Experiment 04

Write a python program to understand User defined and Anonymous functions.

Roll No.	61
Name	V Krishnasubramaniam
Class	D10-A
Subject	Python Lab
LO Mapped	LO1: Understand the structure, syntax, and semantics of the Python language LO2: Interpret advanced data types and functions in python

Aim:

Write a python program to understand User defined and Anonymous functions.

Introduction:

Functions in Python

A function is a block of organized, reusable code that is used to perform a single, related action. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. Furthermore, it avoids repetition and makes code reusable. Python gives you many built-in functions like print (), etc. But you can also create your own functions. These functions are called user-defined functions.

1. You can define functions to provide the required functionality. Here are simple rules to define a function in Python.
2. Function blocks begin with the keyword def followed by the function name and parentheses ().
3. Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
4. The first statement of a function can be an optional statement - the documentation string of the function or docstring.
5. The code block within every function starts with a colon (:) and is indented.
6. The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax:

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

Example:

```
def evenOdd(x):
   if (x % 2 == 0):
      print "even"
   else:
      print "odd"

evenOdd(2)
evenOdd(3)
```

Output:

```
even
odd
```

The first string after the function is called the Document string or Docstring in short. This is used to describe the functionality of the function. The use of docstring in functions is optional but it is considered a good practice.

The below syntax can be used to print out the docstring of a function:
print(function_name.__doc__)

Example:

```
def say_Hi():
   "Hello World!"
print(say_Hi.__doc__)
```

Output:

Hello World!

The return statement is used to exit from a function and go back to the function caller and return the specified value or data item to the caller.

Syntax:

return [expression_list]

The return statement can consist of a variable, an expression, or a constant which is returned to the end of the function execution. If none of the above is present with the return statement a None object is returned.

Example:

```
def square_value(num):
   """This function returns the square value of the entered number"""
   return num**2
print(square_value(2))
```

Output:

4

It is possible to define one function inside another function.

Example:

```
def outer(num1):
        def inner_increment(num1): # hidden from outer code
                return num1 + 1
        num2 = inner_increment(num1)
        print(num1, num2)
outer(10)
```

Output:

10 11

It is possible to pass function as parameter to another function.

Example:

```
def display(fun):
        return ‘Hi’+fun
def message():
        return ‘How are u?’
print(display(message()))
```

Output:

Hi How are u?

Function can return other functions.

Example:

```
def display ():
        def message():
        return ‘How are you?’
        return message
fun=display()
print(fun())
```

Output:

How are you?

In Python every variable name is a reference. When we pass a variable to a function, a new reference to the object is created. Parameter passing in Python is the same as reference passing in Java.

Example:

```
def myFun(x):
        x[0] = 20
lst = [10, 11, 12, 13, 14, 15]
myFun(lst)
print(lst)
```

Output:

[20, 11, 12, 13, 14, 15]

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 printme( str ):
        "This prints a passed string into this function"
        print str
        return;
printme("abc")
```

Output:

abc

Keyword arguments are related to the function calls. When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name. This allows you to skip 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 printme( str ):
        "This prints a passed string into this function"
        print str
        return;
printme( str = "My string")
```

Output:

My String

You may need to process a function for more arguments than you specified while defining the function. These arguments are called variable-length arguments and are not named in the function definition, unlike required and default arguments.

Example:

```
def printinfo( arg1, *vartuple ):
        "This prints a variable passed arguments" 10
        print "Output is: "
        print arg1
        for var in vartuple:
        print var
        return;
printinfo( 10 )
printinfo( 70, 60, 50 )
```

Output:

```
Output is:
10
Output is:
50
60
70
```

Anonymous Functions in Python

In Python, an anonymous function means that a function is without a name. As we already know that the def keyword is used to define a normal function in Python. Similarly, the lambda keyword is used to define an anonymous function in Python.

Syntax:

lambda arguments: expression

1. These functions are called anonymous because they are not declared in the standard manner by using the def keyword. You can use the lambda keyword to create small anonymous functions.
2. An anonymous function cannot be a direct call to print because lambda requires an expression
3. Lambda functions have their own local namespace and cannot access variables other than those in their parameter list and those in the global namespace.
4. Although it appears that lambdas are a one-line version of a function, they are not equivalent to inline statements in C or C++, whose purpose is by passing function stack allocation during invocation for performance reasons.

