = 3 / 2 + 5 * 4 / 3 ;`

```
>>> %Run 'Lab 2 1.py'
  8.166666666666668
>>>
```

3. `c = b = a = 3 + 4 ;`

```
>>> %Run 'Lab 2 1.py'
  7
>>>
```

C. Write the following Python programs:

1. Write a program that calculates area of a circle $A=\pi r^2$. (Consider $r = 50$).

Code:

```
"""
Abdul Moiz Chishti
SE-20F-022
"""

pi = 3.14 #value of pi
r = 50 #value of radius of a circle
area = pi*r**2
print("area=",area,"sq cm")
```

Output:

```
>>> %Run 'Lab 2 1.py'
  area= 7850.0 sq cm
>>>
```

2. Write a program that performs the following four operations and prints their result on the screen.
   a. 50 + 4
b. 50 – 4
c. 50 * 4
d. 50 / 4

Code:

```
"""
Abdul Moiz Chishti
SE-20F-022
"""

a=50+4
b=50-4
c=50*4
d=50/4
print("Four Operations\n")
print("Solution of 50+4 =",a)
print("Solution of 50-4 =",b)
print("Solution of 50*4 =",c)
print("solution of 50/4 =",d)
```

Output:

```
>>> %Run 'Lab 2 1.py'

  Four Operations

  Solution of 50+4 = 54
  Solution of 50-4 = 46
  Solution of 50*4 = 200
  solution of 50/4 = 12.5

>>>
```

3. Write a Python program to convert height (in feet and inches) to centimeters. Convert height of 5 feet 7 inches to centimeters.

- First, convert 5 feet to inches: 5 feet × 12 inches/foot = 60 inches
- Add up our inches: 60 + 7 = 67 inches
- Convert inches to cm: 67 inches × 2.54 cm/inch = 170.18 cm

Code:

```
"""

Abdul Moiz Chishti

SE-20F-022

"""

print("feet = 5")

f=5

print("inches = 7")

i=7

x=(f*12)+i

print("height in Centimeters = (((f*12)+i)*2.54) =", (x*2.54))
```

Output:

```
>>> %Run 'Task 1.py'

  feet = 5
  inches = 7
  height in Centimeters = (((f*12)+i)*2.54) = 170.18

>>>
```

4. Write a program to compute distance between two points by creating variables (Pythagorean Theorem)

$$\text{Distance} = ((x2-x1)^2+(y2-y1)^2)^{1/2}$$

Code:

```
"""
Abdul Moiz Chishti
SE-20F-022
"""
x1=12
x2=24
y1=36
y2=48
print("distance=",((x2-x1)**2+(y2-y1)**2)**(1/2))
```

Output:

```
Python 3.7.7 (bundled)
>>> %Run 'Task 1.py'

  distance= 16.97056274847714

>>>
```

LAB # 03

CONSOLE INPUT AND OUTPUT

OBJECTIVE

Taking input from user and controlling output position.

THEORY

Console I/O Functions

The keyboard and visual display unit (VDU) together are called a console. Python programming language provides many built-in functions to read any given input and to display data on screen, Console (also called Shell) is basically a command line interpreter that takes input from the user i.e one command at a time and interprets it. If it is error free then it runs the command and gives required output otherwise shows the error message.

Accepting Input from Console

To take input from the user we make use of a built-in function *input()*.

Syntax : input(*prompt*)

Displaying Input from Console

The print( ) function prints the specified message to the screen, or other standard output device.

The message can be a string, or any other object, the object will be converted into a string before written to the screen.

Syntax: print*(object(s)*, separator=*separator*, end=*end*, file=*file*, flush=*flush*)

Example:

```
name=input('Please enter your name: "') print("Hello, " , name ,  "!")
```

Output:

```
>>> %Run task1.py Please enter your name:ABC Hello, ABC!
>>>
```

Whatever you enter as input, input function convert it into a string. if you enter an integer value still input() function convert it into a string. You need to explicitly convert it into an integer in your code using typecasting.

Example:

```
# Program to check input     num
= input ("Enter number :")
print(num)
name1 = input("Enter name : ")
print(name1)

# Printing type of input value  print
("type of number", type(num))
print ("type of name",
type(name1))
```

We can also type cast this input to integer, float or string by specifying the input() function inside the type.

Typecasting the input to Integer/Float: There might be conditions when you might require integer input from user/console, the following code takes two input(integer/float) from console and typecasts them to integer then prints the sum.

Example

```
# input  num1 =
int(input())  num2
= int(input())

# printing the sum in integer  print(num1
+ num2)
```

Escape Sequence

In Python strings, the backslash "\" is a special character, also called the "**escape**" character. An escape sequence is a sequence of characters that does not represent itself when used inside a character or string literal, but is translated into another character or a sequence of characters that may be difficult or impossible to represent directly.

Escape Sequence	**Description**	**Example**	**Output**
\\	Prints Backslash	print ("\\")	\
\`	Prints single-quote	print ("\'")	'
\"	Pirnts double quote	print ("\"")	"
\n	ASCII linefeed (LF)	print ("hello\nworld")	hello world

\b	ASCII backspace (BS) removes previous character	print ("az" + "\b" + "c")	ac
\t	ASCII horizontal tab (TAB). Prints TAB	print ("\t*hello")	*hello

EXERCISE

A. Point out the errors or undefined/missing syntax, if any, in the following python programs.

```
1. print("Hello \b World!")
```

In this program space should not be there before \b

```
2. first_number = str ( input ("Enter first number") )
   second_number = str ( input ("Enter second number") )
       sum = (first_number +
second_number)
   print("Addition of two number is: ", sum)
```

In this program "str" should be replaced by "int"

```
3. age = 23
  message = "Happy " + age + "rd Birthday!"
print(message)
```

In this Program "+" should be replaced by ","

B. What would be the output of the following programs:

```
1.  a=5
  print("a =", a, sep='0', end=',')
```

```
Python 3.7.7 (bundled)
>>> %Run practise.py

  a =05,

>>>
```

2.
```
name = input("Enter Employee Name") salary = input("Enter salary") company = input ("Enter Company name") print("Printing Employee Details") print ("Name", "Salary", "Company")
print (name, salary, company)
```

```
>>> %Run practise.py

  Enter Employee Name Abdul Moiz Chishti
  Enter salary 70000
  Enter Company name SSUET
  Printing Employee Details
  Name Salary Company
   Abdul Moiz Chishti  70000  SSUET
```

3.
```
n1=int(input('"enter n1 value'))
n2=int(input('enter n2 value'))
```

```
>>> %Run practise.py

  "enter n1 value 5
  enter n2 value6
```

C. Write Python programs for the following:

1. Write a program to print a student's bio data having his/her Date of birth, Roll no, Section, Percentage and grade of matriculation and Intermediate. All the fields should be entered from the console at run time.

Code:

```
#Bio-Data using input function
print('\t\"BIO_DATA"')
bd= input('Enter Your Birth Date : ')
rn= input('Enter Your Roll No. : ')
sec= input('Enter Section: ')
matric_grade=input('Enter Matric Grade : ')
matric_percentage=input('Enter Matric Percentage : ')
inter_grade=input('Enter Inter Grade : ')
inter_percentage=input('Enter Intermediate Percentage : ')
print('\n\t\"BIO-DATA\"')
print('\nBirth Date : ',bd)
print('\nRoll No. : ',rn)
```

Output:

```
Python 3.7.7 (bundled)
>>> %Run 'Lab3 Task 1.py'

          "BIO_DATA"
  Enter Your Birth Date : 13-07-2002
  Enter Your Roll No. : SE-22A
  Enter Section: A
  Enter Matric Grade : A-1
  Enter Matric Percentage : 84
  Enter Inter Grade : B
  Enter Intermediate Percentage : 61

          "BIO-DATA"

  Birth Date :  13-07-2002

  Roll No. :  SE-22A

  Section :  A

  Matric Grade :  A-1

  Matric Percentage :  84

  Inter Grade :  B

  Inter percentage :  61

>>>
```

2. Write a program that asks the user what kind of food they would like. Print a message about that food, such as “Let me see if I can find you a Chowmein”. Food name must be in uppercase. (hint: use upper( ) for food name)

