1. What is a flow control statement? Discuss **if and if else statements** with flow chart.

2. With Python programming examples to each, explain the syntax and control flow diagrams of **break and continue statements**.

3. Explain TWO ways of **importing modules** into application in Python with syntax and suitable programming examples.

4. How can you prevent a **python program from crashing**? Discuss different ways to avoid crashing.

5. Explain **FOUR scope rules of variables** in Python.

6. Explain **local and Global scope** variable in python with example.

7. What are **functions**? Explain python function with parameters and return statements.

8. Explain string concatenation and replication with examples.

Programs

1. Develop a Python program to generate **Fibonacci sequence** of length (N). Read N from the console.

2. Write a function to calculate **factorial of a number**. Develop a program to **compute binomial coefficient** (Given N and R).

3. Write a function named DivExp which takes TWO parameters a, b and returns a value c (c=a/b). Write suitable assertion for a>0 in function DivExp and raise an exception for when b=0. Dev p a Python program which reads two values from the console and calls a function DivExp.

4. Develop a python program to calculate the area of circle and triangle print the result.

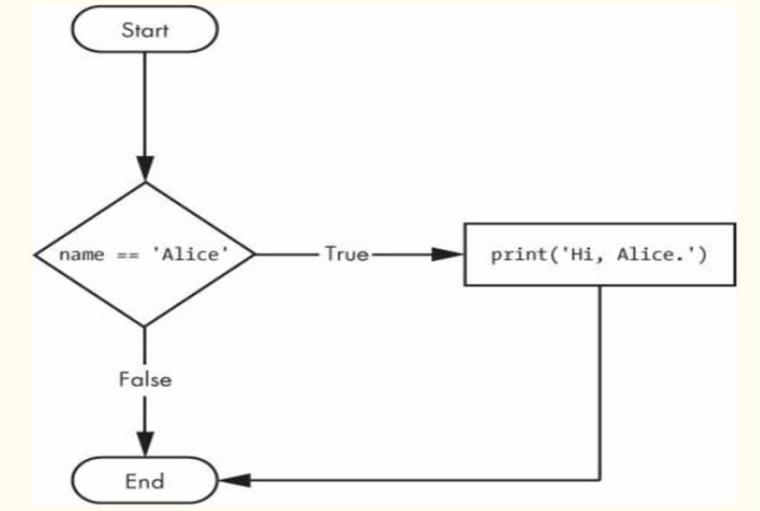
1. **What is a flow control statement? Discuss if and if else statements with flow chart.**

Ans: A program's control flow is the order in which the program's code executes. The control flow of a Python program is regulated by conditional statements, loops, and function calls.

if Statement

The most common type of flow control statement is the if statement. An if statement's clause (that is, the block following the if statement) will execute if the statement's condition is True. The clause is skipped if the condition is False.

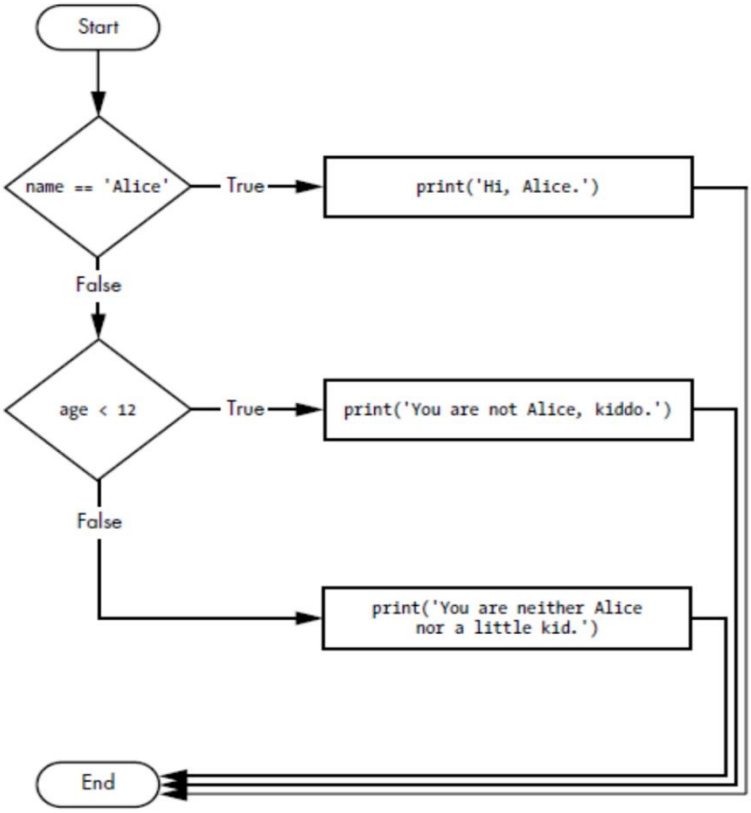
```
if name == 'Alice':
    print('Hi, Alice.')
```



if else statement

The if-else statement is used to execute both the true part and the false part of a given condition. If the condition is true, the if block code is executed and if the condition is false, the else block code is executed

```
if name == 'Alice':
    print('Hi, Alice.')
elif age < 12:
    print('You are not Alice, kiddo.')
else:
    print('You are neither Alice nor a little kid.')
```



2. With Python programming examples to each, explain the syntax and control flow diagrams of **break and continue statements.**

Ans: break statement

There is a shortcut to getting the program execution to break out of a while loop's clause early.