Example:

```
def cube(y):
   return y*y*y
lambda_cube = lambda y: y*y*y
print(cube(5))
print(lambda_cube(5))
```

Output:

```
125
125
```

Lambda functions can be used along with built-in functions like filter(), map() and reduce().

Using lambda() Function with filter(): The filter() function in Python takes in a function and a list as arguments. This offers an elegant way to filter out all the elements of a sequence “sequence”, for which the function returns True.

Example:

```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
final_list = list(filter(lambda x: (x%2 != 0) , li))
print(final_list)
```

Output:

```
[5, 7, 97, 77, 23, 73, 61]
```

Using lambda() Function with map(): The map() function in Python takes in a function and a list as an argument. The function is called with a lambda function and a list and a new list is returned which contains all the lambda modified items returned by that function for each item.

Example:

```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
final_list = list(map(lambda x: x*2, li))
print(final_list)
```

Output:

[10, 14, 44, 194, 108, 124, 154, 46, 146, 122]

Using lambda() Function with reduce(): The reduce() function in Python takes in a function and a list as an argument. The function is called with a lambda function and an iterable and a new reduced result is returned. This performs a repetitive operation over the pairs of the iterable. The reduce() function belongs to the functools module.

Example:

```
from functools import reduce
li = [5, 8, 10, 20, 50, 100]
sum = reduce((lambda x, y: x + y), li)
print (sum)
```

Output:

```
193
```

Recursion in Python

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily.

In the recursive program, the solution to the base case is provided and the solution of the bigger problem is expressed in terms of smaller problems.

Example:

```
def recursive_fibonacci(n):
    if n <= 1:
        return n
    else:
        return(recursive_fibonacci(n-1) + recursive_fibonacci(n-2))
n_terms = 5
if n_terms <= 0:
    print("Invalid input! Please input a positive value")
else:
    print("Fibonacci series:")
    for i in range(n_terms):
    print(recursive_fibonacci(i))
```

Output:

```
0
1
1
2
3
```

A function fun is called direct recursive if it calls the same function fun. A function fun is called indirect recursive if it calls another function say fun_new and fun_new calls fun directly or indirectly.

A unique type of recursion where the last procedure of a function is a recursive call. The recursion may be automated away by performing the request in the current stack frame and returning the output instead of generating a new stack frame. The tail-recursion may be optimized by the compiler which makes it better than non-tail recursive functions.

Example:

```
def Recur_facto(n):
        if (n == 0):
                return 1
        return n * Recur_facto(n-1)
print(Recur_facto(6))
```

Output:

720

Results

Anonymous Functions:

1. Python Program to Display Powers of 2 Using Anonymous Function

Program in powersOfTwo.py:

```
terms = 10
result = list(map(lambda x: 2 ** x, range(terms)))

print("The total terms are:",terms)
for i in range(terms):
   print("2 raised to power",i,"is",result[i])
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python powersOfTwo.py
The total terms are: 10
2 raised to 0  =  1
2 raised to 1  =  2
2 raised to 2  =  4
2 raised to 3  =  8
2 raised to 4  =  16
2 raised to 5  =  32
2 raised to 6  =  64
2 raised to 7  =  128
2 raised to 8  =  256
2 raised to 9  =  512
```

2. Python Program to Find Numbers Divisible by Another Number using Anonymous Function

Program in divisibility.py:

```
num = int(input("Enter a number:"))
my_list = range(150)
result = list(filter(lambda x: (x % num == 0), my_list))
print("Numbers divisible by",num,"are",result)
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python divisibility.py
Enter a number:13
Numbers divisible by 13 are [0, 13, 26, 39, 52, 65, 78, 91, 104, 117, 130, 143]

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python divisibility.py
Enter a number:15
Numbers divisible by 15 are [0, 15, 30, 45, 60, 75, 90, 105, 120, 135]
```

3. Python Program to Convert Decimal to Binary, Octal and Hexadecimal using Anonymous Function

Program in convertBase.py:

```
num = int(input("Enter a decimal number: "))
bin=lambda x : format(x,'b')
print("Equivalent binary number is",bin(num))
oct=lambda x : format(x,'o')
print("Equivalent octal number is",oct(num))
hex=lambda x : format(x,'x')
print("Equivalent hexadecimal number is",hex(num))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python convertBase.py
Enter a decimal number: 10
Equivalent binary number is 1010
Equivalent octal number is 12
Equivalent hexadecimal number is a

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python convertBase.py
Enter a decimal number: 273
Equivalent binary number is 100010001
Equivalent octal number is 421
Equivalent hexadecimal number is 111
```