Code:

```
food= input('What kind of food would you like : ')
#uppercase command
print('\n Let me see if i can Find you ',food.upper())
```

Output:

```
Python 3.7.7 (bundled)
>>> %Run 'Lab 3 Task 2.py'

  What kind of food would you like : CHINESE RICE

   Let me see if i can Find you  CHINESE RICE

>>>
```

3. Take the marks of 5 courses from the user and calculate the average and percentage, display the result:

   Eachcourse=50 marks

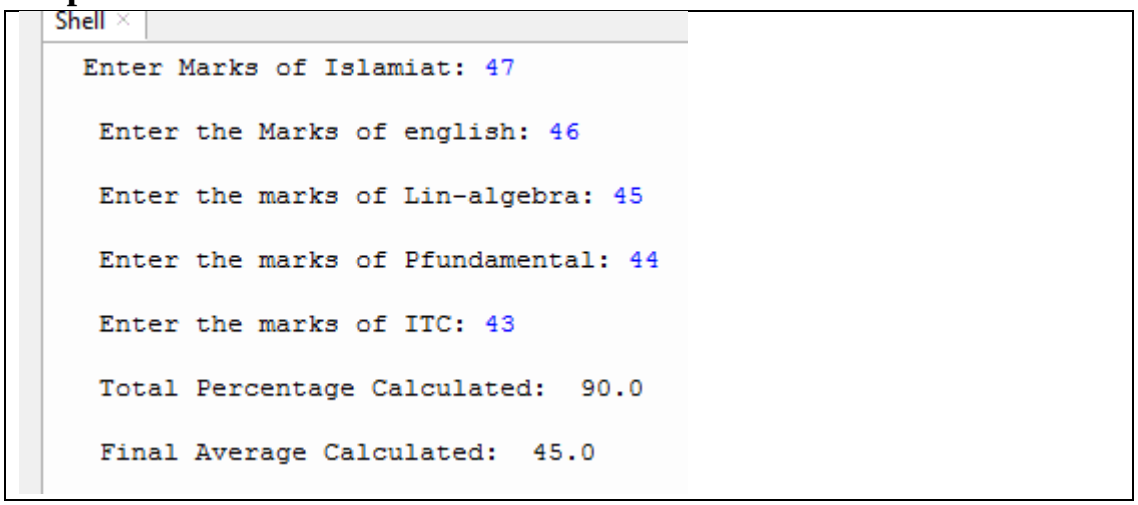
   Total_marks= course1+course2+course3+course4+course5

   average=Total_marks/5 percentage=(Total_marks x 100)/250

Code:

```
#course detail
course=50
Islamiat=input("Enter Marks of Islamiat: ")
English=input("\n Enter the Marks of english: ")
LinAlg=input("\n Enter the marks of Lin-algebra: ")
Pfund=input("\n Enter the marks of Pfundamental: ")
ITC=input("\n Enter the marks of ITC: ")
Obtained_Marks=int(Islamiat)+int(English)+int(LinAlg)+int(Pfund)+int(ITC)
Total_Marks=course*5
average = Obtained_Marks/5
percentage=(Obtained_Marks*100)/Total_Marks
print("\n Total Percentage Calculated: ", percentage)
print("\n Final Average Calculated: ", average)
```

Output:


Shell
Enter Marks of Islamiat: 47
Enter the Marks of english: 46
Enter the marks of Lin-algebra: 45
Enter the marks of Pfundamental: 44
Enter the marks of ITC: 43
Total Percentage Calculated: 90.0
Final Average Calculated: 45.0

LAB # 04

DECISIONS

OBJECTIVE

To get familiar with the concept of conditional statement for simple decision making.

THEORY

Decision making statements in programming languages decides the direction of flow of program execution.

The if…elif…else statement is used in Python for decision making.

The *if* Statement

Like most languages, python uses the keyword if to implement the decision control instruction. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not. The general form of if statement looks like this:

Syntax: if condition:
Statements to execute if
  # condition is true

As a general rule, we express a condition using python's 'relational' operators. The relational operators allow us to compare two values to see whether they are equal to each other, unequal, or whether one is greater than the other. Here is how they look and how they are evaluated in python.

This expression	Is true if
`x==y`	x is equal to y
`x!=y`	x is not equal to y
`x<y`	x is less than y
`x>y`	x is greater than y
`x<=y`	x is less than or equal to y
`x>=y`	x is greater than or equal to y

Example:

```
# python program to illustrate If statement
i = 10 if (i > 15):     print (i,
"is greater than 15") print ("I
am not greater")
```

Output:

```
>>> %Run task1.py
I am not greater
>>>
```

The *if-else* Statement

We can use the else statement with if statement to execute a block of code when the condition is false.

Syntax

if (condition):
  # Executes this block if
condition is true else:
  # Executes this block if
  # condition is false

Example:

```
# python program to illustrate If else
statement i = 20;  if (i < 15):      print
(i,“is smaller than 15")      print ("i'm in if
Block")  else:      print (i,"is greater than
15")      print ("i'm in else Block")
print ("i'm not in if and not in else Block")
```

Output:

```
>>> %Run task2.py
20 is greater than 15
i'm in else Block
i'm not in if and not in else Block
>>>
```

The *if-elif-else* Statement

The elif is short for else if. It allows us to check for multiple expressions. If the condition for if is False, it checks the condition of the next elif block and so on. If all the conditions are False, body of else is executed.

Syntax
if (condition):
statement elif
(condition):
statement
.
. else:
statement

Example:

```
# Python program to illustrate if-elif-else
i = 30
if (i == 10):
    print ("i is 10")
elif (i == 20):
    print ("i is 20")
elif (i == 30):
    print ("i is 30")
else:
    print ("i is not present")
```

Output:

```
>>> %Run task3.py
i is 30
>>>
```

EXERCISE

A. Point out the errors, if any, in the following Python programs.

1.

```
a     =
500,b,c; if (
a >= 400 ):
b     = 300
c = 200
    print( "Value is:", b, c )
```

In this program the numeric value in variable in string is causing an error

2. Code

```
&number = eval(input("Enter an integer: "))
print(type(number)) if number % 5 == 0
print("HiFive") else
    print("No answer")
```

In this program the variable name is starting with "&" which is causing an error.

3. Code

```
if score >= 60.0     grade =
'D' elif score >= 70.0
grade = 'C' elif score >= 80.0
grade = 'B' elif score >= 90.0
grade = 'A' else:
    grade = 'F'
```

In this program colon":" is necessary after each condition

B. What would be the output of the following programs:

1. Code

```
requested_topping = 'mushrooms' if
requested_topping != 'anchovies':
    print("Hold the anchovies!")
```

Output

```
>>> %Run practise.py

  Hold the anchovies!

>>>
```

2. Code

```
num = 3 if num >= 0:
print("Positive or Zero") else:
    print("Negative number")
```

Output

```
>>> %Run practise.py

  Positive or Zero

>>>
```

3. Code

```
age = 15 if age
< 4:     price =
0 elif age < 18:
price = 1500
else:
    price = 2000
print("Your admission cost is Rs" + str(price) + ".")
```

Output

```
>>> %Run practise.py

  Your admission cost is Rs1500.

>>>
```

C. Write Python programs for the following:

1. Any integer is input through the keyboard. Write a program to find out whether it is an odd number or even number.

CODE:

```
#Check whether the number is even or odd
Num= int(input('Enter Any Number'))
if(Num%2==0):
   print(Num,"is even number")
else:
   print(Num,"is odd number")
```

OUTPUT:

```
>>> %Run Task1.py

  Enter Any Number175
  175 is odd number

>>> %Run Task1.py

  Enter Any Number144
  144 is even number

>>>
```

2. Write a program that asks for years of service and qualification from the user and calculates the salary as per the following table:

Years of Service	Qualifications	Salary
>= 10	Masters	150,000
>= 10	Bachelors	100,000
< 10	Masters	100,000
< 10	Bachelors	70,000

CODE:

```
yos= int(input("enter year of service"))
q= input("enter your qualification")
que=q.lower()
if yos>=10 and que=="masters":
   print("Your Salary is 150000")
elif yos>=10 and que=="bachelors":
   print("your salary is 100000")
elif yos<10 and que=="masters":
   print("your salary is 100000")
elif yos<10 and que=="bachelors":
   print("your salary is 70000" )
else:
   print("you are not qualified enough")
```

OUTPUT:

```
>>> %Run task2.py

  enter year of service11
  enter your qualificationMASTERS
  Your Salary is 150000

>>> %Run task2.py

  enter year of service12
  enter your qualificationbachelors
  your salary is 100000

>>> %Run task2.py

  enter year of service8
  enter your qualificationmasters
  your salary is 100000

>>> %Run task2.py

  enter year of service7
  enter your qualificationBACHELORS
  your salary is 70000
```

3. Write an if-elif-else chain that determines a person's stage of life, take input value for the variable age, and then apply these conditions:

• If the person is less than 2 years old, print a message that the person is a baby.