If the execution reaches a break statement, it immediately exits the while loop's clause. In code, a break statement simply contain the break keyword.

```
while True:
      print('Please type your name.')
     name=input()
    if name == 'your name':
       break
print('Thank you!')
```

Continue statement:

Like break statements, continue statements are used inside loops. When the program reaches a continue statement, the program execution immediately jumps back to the start of the loop and reevaluates the loop's condition.

execution

```
while True:
   print('Who are you?')
   if name != 'Joe':
     continue

   name = input()
   print('Hello, Joe. What is the password? (It is a fish.)')
   password = input()
   if password == 'swordfish':
     break
print('Access granted.')
```

3. **Explain TWO ways of importing modules into application in Python with syntax and suitable programming examples.**

   Each module is a Python program that contains a related group of functions that can be embedded in your programs. For example, the **math module** has mathematics related functions, the **random module** has random number–related functions, and so on.

   In code, an import statement consists of the following:

   • The import keyword
   • The name of the module
   • Optionally, more module names, as long as they are separated by commas

```
import random
for i in range(5):
        print(random.randint(1, 10))
```

   When you run this program, the output will look something like this:

   4
   1
   8
   4
   1

The **random.randint()** function call evaluates to a random integer value between the two integers that you pass it. Since **randint()** is in the random module, you must first type random. in front of the function name to tell Python to look for this function inside the random module.
Here's an example of an import statement that imports four different modules:

import random, sys, os, math

4. **How can you prevent a python program from crashing? Discuss different ways to avoid crashing.**

Exception Handling

Right now, getting an error, or exception, in your Python program means the entire program will crash. You don't want this to happen in real-world programs. Instead, you want the program to detect errors, handle them, and then continue to run.
For example, consider the following program, which has a "divide-byzero" error.

```
def spam(divideBy):
return 42 / divideBy

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```

Errors can be handled with try and except statements. The code that could potentially have an error is put in a try clause. The program execution moves to the start of a following except clause if an error happens.
You can put the previous divide-by-zero code in a try clause and have an except clause contain code to handle what happens when this error occurs.

```
def spam(divideBy):
try:
    return 42 / divideBy
except ZeroDivisionError:
    print('Error: Invalid argument.')
print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```

5. **Explain FOUR scope rules of variables in Python.**

Scopes matter for several reasons:
• Code in the global scope cannot use any local variables.
• However, a local scope can access global variables.
• Code in a function's local scope cannot use variables in any other local scope.
• You can use the same name for different variables if they are in different scopes. That is, there can be a local variable named spam and a global variable also named spam.

The reason Python has different scopes instead of just making everything a global variable is so that when variables are modified by the code in a particular call to a function, the function interacts with the rest of the program only through its parameters and the return value.

6. **Explain local and Global scope variable in python with example.**

Parameters and variables that are assigned in a called function are said to exist in that function's **local scope**.
Variables that are assigned outside all functions are said to exist in the **global scope.**
A variable that exists in a local scope is called a **local variable,** while a variable that exists in the global scope is called a **global variable.**
A variable must be one or the other; it cannot be both local and global

Think of a scope as a container for variables. When a scope is destroyed, all the values stored in the scope's variables are forgotten. There is only one global scope, and it is created when your program begins. When your program terminates, the global scope is destroyed, and all its variables are forgotten.

Local scope

```
def myfunc():
    x = 300  ----- Local scope
    print(x)

myfunc()
```

x=300 is the local scope

Global scope

```
x = 300 ------Global scope

def myfunc():
  print(x)

myfunc()

print(x
```

7. **What are functions? Explain python function with parameters and return statements.**

A *function* is like a mini-program within a program.

In Python, functions are defined using def statements, with parameters enclosed in parentheses (), and return values are indicated by the return statement. Note that blocks are expressed with indentation (usually four spaces) rather than brackets.

def Statements with Parameters

When you call the print() or len() function, you pass in values, called arguments in this context, by typing them between the parentheses. You can also define your own functions that accept arguments.

```
def hello(name):
        print('Hello ' + name)
hello('Alice')
hello('Bob')
```

Return Values and return Statements

When you call the len() function and pass it an argument such as 'Hello', the function call evaluates to the integer value 5, which is the length of the string you passed it. In general, the value that a function call evaluates to is called the return value of the function.

A return statement consists of the following:

• The return keyword

• The value or expression that the function should return

8. **Explain string concatenation and replication with examples**.

String Concatenation

When + is used on two string values, it joins the strings as the string concatenation operator.

```
>>> 'Alice' + 'Bob'
 'AliceBob‘
```

String Replication

The * operator is used for multiplication when it operates on two integer or floating-point values. But when the * operator is used on one string value and one integer value, it becomes the string replication operator.

```
 >>> 'Alice' * 5
'AliceAliceAliceAliceAlice'
```