7. Python Program to Find Factors of Number Using Anonymous Function

Program in getFactors.py:

```
num = int(input("Enter a number: "))
my_list = []
for i in range(1, num + 1):
   my_list.append(i)
new_list = list(filter(lambda x: (num % x == 0), my_list))
```

```
print("Factors of",num,":",new_list)
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getFactors.py
Enter a number: 30
Factors of 30 : [1, 2, 3, 5, 6, 10, 15, 30]

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getFactors.py
Enter a number: 15
Factors of 15 : [1, 3, 5, 15]
```

User Defined Functions:

4. Python Program to Find ASCII Value of Character

Program in getAscii.py:

```
def getAscii(chr):
  return ord(chr)

char = input("Enter a character: ")
print("The ASCII value of '" + char + "' is", getAscii(char))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getAscii.py
Enter a character: e
The ASCII value of 'e' is 101

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getAscii.py
Enter a character: 0
The ASCII value of '0' is 48

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getAscii.py
Enter a character: C
The ASCII value of 'C' is 67
```

5. Python Program to Find HCF or GCD

Program in getHcf.py:

```
def computeGCD(x, y):
   while(y):
      x, y = y, x % y
   return x
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
print ("GCD of",a,"and",b,"=",computeGCD(a,b))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getHcf.py
Enter the first number: 8
Enter the second number: 12
GCD of 8 and 12 = 4

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getHcf.py
Enter the first number: 45
Enter the second number: 40
GCD of 45 and 40 = 5
```

6. Python Program to Find LCM

Program in getLcm.py:

```
from getHcf import computeGCD
def compute_lcm(x, y):
   lcm = (x*y)//computeGCD(x,y)
   return lcm
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("LCM of",num1,"and",num2,"=", compute_lcm(num1, num2))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getLcm.py
Enter first number: 3
Enter second number: 5
LCM of 3 and 5 = 15

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python getLcm.py
Enter first number: 8
Enter second number: 12
LCM of 8 and 12 = 24
```

8. Python Program to Make a Simple Calculator

Program in calc.py:

```
def add(x, y):
   return x + y
def subtract(x, y):
   return x - y
def multiply(x, y):
   return x * y
def divide(x, y):
   return x / y
while True:
   print()
   print("Select operation: ")
   print("1. Add")
```

```
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")
    print("5. Exit")
    choice = input("Enter choice(1/2/3/4): ")
    print()
    if choice in ('1', '2', '3', '4', '5'):
        if choice == '5':
            break
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))
        if choice == '1':
            print(num1, "+", num2, "=", add(num1, num2))
        elif choice == '2':
            print(num1, "-", num2, "=", subtract(num1, num2))
        elif choice == '3':
            print(num1, "*", num2, "=", multiply(num1, num2))
        elif choice == '4':
            print(num1, "/", num2, "=", divide(num1, num2))
        else:
            exit()
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python calc.py

Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter choice(1/2/3/4): 1

Enter first number: 1
Enter second number: 3
1.0 + 3.0 = 4.0
```

```
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter choice(1/2/3/4): 2

Enter first number: 1
Enter second number: 3
1.0 - 3.0 = -2.0

Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter choice(1/2/3/4): 3

Enter first number: 1
Enter second number: 3
1.0 * 3.0 = 3.0

Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter choice(1/2/3/4): 4

Enter first number: 1
Enter second number: 3
1.0 / 3.0 = 0.3333333333333333

Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter choice(1/2/3/4): 5
```

9. Python Program to Display Calendar

Program in myCalendar.py:

```
import calendar
year = int(input("Enter year: "))
month = int(input("Enter month: "))
print(calendar.month(year,month))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python myCalendar.py
Enter year: 2021
Enter month: 3
     March 2021
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 31
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python myCalendar.py
Enter year: 2020
Enter month: 12
   December 2020
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 31
```

10. Python Program to Display Fibonacci Sequence Using Recursion

Program in fibonacci.py:

```
def recur_fibo(n):
   if n <= 1:
      return n
   else:
      return(recur_fibo(n-1) + recur_fibo(n-2))
nterms = int(input("Enter number of terms:"))
if nterms <= 0:
   print("Please enter a positive integer")
else:
   print("Fibonacci sequence:")
   for i in range(nterms):
     print(recur_fibo(i))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python fibonacci.py
Enter number of terms:10
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python fibonacci.py
Enter number of terms:20
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

11. Python Program to Find Sum of Natural Numbers Using Recursion

Program in sumOfNatNum.py:

```
def recur_sum(n):
    if n <= 1:
        return n
    else:
        return n + recur_sum(n-1)
num = int(input("Enter a number: "))
if num < 0:
    print("Enter a positive number")
else:
    print("The sum is",recur_sum(num))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python sumOfNatNum.py
Enter a number: 10
The sum is 55

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python sumOfNatNum.py
Enter a number: 20
The sum is 210
```

12. Python Program to Find Factorial of Number Using Recursion

Program in factorial.py:

```
def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)
num =  int(input("Enter a number: "))
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    print("The factorial of", num, "is", recur_factorial(num))
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python factorial.py
Enter a number: 5
The factorial of 5 is 120

C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python factorial.py
Enter a number: 7
The factorial of 7 is 5040
```

13. Python Program to Convert Decimal to Binary Using Recursion

Program in toBinary.py:

```
def convertToBinary(n):
  if n > 1:
    convertToBinary(n//2)
  print(n % 2,end = '')
dec = int(input("Enter a decimal number: "))
convertToBinary(dec)
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python toBinary.py
Enter a decimal number: 20
10100
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python toBinary.py
Enter a decimal number: 15
1111
```

14. Write a Menu driven program in python to implement simple banking application. Application should read the customer name, account number, initial balance, rate of interest, contact number and address field etc. Application should have following methods.

1. createAccount()
2. deposit()
3. withdraw()
4. computeInterest()
5. displayBalance()

Program in bank.py:

```
global balance
global name
global account_no
global contact_no
global address_field

balance=0.0
name=""
account_no=-1
contact_no=0
```

```
adress_field=0

def deposit():
  global balance
  amount = float(input("Enter amount to be deposited: "))
  balance = amount + balance
  print("Amount Deposited:",amount)

def computeinterest():
  global balance
  roi = int(input("Enter rate of interest: "))
  balance += (balance * roi / 100)
  print("New Balance:",balance)

def withdraw():
  global balance
  amount = float(input("Enter amount to be withdrawn: "))
  if balance >= amount:
    balance -= amount
    print("Withdrew Amount:", amount)
  else:
    print("Insufficient balance ")

def display():
  print("Net Available Balance:",balance)

while True:
  print("1. Create Account")
  print("2. Deposit Amount")
  print("3. Withdraw Amount")
  print("4. Rate of Interest")
  print("5. Display Amount")
  print("6. Exit")
  choice = input("Enter choice(1/2/3/4/5/6): ")
  if choice in ('1', '2', '3', '4', '5', '6'):
    if choice == '1':
      name = input("Enter your name: ")
      account_no = int(input("Enter your account number: "))
      contact_no = int(input("Enter your phone number: "))
      address_field = input("Enter your address: ")
    elif choice == '2':
      if account_no == -1:
        print("Please create account first")
```

```
    else:
      deposit()
  elif choice == '3':
    if account_no == -1:
      print("Please create account first")
    else:
      withdraw()
  elif choice == '4':
    if account_no == -1:
      print("Please create account first")
    else:
      computeinterest()
  elif choice == '5':
    if account_no == -1:
      print("Please create account first")
    else:
      display()
  else:
    exit()
```

Output:

```
C:\Users\vkris\Dropbox\Programming\Python\Experiments\Exp 4>python bank.py
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 2
Please create account first
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 1
Enter your name: Krishna
Enter your account number: 1234567890
Enter your phone number: 9087654321
Enter your address: Thane
```

```
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 2
Enter amount to be deposited: 1000
Amount Deposited: 1000.0
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 3
Enter amount to be withdrawn: 400
Withdrew Amount: 400.0
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 5
Net Available Balance: 600.0
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 4
Enter rate of interest: 3
New Balance: 618.0
```