- If the person is at least 2 years old but less than 4, print a message that the person is a toddler.
- If the person is at least 4 years old but less than 13, print a message that the person is a kid.
- If the person is at least 13 years old but less than 20, print a message that the person is a teenager.
- If the person is at least 20 years old but less than 65, print a message that the person is an adult.
- If the person is age 65 or older, print a message that the person is an elder.

CODE:

```
age=(int(input("Enter your Age")))
if age < 2:
   print("the person is baby")
elif age >=2 and age <4:
   print ("The person is a toddler")
elif age >=4 and age <13:
   print("The person is a kid")
elif age >=13 and age <20:
   print("The person is a teenager")
elif age >20 and age <65:
   print("The person is an adult")
elif age >=65:
   print("The person is an elder")
```

OUTPUT:

```
>>> %Run 'task 3.py'

  Enter your Age1
  the person is baby

>>> %Run 'task 3.py'

  Enter your Age3
  The person is a toddler

>>> %Run 'task 3.py'

  Enter your Age18
  The person is a teenager

>>> %Run 'task 3.py'

  Enter your Age25
  The person is an adult

>>> %Run 'task 3.py'

  Enter your Age76
  The person is an elder

>>>
```

LAB # 05

ITERATION WITH LOOPS

OBJECTIVE

To get familiar with the different types of loops.

THEORY

A loop can be used to tell a program to execute statements repeatedly. In other word, to keep a computer doing useful work we need repetition, looping back over the same block of code again and again.

Type of Loop Statements in Python:

There are 2 types of loop statements in Python language. They are,

1. for
2. while

The for loop:

A Python **for** loop iterates through each value in a sequence.

Syntax:

In general, the syntax of a **for** loop is:

```
for var in sequence:
    # Loop body
```

for loop can be used to simplify the preceding loop:

```
a) for i in range(endValue):
       #Loop body

b) for i in range(initialValue, endValue):
       # Loop body

c) for i in range range(initialValue, endValue,k):   #k=step value
   # Loop body
```

How for loop works:

- A ‘sequence’ holds multiple items of data, stored one after the other. In sequence introduce strings and data storing techniques.They are sequence-type objects in Python.

- The variable '**Var'** takes on each successive value in the sequence, and
- The statements in the body of the loop are executed once for each value.
- Generate a sequence of numbers using '**range() function'**, range(10) will generate numbers from 0 to 9 (10 numbers).

Example program for 'for loop':

```
#simple for loop using sequence(string) for letter in
'Python':
   print ('Current Letter :', letter)
```

Output:

```
>>> %Run task1.py
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
```

Example program for 'for loop using range( )':

```
#Simple for loop using range()
for i in range(1,6):
   print("Programming is fun")
```

Output:

```
>>> %Run task2.py
Programming is fun
Programming is fun
Programming is fun
Programming is fun
Programming is fun
```

The while loop:

A **while** loop executes statements repeatedly as long as a condition remains true.

Syntax:

The syntax of a **while** loop in Python programming language is:

```
while loop-continuation-condition:
   # Loop body
     Statement(s)
```

How while loop works:

In while loop first the condition (boolean expression) is tested; if it is false the loop is finished without executing the statement(s). If the condition is true, then the statements are executed and the loop executes again and again until the condition is false. Each loop contains a loop-continuation-condition, a Boolean expression that controls the body's execution.

Example program for 'while loop':

```
count = 0 while count < 5:
print("Programming is fun!")
count += 1
```

Output:

```
>>> %Run task3.py
Programming is fun!
Programming is fun!
Programming is fun!
Programming is fun! Programming
is fun!
```

EXERCISE

A. Point out the errors, if any, in the following Python programs.

1. Code:

```
for(;;)
{
   printf("SSUET")
}
```

In this program semi colon ";" is causing an error.

2. Code:

```
count = 4 while
n < 8
     count = count + 3:
print(count)
```

In this program ":" is missing after the condition and is causing an error.

3.Code

```
for v in range(4:8)
print(v)
```

In this program the error is in the first line because of ":" . it should be replaced with "," .

B. What will be the output of the following programs:

1. Code

```
i = 1 while i < 10:
if i % 2 == 0:
print(i)
    i += 1
```

Output

```
>>> %Run practise.py

  2
  4
  6
  8

>>>
```

2. Code

```
i = 1 while i>0:
print(i)
    i = i + 1
```

Output

```
Python 3.7.7 (bundled)
>>> %Run practise.py

  1
  2
  3
  4
  5
  6
  7
  8
  9
  10
  11
  12
  13
  14
  15
  16
  17
  18
  19
  20
  21
  22
  23
  24
```

3. Code

```
for v in range(3, 9, 2):
print(v)
```

Output

```
Python 3.7.7 (bundled)
>>> %Run practise.py

  3
  5
  7

>>>
```

C. Write Python programs for the following:

1. Write a program that prints the first 10 natural numbers and their sum using 'for loop'.
   Sample output:
   The first 10 natural number are :
   1 2 3 4 5 6 7 8 9 10
   The Sum is : 55

CODE:

```
#ABDUL MOIZ CHISHTI BSE-20F-022
i=0
a=0
for i in range(1,11):
   print (i,end=',')
   a=a+i
print( " \nthe sum of 10 natural no", a)
```

OUTPUT:

```
Python 3.7.7 (bundled)
>>> %Run s.py

  1,2,3,4,5,6,7,8,9,10,
  the sum of 10 natural no 55

>>>
```

USING WHILE LOOP:

CODE:

```
i=1
a=0
while i<11:
   print(i,end=',')
   i +=1
   a=a+i

a=a-10
print("\n the sum of first 10 natural number" ,"=", a)
```

OUTPUT:

```
>>> %Run s.py

  1,2,3,4,5,6,7,8,9,10,
   the sum of first 10 natural number = 55

>>>
```

2. Write a program to print the multiplication table of the number entered by the user. The table should get displayed in the following form.

   29 x 1 = 29
   29 x 2 = 58
   …
   29 x 10 = 290

CODE:

```
#ABDUL MOIZ CHISHTI BSE-20F-022
num = int(input("Enter the number: "))

print("Multiplication Table of", num)
for i in range(1, 11):
  print(num,"X",i,"=",num * i)
```

OUTPUT:

```
Enter the number: 23
Multiplication Table of 23
23 X 1 = 23
23 X 2 = 46
23 X 3 = 69
23 X 4 = 92
23 X 5 = 115
23 X 6 = 138
23 X 7 = 161
23 X 8 = 184
23 X 9 = 207
23 X 10 = 230
```

USING WHILE LOOP:

Code:

```
num= int(input("Enter the number: "))
i=1
while i<=10:
   print(num,"X",i,"=",num * i)
   i=i+1
```

OUTPUT:

```
>>> %Run 'while loop1.py'

  Enter the number: 5
  5 X 1 = 5
  5 X 2 = 10
  5 X 3 = 15
  5 X 4 = 20
  5 X 5 = 25
  5 X 6 = 30
  5 X 7 = 35
  5 X 8 = 40
  5 X 9 = 45
  5 X 10 = 50

>>>
```

3. Write a program that take vowels character in a variable named “vowels” then print each vowels characters in newline using ‘for loop’.

CODE:

```
#ABDUL MOIZ CHISHTI BSE-20F-022
var = input("enter a phrase")
for i in var:
    if i in ('aeiou'):
        print(i)

print("\t these are the vowels present in the phrase", var)
```

OUTPUT:

```
Python 3.7.7 (bundled)
>>> %Run 'task 31.py'

  enter a phrase sir syed university
  i
  e
  u
  i
  e
  i
           these are the vowels present in the phrase  sir syed university

>>>
```

LAB # 06

NESTED STATEMENTS, BREAK AND CONTINUE STATEMENTS

OBJECTIVE

Working on nested statements and control loop iteration using break and continue.

THEORY

Nested Statements:

A Nested statement is a statement that is the target of another statement. **Nested if:**

A Nested ***if*** is an ***if*** statement that is the target of another ***if*** statement. Nested ***if*** statements means an ***if*** statement inside another ***if*** statement.

Syntax:

```
if (condition1):
  # Executes when condition1 is true    if
(condition2):
     # Executes when condition2 is true
  # if Block is end here
# if Block is end here
```

Example:

```
#using nested if
x=int(input("enter number="))
y=int(input("enter 2nd number="))
if x > 2:     if y > 2:
z = x + y          print("z is",
z)  else:     print("x is", x)
```

Output:

```
>>> %Run task1.py
enter number=3
enter 2nd number=8
       z is 11 >>>
```

Nested loops:

Nested loops consist of an outer loop and one or more inner loops. Each time the outer loop repeats, the inner loops are reinitialize and start again.

Example:

```
height=int(input("Enter height:
")) for row in range(1, height):
for column in range(1,height):
print(row, end=" ")
print()
```

Output:

```
>>> %Run task2.py
Enter height: 7
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
5 5 5 5 5 5
6 6 6 6 6 6
```

Keywords break and continue:

The break and continue keywords provide additional controls to a loop.

The Break Statement:

The keyword ***break*** in a loop to immediately terminate a loop. Listing example presents a program to demonstrate the effect of using ***break*** in a loop.

Syntax: break

Example:

```
# Use of break statement inside
loop for word in "string":     if
word == "i":
        break
print(word) print("The
end")
```

Output:

```
>>> %Run task3.py'
s t r The end >>>
```

The continue Statement:

The ***continue*** statement breaks out of the current iteration in the loop.

Syntax: continue

Example:

```
# Program to show
the use of continue
statement inside
loops for val in
"string":     if
val == "i":
continue
    print(word)
print("The end"
```

Output:

```
>>> %Run task4.py' s t r n g
The end
>>>
```

EXERCISE

A. Point out the errors, if any, in the following Python programs.

1. Code

```
prompt = "\nPlease enter the name of a city you have visited:"
prompt+="\n(Enter 'quit' when you are finished.)"  while
True:     city = str(input(prompt))     if city == quit:
break;     else:         print("I'd love to go to " ,
city.title() , "!")
```

Output

```
In this program there is an indended block before if condition
```

2. **Code**

```
if x>2:    if
y>2:
z=x+y
      print(“z is”,
y)    else
print(“x is”, x)
```

Output

In this program variable is not assigned.

2. Code

```
balance = int(input("enter your
balance1:")) while true:  if balance
<=9000:     continue;    balance =
balance+999.99 print("Balance is",
balance)
```

Output

In this program variable true is not defined .

B. What will be the output of the following programs:

1. Code

```
i = 10 if (i ==
10):
    #  First if statement      if (i <
15):          print ("i is smaller than
15")
    # Nested - if statement
    # Will only be executed if statement above
# it is true      if (i < 12):
print ("i is smaller than 12 too")      else:
print ("i is greater than 15")
```

Output

```
>>> %Run 63.py

  i is smaller than 15
  i is smaller than 12 too

>>>
```

2. Code

```
i = 1 j = 2 k = 3 if i
> j:       if  i  >  k:
print('A')  else:
print('B')
```

```
i = 1 j = 2 k = 3 if
i > j:      if
i> k:
print('A')     else:
        print('B')
```

Output :

```
>>> %Run 63.py

  B
```

```
>>> %Run 63.py

  B
```

3. Code

```
# nested for loops  for i
in range(0, 5):  for j
in range(i):
print(i, end=' ')
print()
```

Output

```
>>> %Run 63.py


  1
  2 2
  3 3 3
  4 4 4 4

>>>
```

C. Write Python programs for the following:

1. Write a program to add first seven terms twice of the following series:

$$\frac{1}{1!}+\frac{2}{2!}+\frac{3}{3!}+\cdots$$

CODE:

```
s=0
for num in range(1,8):
   factorial=1
   for i in range(1,num+1):
      factorial=factorial*i

   factorial_s= num/factorial
   s= s+factorial_s

print("sum of first seven numbers of the series is
=",round(s,3))
```

OUTPUT:

```
>>> %cd 'D:\Pfundamental\Pfundamental Lab\Lab 6'
>>> %Run 'Task 1.py'

  sum of first seven numbers of the series is = 2.718
```

2. Write a program to print all prime numbers from 900 to 1000.
   [Hint: Use nested loops, break and continue]

Code:

```
a = 900
b = 1000

print("\tFollowing are the Prime numbers between", a, "and", b,"\n")

for num in range(a, b + 1):
   if num > 1:
      for i in range(2, num):
         if (num % i) == 0:
            break

      else:
         print(num, end =" ")
```

OUTPUT:

```
>>> %Run 'Task 2.py'

          Following are the Prime numbers between 900 and 1000

  907 911 919 929 937 941 947 953 967 971 977 983 991 997

>>>
```

3. Write a program to display multiplication table(1-5) using nested looping
   Sampled output:[hint: '{ } ' .format(value)]
   02 X 01 = 02

CODE:

```
for i in range (1,6):
   print ("\tTable of ",i,"\n")
   for j in range(1,11):
      print(i,"x",j,"=","{:2d}".format(i * j) )
   print("\n")
```

OUTPUT:

```
>>> %Run 'Nested for loop.py'

          Table of  1

  1 x 1 =  1
  1 x 2 =  2
  1 x 3 =  3
  1 x 4 =  4
  1 x 5 =  5
  1 x 6 =  6
  1 x 7 =  7
  1 x 8 =  8
  1 x 9 =  9
  1 x 10 = 10


          Table of  2

  2 x 1 =  2
  2 x 2 =  4
  2 x 3 =  6
  2 x 4 =  8
  2 x 5 = 10
  2 x 6 = 12
  2 x 7 = 14
  2 x 8 = 16
  2 x 9 = 18
  2 x 10 = 20


          Table of  3

  3 x 1 =  3
```

```
          Table of  3

3 x 1 =  3
3 x 2 =  6
3 x 3 =  9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30


          Table of  4

4 x 1 =  4
4 x 2 =  8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40


          Table of  5

5 x 1 =  5
5 x 2 = 10
```

```
        Table of  5

5 x 1 =  5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

LAB # 07

FUNCTIONS

OBJECTIVE

Create python function using different argument types.

THEORY

Functions can be used to define reusable code and organize and simplify code.
Basically, we can divide functions into the following two types:

1. Built-in functions - Functions that are built into Python.
2. User-defined functions - Functions defined by the users themselves.

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.
As already known, Python gives you many built-in functions like print(), etc. But also allow to create your own functions. These functions are called user-defined functions.

Defining a Function:

A function definition consists of the function's name, parameters, and body. Python allows to define functions to provide the required functionality. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword ***def*** followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The code block within every function starts with a colon (:) and is indented.

Syntax:

def functioname (list of parameters):
    Statement

Example:

```
def greet_user(username):
    """Display a simple greeting"""
    print("Hello," , username , "!")
```

Calling a Function:

Calling a function executes the code in the function. If the function returns a value, a call to that function is usually treated as a value.

Example:

```
def greet_user(username):
    """Display a simple greeting"""
    print("Hello," , username , "!")
greet_user('Jesse')
```

Output:

```
>>> %Run task1.py
Hello, Jesse !
>>>
```

Return Value:

The python return statement is used in a function to return something to the caller program. Use the return statement inside a function only.

Example:

```
def xFunction(x, y):
    print("Add", x + y)
    return
xFunction(2, 3)
```

Output:

```
>>> %Run task2.py
Add: 5
>>>
```

When a function doesn't have a return statement and return statement has no value, the function returns ***None.***

Argument Types:

An argument is a piece of information that is passed from a function, When a function is called, it place the value to work with in parentheses.

Following are the ways to call a function by using the following types of formal arguments:

- Required arguments
- Keyword arguments
- Default arguments

Required Arguments:
Required arguments are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition.

Example:

```
def square(x):
    y=x*x
    return y
x=10
result=square(x)
print("The result of" , x , "squared is", result)
```

Output:

```
>>> %Run task2.py
The result of 10 squared is 100
>>>
```

Keyword Arguments:
Keyword arguments are related to the function calls. When thw keyword argument is used in a function call, the caller identifies the arguments by the parameter name. This allows to skip the arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters.

Example:

```
def print_info(name,age):
    "This prints a passed info into this function"
    print("Name:", name)
    print("Age:" , age)
    return

print_info(name ='Mike',age=50)
```

Output:

```
>>> %Run task3.py
Name: Mike
Age: 50
>>>
```

Default Arguments:
A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

Example:

```
def print_info(name,age=35):
    "This prints a passed info into this function"
    print("Name:", name)
    print("Age:" , age)
    return

print_info(name ='Mike',age=50)
print_info(name ='Mary')
```

Output:

```
>>> %Run task4.py
Name: Mike
Age: 50
Name: Mary
Age: 35
>>>
```

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code

```
define sub(x, y)
return x + y
```

Output

In this program define is used instead of (def) and there is no identation before return. These are causing an error.

2. Code

```
define describe_pet(pet_name, animal_type='dog'):
    print("\nI have a " , animal_type ,".")
    print("My " , animal_type + "'s name is " , pet_name + ".")
```

Output

In this program define is used instead of (def)

2. Code

```
def type_of_int(i):
    if i // 2 == 0:
        return 'even'
    else:
        return 'odd'
```

Output

```
In this program there is no error.
```

B. What will be the output of the following programs:

1. Code

```
def test(a):
    def add(b):
        a =+ 1
        return a+b
    return add
func = test(4)
print(func(4))
```

Output

```
>>> %Run cp.py

  5

>>>
```

2. Code

```
def return_none():
   return
print(return_none())
```

Output

```
>>> %Run cp.py

  None

>>>
```

3. Code

```
def test_range(n):
```

```
    if n in range(3,9):
        print( n,"is in the range")
    else :
        print("The number is outside the given range.")
test_range(7)
test_range(10)
test_range(4)
```

Output

```
>>> %Run cp.py

  7 is in the range
  The number is outside the given range.
  4 is in the range
```

C. Write Python programs for the following:

1. Write a function called favorite_book() that accepts one parameter, title. The function should print a message, such as One of my favorite books is Alice in Wonderland. Call the function, making sure to include a book title as an argument in the function call.

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def favourite_book(title):
   print(title,"is my favourite book")
favourite_book("The Great Gatsby")
favourite_book("Charlie and the chocolate Factory")
favourite_book("The Maze Runner")
```

OUTPUT:

```
>>> %Run 'task 1.py'

  The Great Gatsby is my favourite book
  Charlie and the chocolate Factory is my favourite book
  The Maze Runner is my favourite book

>>>
```

2. Write a function called max( ), that returns the maxium of three integer numbers.

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def max(a, b, c):
   if a>b and a>c:
      print(a)
   elif b>c and b>a:
      print(b)
   else:
```

```
        print(c)
a=int(input("first number="))
b=int(input("second number="))
c=int(input("third number="))
max(a, b, c)
print("is the maximum of three integers")
```

OUTPUT:

```
>>> %Run 'task 2.py'

   first number=78
   second number=90
   third number=164
   164
   is the maximum of three integers

>>>
```

3. Write a Python program to find GCD of two numbers

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def GCD(a,b):
    if a>b:
        c=a
    else:
        c=b
    for x in range(c,0,-1):
        if a%x==0 and b%x==0:
            return x

a=int(input("Enter any number:"))
b=int(input("Enter second number:"))
gcd=GCD(a,b)
print(gcd,"is the gcd among",a,"and",b)
```

OUTPUT:

```
>>> %Run 'task 3.py'

   Enter any number:56
   Enter second number:78
   2 is the gcd among 56 and 78
```

4. Write a function called describe_city() that accepts the name of a city and its country. The function should print a simple sentence, such as Reykjavik is in Iceland. Give the parameter for the country a default value. Call your function for three different cities, at least one of which is not in the default country.

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def describe_city(city,country):
    print(city,"is in",country)
describe_city("Karachi","Pakistan")
describe_city("Bonn","Germany")
```

```
describe_city("Moscow","Russia")
```

OUTPUT:

```
>>> %Run 'task 4.py'

  Karachi is in Pakistan
  Bonn is in Germany
  Moscow is in Russia
```

LAB # 08

LISTING

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code

```
Def max_list( list ):
    max = list[ 0 ]
    for a is in list:
        elif a > max:
            max = a
    return max
print(max_list[1, 2, -8, 0])
```

Output

```
>>> %Run 'kist 2.py'

  2
```

Capital D in def function , elif should be replaced by if and print should not be ther in the call function.

2. Code

```
motorcycles = {'honda', 'yamaha', 'suzuki'}
print(motorcycles)
del motorcycles(0)
print(motorcycles)
```

Output:

```
>>> %Run 'error 3.py'

  ['honda', 'yamaha', 'suzuki']
  ['yamaha', 'suzuki']

>>>
```

In list [] are used istead of {}and in line3 index is also called by [].

3. Code

```
Def dupe_v1(x):
  y = []
  for i in x:
    if i not in y:
```

```
      y(append(i))
      return y

a = [1,2,3,4,3,2,1]
print a
print dupe_v1(a)
```

Output:

```
>>> %Run 'error 2.py'

  [1, 2, 3, 4, 3, 2, 1]

>>>
```

Capital D in def function , append function is used as (.append) and brackets are missing in print function in line 9

B. What will be the output of the following programs:

1. Code

```
list1= [1,2,4,5,6,7,8]
print("Negative Slicing:",list1[-4:-1])
x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print("Odd number:", x[::2])
```

Output

```
>>> %Run list.py

  Negative Slicing: [5, 6, 7]
  Odd number: [1, 3, 5, 7, 9]
```

2. Code

```
def multiply_list(elements):
    t = 1
    for x in elements:
        t*= x
    return t
print(multiply_list([1,2,9]))
```

Output

```
>>> %Run K:

  18

>>>
```

3. Code

```
def add(x,lst=[] ):
    if x not in lst:
        lst.append(x)
        return lst

def main():
    list1 = add(2)
    print(list1)
    list2 = add(3, [11, 12, 13, 14])
    print(list2)
main()
```

Output

```
>>> %Run -K list 2.py

  [2]
  [11, 12, 13, 14, 3]

>>>
```

C. Write Python programs for the following:

1. Write a program that store the names of a few of your friends in a list called 'names'. Print each person's name by accessing each element in the list, one at a time.

SOURCE CODE:

```
print("\t ***Names of my Friends***")
names=["Talal Moin","Abdul Moiz Khan","Shaheer Khan Qureshi","Komal Shakeel","Tooba Shakeel","Asma Hashim Khan"]

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

OUTPUT:

```
>>> %Run list.py

          ***Names of my Friends***
  Talal Moin
  Abdul Moiz Khan
  Shaheer Khan Qureshi
  Komal Shakeel
  Tooba Shakeel
  Asma Hashim Khan
```

2. Write a program that make a list that includes at least four people you'd like to invite to dinner. Then use your list to print a message to each person, inviting them to dinner. But one of your guest can't make the dinner, so you need to send out a new set of invitations. Delete that person on your list, use ***del statement*** and add one more person at the same specified index, use the ***insert( )*** method. Resend the invitation.