```
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 5
Net Available Balance: 618.0
1. Create Account
2. Deposit Amount
3. Withdraw Amount
4. Rate of Interest
5. Display Amount
6. Exit
Enter choice(1/2/3/4/5/6): 6
```

15. Write a menu driven code in python which will read a number and should implement the following methods
1. factorial()
2. reverse()
3. testArmstrong()
4. testPalindrome()
5. testPrime()
6. fibonacciSeries()

Program in opsOnNums.py:

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

def reverse(n, r):
  if n==0:
    return r
  else:
    return reverse(n//10, r*10 + n%10)

def testArmstrong(n):
  order = len(str(n))
  sum = 0
  temp = n
  while temp > 0:
```

```
    digit = temp % 10
    sum += digit ** order
    temp //= 10
  return(sum)

def testPalindrome(n):
  rev = 0
  while n > 0:
    dig = n % 10
    rev = rev * 10 + dig
    n = n // 10
  return(rev)

def testPrime(n):
  flag = 0
  if n > 1:
    for i in range(2, n):
      if (n % i) == 0:
        flag = 1
        break
  return(flag)

def fibonacciSeries(n):
  if n <= 1:
    return n
  else:
    return(fibonacciSeries(n-1) + fibonacciSeries(n-2))

while True:
  print("\nSelect operation:")
  print("1. Factorial")
  print("2. Reverse")
  print("3. Test Armstrong number")
  print("4. Test Palindrome number")
  print("5. Test Prime number")
  print("6. Fibonacci Series")
  choice = input("Enter choice(1/2/3/4/5/6): ")
  if choice in ('1', '2', '3', '4', '5', '6'):

    if choice == '1':
      num = int(input("Enter a number: "))
      print("The factorial of", num, "is", factorial(num))
```

```
    elif choice == '2':
      num = int(input("Enter a number: "))
      reversed_number = reverse(num,0)
      print("Reverse of %d is %d" %(num, reversed_number))

    elif choice == '3':
      num = int(input("Enter a number: "))
      new_num = testArmstrong(num)
      if num == new_num:
        print(num,"is an Armstrong number")
      else:
        print(num,"is not an Armstrong number")

    elif choice == '4':
      num = int(input("Enter a number: "))
      new_num = testPalindrome(num)
      if num == new_num:
        print(num,"is a Palindrome number")
      else:
        print(num,"is not a Palindrome number")

    elif choice == '5':
      num = int(input("Enter a number: "))
      flag = testPrime(num)
      if flag == 0:
        print(num,"is a Prime number")
      else:
        print(num,"is not a Prime number")

    elif choice == '6':
      num = int(input("Enter a number: "))
      if num <= 0:
        print("Please enter a positive integer")
      else:
        print("Fibonacci sequence:")
      for i in range(num):
        print(fibonacciSeries(i),end=" ")

    else:
      break
      quit()
```

Output:

```
Select operation:
1. Factorial
2. Reverse
3. Test Armstrong number
4. Test Palindrome number
5. Test Prime number
6. Fibonacci Series
Enter choice(1/2/3/4/5/6): 1
Enter a number: 5
The factorial of 5 is 120

Select operation:
1. Factorial
2. Reverse
3. Test Armstrong number
4. Test Palindrome number
5. Test Prime number
6. Fibonacci Series
Enter choice(1/2/3/4/5/6): 2
Enter a number: 123
Reverse of 123 is 321
Select operation:
1. Factorial
2. Reverse
3. Test Armstrong number
4. Test Palindrome number
5. Test Prime number
6. Fibonacci Series
Enter choice(1/2/3/4/5/6): 3
Enter a number: 153
153 is an Armstrong number

Select operation:
1. Factorial
2. Reverse
3. Test Armstrong number
4. Test Palindrome number
5. Test Prime number
6. Fibonacci Series
Enter choice(1/2/3/4/5/6): 4
Enter a number: 12321
12321 is a Palindrome number
```

```
Select operation:
1. Factorial
2. Reverse
3. Test Armstrong number
4. Test Palindrome number
5. Test Prime number
6. Fibonacci Series
Enter choice(1/2/3/4/5/6): 5
Enter a number: 13
13 is a Prime number

Select operation:
1. Factorial
2. Reverse
3. Test Armstrong number
4. Test Palindrome number
5. Test Prime number
6. Fibonacci Series
Enter choice(1/2/3/4/5/6): 6
Enter a number: 10
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
Select operation:
1. Factorial
2. Reverse
3. Test Armstrong number
4. Test Palindrome number
5. Test Prime number
6. Fibonacci Series
Enter choice(1/2/3/4/5/6): 7
```

Conclusion:

Thus, we have understood the concepts of functions, types of functions, anonymous functions and recursion in Python, along with their practical uses by performing hands-on programs.