CODE:

```
#Sending Invitation to the guests
print("***\tInvitation to  the Guests\t***\n")
guest_list=["Talal ","Abdul Moiz Khan","Shaheer","Owais Ahmed"]
for i in guest_list:
    print("->",i,"I invite u on dinner at 10:00 clock  ")

#A guest cannot come to the dinner
print("\n\n>>>",guest_list[3],"can't come at dinner <<<\n\n")

#adding a guest to the list and removing the old guest
guest_list[guest_list.index("Owais Ahmed")]="Naveed"
print("***\tResending Invitation to  the Guests\t***\n")
for x in guest_list:
    print("->",x,"I invite u on dinner at 10:00 clock")
```

OUTPUT:

```
>>> %Run 'task 2.py'

***     Invitation to  the Guests:      ***

-> Talal  I invite u on dinner at 10:00 clock
-> Abdul Moiz Khan I invite u on dinner at 10:00 clock
-> Shaheer I invite u on dinner at 10:00 clock
-> Owais Ahmed I invite u on dinner at 10:00 clock


>>> Owais Ahmed can't come at dinner <<<


***     Resending Invitation to  the Guests:    ***

-> Talal  I invite u on dinner at 10:00 clock
-> Abdul Moiz Khan I invite u on dinner at 10:00 clock
-> Shaheer I invite u on dinner at 10:00 clock
-> Naveed I invite u on dinner at 10:00 clock

>>>
```

3. Write a program that take list = [30, 1, 2, 1, 0], what is the list after applying each of the following statements? Assume that each line of code is independent.

- list.append(40)

  Code:

```
list = [30, 1, 2, 1, 0]
list.append(40)
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 1, 2, 1, 0, 40]
```

- list.remove(1)

Code:

```
list = [30, 1, 2, 1, 0]
list.remove(1)
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 2, 1, 0]
```

- list.pop(1)

Code:

```
list = [30, 1, 2, 1, 0]
list.pop(1)
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 2, 1, 0]
```

- list.pop()

Code:

```
list = [30, 1, 2, 1, 0]
list.pop()
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 1, 2, 1]
```

- list.sort()

Code:

```
list = [30, 1, 2, 1, 0]
list.sort()
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [0, 1, 1, 2, 30]
```

- list.reverse()

Code:

```
list = [30, 1, 2, 1, 0]
list.reverse()
```

```
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [0, 1, 2, 1, 30]

>>>
```

4. Write a program to define a function called 'printsquare' with no parameter, take first 7 integer values and compute their square and stored all square values in the list.

CODE:

```
def printsquare():
    a=list()
    for i in range(1,8):
        a.append(i**2)
    print(a)

printsquare()
```

OUTPUT:

```
>>> %Run 'task 2.py'

  *Following are the squares of the first seven integers stored in a list*
  [1, 4, 9, 16, 25, 36, 49]
```

LAB # 09

SEARCHING & SORTING

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code

```
'apple' is in ['orange', 'apple', 'grape']
```

Output

Print Statement is missing

```
>>> %Run errorr.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\Lab10\errorr.py", line 1
      'apple' is in ['orange', 'apple', 'grape']
                  ^
  SyntaxError: invalid syntax
```

2. Code

```
def countX(lst, x):
    return lst.count(x)
```

Output:

X is causing an error in the calling function countX

```
>>> %Run errorr.py
>>>
```

What will be the output of the following programs:

1. Code

```
strs = ['aa', 'BB', 'zz', 'CC']
print (sorted(strs))
print (sorted(strs, reverse=True))
```

Output

```
>>> %Run er1.py

  ['BB', 'CC', 'aa', 'zz']
  ['zz', 'aa', 'CC', 'BB']

>>>
```

2. Code

```
test_list = [1, 4, 5, 8, 10]
print ("Original list : " , test_list)

# check sorted list

if(test_list == sorted(test_list)):
    print ("Yes, List is sorted.")
else :
    print ("No, List is not sorted.")
```

Output

```
>>> %Run er2.py

  Original list :  [1, 4, 5, 8, 10]
  Yes, List is sorted.

>>>
```

C. Write Python programs for the following:

1. Write a program that take function which implements linear search. It should take a list and an element as a parameter, and return the position of the element in the list. The algorithm consists of iterating over a list and returning the index of the first occurrence of an item once it is found. If the element is not in the list, the function should return 'not found' message.

Code:

```
list = [1, 3, 5, 7, 9] #list of odd numbers
print(list)
num = int(input("Enter Any number from the above list :")) #selection of number
length = len(list)

def search(list , length, num): # to search the entered number in the list
```

```
    for i in range(0, length):
        if (list[i] == num):
            return i
    return 0

result = search(list, length, num)

if(result == 0):
    print("Not found in the above list")
else:
    print("found at index", result)
```

Output:

```
>>> %Run 'task 1.py'

  [1, 3, 5, 7, 9]
  Enter Any number from the above list :3
  found at index 1

>>> %Run 'task 1.py'

  [1, 3, 5, 7, 9]
  Enter Any number from the above list :4
  Not found in the above list
```

2. Write a program that create function that takes two lists and returns True if they have at least one common member. Call the function with atleast two data set for searching.

Code:

```
def common_element(list1, list2):
    result1 = False
    for x in list1:
        for y in list2:
            if x == y:
                result = True
                return result
#For list 1
print(common_element([1,2,3,4,5,6,7,8,9,10], [2,4,6,8,10]))
#For list 2
print(common_element([2,4,6,8,10], [1,3,5,7,9]))
```

Output:

```
>>> %Run 'Task 2.py'

  True
  None

>>>
```

3. Write a program that create function that merges two sorted lists and call two list with random numbers.

Code:

```
def merging(list1, list2):
    print(list1)
    print(list2)
    list3 = list1 + list2
    list3.sort()
    print(list3)
merging([2,10,4,8],[1,9,3,5,6])
```

Output:

```
>>> %Run 'task 3.py'

  [2, 10, 4, 8]
  [1, 9, 3, 5, 6]
  [1, 2, 3, 4, 5, 6, 8, 9, 10]

>>>
```

LAB # 10

TUPLE AND DICTIONARY

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code

```
t = (1, 2, 3)
t.append(4)
t.remove(0)
del tup[0]
```

Output

Tuple does not support item deletion.

```
>>> %Run er1.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\Lab10\er1.py", line 2, in <module>
      t.append(4)
  AttributeError: 'tuple' object has no attribute 'append'

>>>
```

2. Code

```
1user_0=['username':'efermi','first':'enrico','last':'fermi',]
for key, value in 1user_0.items():
    print("\nKey: " ,key)
    print("Value: " ,value)
```

Output:

Dictionary is used in []

Integer at the beginning

```
>>> %Run er1.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\Lab10\er1.py", line 1
      1user_0=['username':'efermi','first':'enrico','last':'fermi',]
              ^
  SyntaxError: invalid syntax
```

What will be the output of the following programs:

1. Code

```
tuple1 = ("green", "red", "blue")
tuple2 = tuple([7, 1, 2, 23, 4, 5])
tuple3 = tuple1 + tuple2
```

```
print(tuple3)
tuple3 = 2 * tuple1
print(tuple3)
print(tuple2[2 : 4])
print(tuple1[-1])
```

Output

```
>>> %Run er1.py

  ('green', 'red', 'blue', 7, 1, 2, 23, 4, 5)
  ('green', 'red', 'blue', 'green', 'red', 'blue')
  (2, 23)
  blue
```

2. Code

```
def main():
    d = {"red":4, "blue":1, "green":14, "yellow":2}
    print(d["red"])
    print(list(d.keys()))
    print(list(d.values()))
    print("blue" in d)
    print("purple" in d)
    d["blue"] += 10
    print(d["blue"])
main() # Call the main function
```

Output

```
>>> %Run er1.py

  4
  ['red', 'blue', 'green', 'yellow']
  [4, 1, 14, 2]
  True
  False
  11
```

C. Write Python programs for the following:

1. Write a program that create a buffet-style restaurant offers only five basic foods. Think of five simple foods, and store them in a tuple. (Hint:Use a for loop to print each food the restaurant offers. Also the restaurant changes its menu, replacing two of the items with different foods and display the menu again.

CODE :

```
print("->Welcome to The Restaurant<- \n\n ")
menu = ('Fish','Fried Rice' ,'Biryani','Chicken Karhai','Jalfrezi')#Menu
print("Menu is as Followed:\n")
for item in menu:
 print("- " , item)
menu = ('Prawns', 'Daal','Biryani','Chicken Karhai','Jalfrezi')#Updated
```

```
print("\nUpdated Menu is as Followed \n")
for item in menu:
 print("- " , item)
```

OUTPUT:

```
>>> %Run 'Task 1.py'

  ->Welcome to The Restaurant<-


  Menu is as Followed:

  -  Fish
  -  Fried Rice
  -  Biryani
  -  Chicken Karhai
  -  Jalfrezi

  Updated Menu is as Followed

  -  Prawns
  -  Daal
  -  Biryani
  -  Chicken Karhai
  -  Jalfrezi
```

2. Write a program for “Guess the capitals” using a dictionary to store the pairs of states and capitals so that the questions are randomly displayed. The program should keep a count of the number of correct and incorrect responses.

CODE :

```
#Program for Guess the capitals by using a dictionary
Guess = {"Pakistan" : "Islamabad","Russia" : "Moscow","Germany" : "Bonn"}
valid=0
invalid=0
for Country , Capital in Guess.items():
    print("\t Give The Capital Of The City = " ,Country)
    ans= input("Write Answer : " )
    if ans == Capital:
        print("*Your Answer Is Correct*" )
        valid=valid+1

    else:
        print("*Your Answer Is Invalid*")
        invalid=invalid+1

print("corrects: ",valid)
print("In-corrects: ",invalid)
```

OUTPUT:

```
>>> %Run 'task 10 2.py'

          Give The Capital Of The City =  Pakistan
  Write Answer : Islamabad
  *Your Answer Is Correct*
          Give The Capital Of The City =  Russia
  Write Answer : Munich
  *Your Answer Is Invalid*
          Give The Capital Of The City =  Germany
  Write Answer : Bonn
  *Your Answer Is Correct*
  corrects:  2
  In-corrects:  1
```

3. Write a pogram that make a dictionary called favorite_places. Think of three names to use as keys in the dictionary, and store three favorite places for each person through list. Loop through the dictionary, and print each person's name and their favorite places.
Output look alike:
abc likes the following places:
- Bear Mountain
- Death Valley
- Tierra Del Fuego

CODE :

```
favorite_places= {'Moiz': ['Niagra Fall', 'Dream World', 'Maldives']
,'\nShaheer': ['Tibetan plateau','Taj Mahal' ,'badshahi masjid']
,'\nTalal': ['Burj Khalifa', 'Eifel tower', 'Devils Point']}
for names, places in favorite_places.items():
 print(names, " likes the following places:")
 for place in places:
    print("- " , place)
```

OUTPUT:

```
>>> %Run 'task 3.py'

  Moiz  likes the following places:
  -  Niagra Fall
  -  Dream World
  -  Maldives

  Shaheer  likes the following places:
  -  Tibetan plateau
  -  Taj Mahal
  -  badshahi masjid

  Talal  likes the following places:
  -  Burj Khalifa
  -  Eifel tower
  -  Devils Point
```

LAB # 11

MODULES AND PACKAGES

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code:

```
import sys as s
print(sys.executable)
print(sys.getwindowsversion())
```

Output:

Sys is written instead of s.

```
>>> %Run cp2.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\LAb 11\cp2.py", line 2, in <module>
      print(sys.executable)
  NameError: name 'sys' is not defined
```

2. Code:

```
import datetime
from datetime import date
import times
# Returns the number of seconds
print(time.time())
# Converts a number of seconds to a date object
print(datetime.datetime.now())
```

Output:

Times is written instead of time.

```
>>> %Run cp2.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\LAb 11\cp2.py", line 3, in <module>
      import times
  ModuleNotFoundError: No module named 'times'
```

3. Code:

```
From math import math
# using square root(sqrt) function contained
```

```
print(Math.sqrt(25) )
print(Math.pi)
# 2 radians = 114.59 degreees
print(Math.degrees(2))
```

Output:

```
>>> %Run cp2.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\LAb 11\cp2.py", line 1
      From math import math
               ^
  SyntaxError: invalid syntax
```

Math is causing error in line 1

B. What would be the output of the following programs:

1. Code:

```
import calendar
yy = 2017
mm = 11
# display the calendar
print(calendar.month(yy, mm))
```

Output:

```
>>> %Run 'cp 1.py'

     November 2017
  Mo Tu We Th Fr Sa Su
           1  2  3  4  5
   6  7  8  9 10 11 12
  13 14 15 16 17 18 19
  20 21 22 23 24 25 26
  27 28 29 30
```

2. Code:

```
import sys
print(sys.argv)
for i in range(len(sys.argv)):
   if i==0:
       print("The function is",sys.argv[0])
   else:   print("Argument:",sys.argv[i])
```

Output:

```
>>> %Run cp2.py

  ['cp2.py']
  The function is cp2.py
```

3. Code:

```
import numpy as np
# Creating array object
arr = np.array( [[ 1, 2, 3],
                 [ 4, 2, 5]] )

# Printing array dimensions (axes)
print("No. of dimensions: ", arr.ndim)

# Printing shape of array
print("Shape of array: ", arr.shape)

# Printing size (total number of elements) of array
print("Size of array: ", arr.size)
```

Output:

```
>>> %Run cp2.py

  No. of dimensions:  2
  Shape of array:  (2, 3)
  Size of array:  6
```

C. Write Python programs for the following:

1. Write a NumPy program to create an 1D array of 10 zeros, 10 ones, 10 fives

CODE:

```
import numpy as npy
Ones=npy.ones(10)
Zeros=npy.zeros(10)
Fives=npy.ones(10)*5
print("10 Ones Of 1 Dimension Arrays \n ",Ones,'\n')
print("10 Zeros Of 1 Dimension Arrays \n",Zeros,'\n')
print("10 Fives Of 1 Dimension Arrays \n",Fives,'\n')
```

OUTPUT:

```
>>> %Run task1.py

  10 Ones Of 1 Dimension Arrays
    [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

  10 Zeros Of 1 Dimension Arrays
   [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

  10 Fives Of 1 Dimension Arrays
   [5. 5. 5. 5. 5. 5. 5. 5. 5. 5.]
```

2. Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.

CODE:

```
import numpy as npy
Mat=npy.arange(1,10).reshape(3,3)
print("A Matrix of 3x3 Matrix")
print(Mat)
```

OUTPUT:

```
>>> %Run 'task 2.py'

  A Matrix of 3x3 Matrix
  [[1 2 3]
   [4 5 6]
   [7 8 9]]
```

LAB # 12

STRINGS

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code:

```
a = "PYTHON"
a[0] = "x"
#Apply Exception for mention error
```

Output:

Str function is not used.

```
>>> %Run cp1.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\Lab 12\cp1.py", line 2, in <module>
      a[0] = "x"
  TypeError: 'str' object does not support item assignment
```

2. Code:

```
a = STRING
i = 0
while i < len(b):
    c = a[i]
    print(c)
    i+=i + 1
```

Output:

B is written instead of a.

```
>>> %Run cp1.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\Lab 12\cp1.py", line 1, in <module>
      a = STRING
  NameError: name 'STRING' is not defined
```

3. Code:

```
Def 1my_function(x):
return x[::-1]
```

```
mytxt = 1my_function("I wonder how this text looks like backwards")
print(mytxt)
```

Output:

Numeric value in defining function is causing an error.

```
>>> %Run cp1.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\Lab 12\cp1.py", line 1
      Def 1my_function(x):
          ^
  SyntaxError: invalid syntax
```

B. What would be the output of the following programs:

1. Code:

```
s= "Welcome"
for i in range(0, len(s), 2):
    print(s[i], end = '')
```

Output:

```
>>> %Run cp
  Wloe
```

2. Code:

```
s = input("Enter a string: ")
if "Python" in s:
    print("Python", "is in", s)
else:
    print("Python", "is not in", s)
```

Output:

```
>>> %Run cp1.py
  Enter a string: If it is not working
  Python is not in If it is not working
>>>
```

3. Code:

```
str='cold'
list_enumerate=list(enumerate(str))
print("list enumerate:", list_enumerate)
print("list length:", len(str))
s1 = "Welcome to Python"
s2 = s1.replace("o","abc")
```

```
print(s2)
a = "Python" + "String"
b = "<" + a*3 + ">"
print(b)
```

Output:

```
>>> %Run cp1.py

  list enumerate: [(0, 'c'), (1, 'o'), (2, 'l'), (3, 'd')]
  list length: 4
  Welcabcme tabc Pythabcn
  <PythonStringPythonStringPythonString>
```

C. Write Python programs for the following:

1. Write a program that Store a person's name, and include some whitespace characters at the beginning and end of the name. Make sure you use each character combination, "\t" and "\n", at least once. Print the name once, so the whitespace around the name is displayed. Then print the name using each of the three stripping functions, lstrip(),rstrip(), and strip().

CODE:

```
name=input('Enter Name: ')
print(name,'\n')
LStp_name=name.lstrip( '   Abdul')
RStp_name=name.rstrip( ' Chishti')
Stp_name=name.strip( ' Abdul Chishti')
print('Lstripped Name:')
print(LStp_name,'\n')
print('Rstripped Name:')
print(RStp_name,'\n')
print('Stripped Name:')
print(Stp_name,'\n')
```

OUTPUT:

```
Enter Name: Abdul Moiz Chishti
Abdul Moiz Chishti

Lstripped Name:
Moiz Chishti

Rstripped Name:
Abdul Moiz

Stripped Name:
Moiz
```

2. Write a program that asks the user for their favourite color. Create the following output (assuming blue is the chosen color) (hint: use '+' and '*')

```
blueblueblueblueblueblueblueblueblueblue
blue                                blue
blueblueblueblueblueblueblueblueblueblue
```

CODE:

```
clr=input('Enter a colour: ')
print('\t\t',clr*10)
print('\t\t',clr,"\t\t\t   ",clr)
print('\t\t',clr*10)
```

OUTPUT:

```
>>> %Run 'task 2.py'

  Enter a colour: red
                   redredredredredredredredredred
                   red                        red
                   redredredredredredredredredred

>>>
```

LAB # 13

FILE PROCESSING

OBJECTIVE

To explore methods to access, open/close , modes contained in external files

THEORY

When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.

File Handling in Python
Python allows users to handle files to read and write files, along with many other file handling options. Python treats file differently as **text** or **binary** and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun..

File Operations
There are different operations that can be carried out on a file, these are:

- Creation of a new file
- Opening an existing file
- Reading from a file
- Writing to a file
- Closing a file

Open() function
Open () function in Python to open a file in read or write mode.

syntax: open(Directory:\\filename, mode)

There are various kinds of mode, that python provides:

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)
'rb'	Opens a file for reading binary data.
'wb'	Opens a file for writing binary data.

Example

```
# a file named "python", will be opened with the reading mode
file = open('python.txt', 'r')
# This will print every line one by one in the file
for each in file:
     print (each)
```

Read() mode

There is more than one way to read a file in Python. To extract a string that contains all characters in the file then it can be use as;

syntax: file.read()

Example

```
# Python code to illustrate read() mode
file = open("python.text", "r")
print file.read()
```

Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:

Example

```
# Python code to illustrate read() mode character wise
file = open("python.txt", "r")
print file.read(5)
```

Write() mode

To manipulate the file, write the following in your Python environment:

syntax: file.write()

The close() command terminates all the resources in use and frees the system of this particular program.

syntax: file.close()

Example:

```
# Python code to create a file
file = open(' python.txt','w')
file.write("This is the write command")
file.write("It allows us to write in a particular file")
file.close()
```

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code

```
file=open('python.txt','r')
print("using read() mode character wise:")
s1=file.read(19)
print(s1)
```

Output:

```
>>> %Run er.py

  using read() mode character wise:
  Loop statements usu

>>>
No error
```

2. Code

```
f1=open("jj",)
f1.write("something")
```

Output:

```
Extension of file is missing.
>>> %Run er.py
  Traceback (most recent call last):
    File "E:\Semester 1\P fund\Lab 13\er.py", line 1, in <module>
      f1=open("jj",)
  FileNotFoundError: [Errno 2] No such file or directory: 'jj'

>>>
```

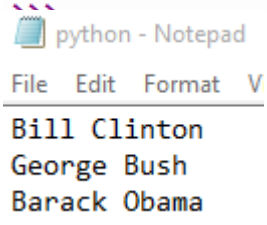
B. Creat a text file named python, Write the following code. Execute it and show the output. (You can use the Snipping Tool to take a snapshot of your txt.file)

1. Code

```
def main():
   # Open file for output
       outfile=open('D:\\python.txt','w')
   # Write data to the file
      outfile.write("Bill Clinton\n")
      outfile.write("George Bush\n")
       outfile.write("Barack Obama")
       print(outfile)
        # Close the output file
      outfile.close()
main()
```

Output:

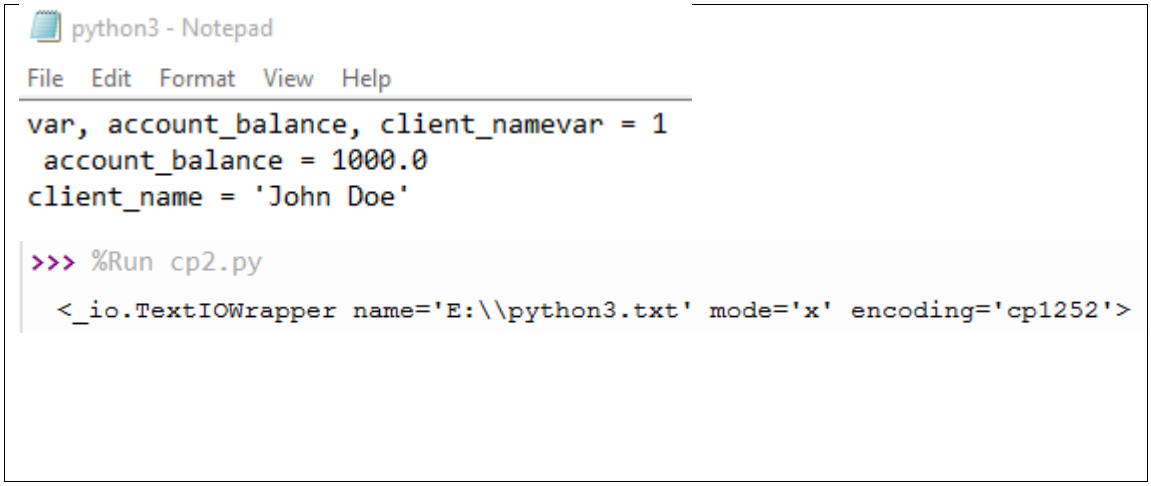
```
>>> %Run cp1.py
  <_io.TextIOWrapper name='E:\\python.txt' mode='w' encoding='cp1252'>
```



2. Code

```
def main():
# Open file for output
    outfile=open('D:\\python.txt','x')
# Write data to the file
    outfile.write("var, account_balance, client_name")
    outfile.write("var     =     1\n     account_balance     =
1000.0\nclient_name = 'John Doe'")
    print(outfile)
    # Close the output file
    outfile.close()
main()
```

Output:



C. Write Python programs for the following:

1. Write a program that create a function called “file_read”, to read an entire text file.

CODE:

```
file_read=open('file_read.txt','r')
for each in file_read:
    print(each)
```

OUTPUT:

2. Write a program that reads the content and replace any word in a string with different word.

Example:
Replace ‘dog’ with ‘cat’in a sentence

CODE:

```
string="The red house is between the blue house and the old house"
```

```
print(string)
print(string.replace("house","car"))
```

OUTPUT:

```
>>> %Run 'task 2.py'

  The red house is between the blue house and the old house
  The red car is between the blue car and the old car

>>>
```

3. Write a program that prompts the user for their name. When they respond, write their name to a file called guest.txt.

CODE:

```
user=input("Enter name")
file = open(' python.txt','w')
file.write(user)


file.close()
```

OUTPUT: