1. **Illustrate the different types of control flow statements available in Python with flowcharts.**

There are 6 different types of flow control statements available in Python:

1. *if-else*
2. *Nested if-else*
3. *for*
4. *while*
5. *break*
6. *Continue*

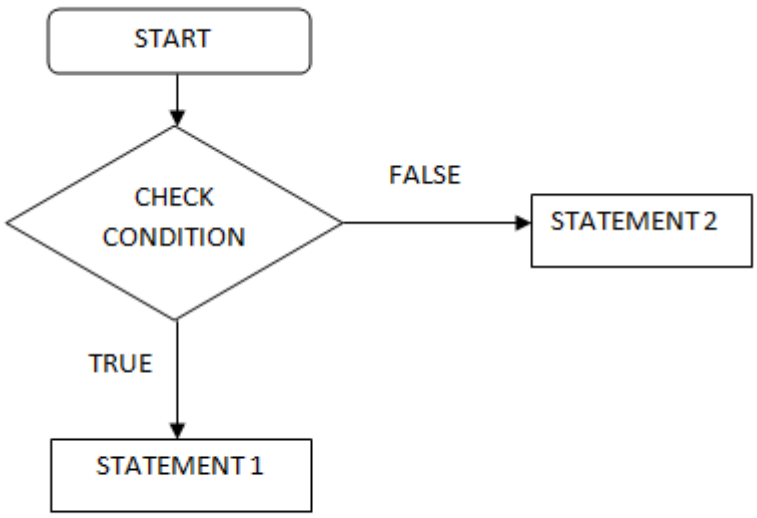
if-else

If-else is used for decision making; when code statements satisfy the condition, then it will return non-zero values as true, or else zero or NONE value as false, by the Python interpreter.

Syntax

```
1. if(<condition>):
2.     Statement 1
3.     ...
4. else:
5.     Statement 2
6.     ...
```

Understand this statement with a flow chart.



Example

Check In[3] and In[4] and also In[5] and In[6].

In [3]:
```
num = 7
if num > 0:
    print(num, "is a Non-zero value.")
print("Thank you.")
```

```
7 is a Non-zero value.
Thank you.
```

In [4]:
```
num = 0
if num > 0:
    print(num, "is a Non-zero value.")
print("Thank you.")
```

```
Thank you.
```

In [5]:
```
num = 7
if num >= 0:
    print("Non-zero or Zero")
else:
    print("Neg. number")
```

```
Non-zero or Zero
```

In [6]:
```
num = -3
if num >= 0:
    print("Non-zero or Zero")
else:
    print("Neg. number")
```

```
Neg. number
```

Nested if-else

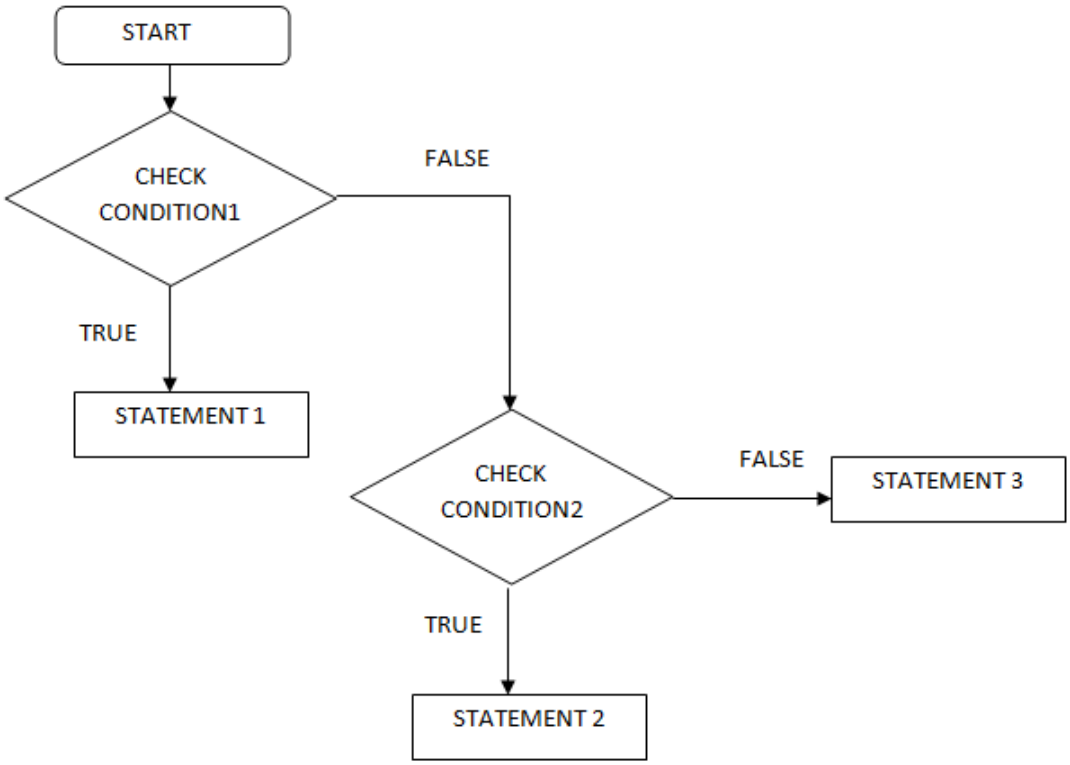
With if...elif...else, elif is a shortened form of else if. It works the same as 'if' statements, where the if block condition is false then it checks to elif blocks. If all blocks are false, then it executes an else statement. There are multiple elif blocks possible for a Nested if...else.

Syntax

```
1. if  (<condition 1>):
2.     Statement 1
3.     ...
4. elif (<condition 2>):
```

```
    Statement 2
    ...
else
    Statement 3
    ...
```

Flow chart of Nested-if else



Remember there is no condition statement associated with else part of these flow control statements. It will execute ig statements only in the case that of all conditions are false.

Example

Check In[3] and In[8] and also In[9] and In[10].

```
In [8]: num=6.7
        if num > 0:
            print("Pos. number")
        elif num == 0:
            print("Zero")
        else:
            print("Neg. number")

        Pos. number
```

```
In [9]: num=-8.9
        if num > 0:
            print("Pos. number")
        elif num == 0:
            print("Zero")
        else:
            print("Neg. number")

        Neg. number
```

```
In [10]: num=0
         if num > 0:
             print("Pos. number")
         elif num == 0:
             print("Zero")
         else:
             print("Neg. number")

         Zero
```

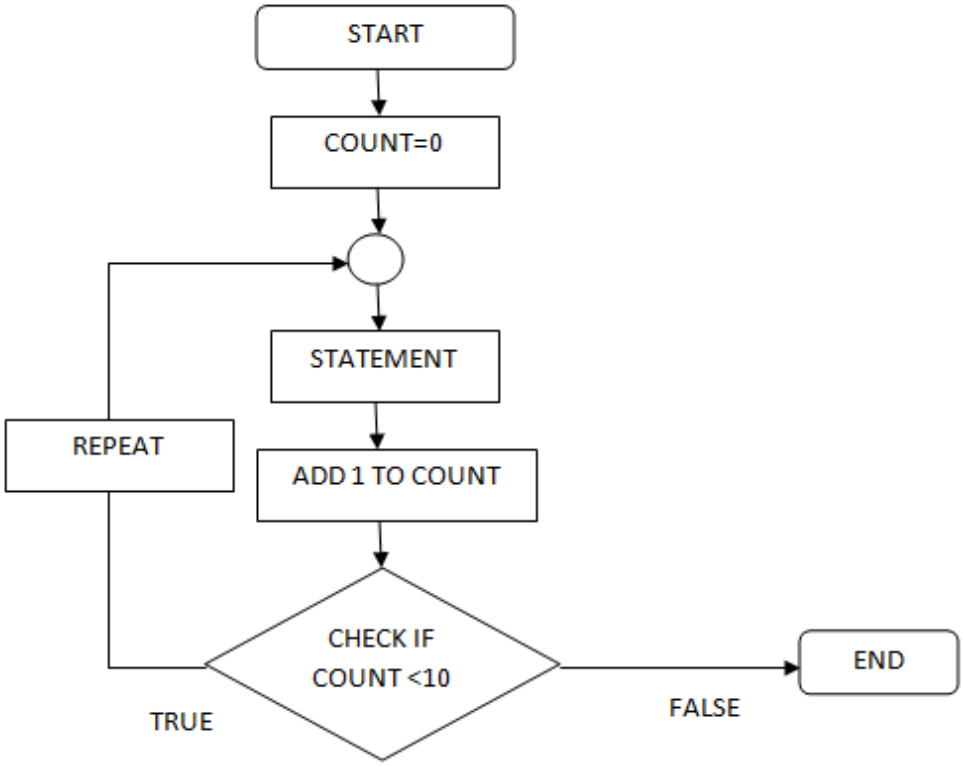
for Statement

The for loop statement has variable iteration in a sequence(list, tuple or string) and executes statements until the loop does not reach the false condition.

Syntax

```
1. for value in sequence:
2.     ...body statement of for
```

Flow chart of for statement



Example

Check In[14] and In[16]. The continue statement is used to stop for loop, in case there is an else block missed.

```
In [14]: numbers = [6, 5, 3, 8, 4, 2, 5, 4]
         sum = 0
         for val in numbers:
             sum = sum+val
         print("The sum is", sum)

         The sum is 37

In [16]: numbers = [0, 1, 3, 6]

         for i in numbers:
             print(i)
         else:
             print("No items left.")

         0
         1
         3
         6
         No items left.
```

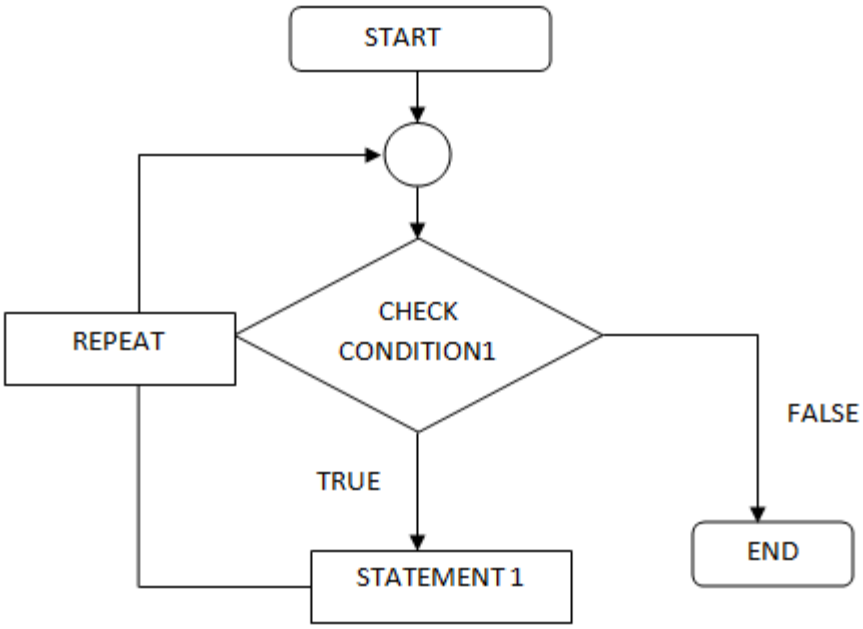
while loop

A while loop is used in python to iterate over the block of expression which matches to true. Non-zero values are True and zero and negative values are False, as interpreted in Python.

Syntax

```
1. while(<condition>):
2.     statement 1..
```

Flow chart of while loop



Example

Check In[4] and In[7]. In[7]. If the user wants to add a number of his choice, then use: n = int(input("Enter number: ")) instead of n=20. Also, check-in In[3] for a while..else loop.

In [4]:
```
i = 2
while i < 8:
  print(i)
  i += 1
```

```
2
3
4
5
6
7
```

In [7]:
```
# Program to add natural numbers upto
# add = 1+2+3+...+n
n = 20
add = 0
i = 1 #counter value

while i <= n:
    add = add + i
    i = i+1    #counter updated

print("The sum is", add)
```

```
The sum is 210
```

Break statement

The Python Break statement is used to break a loop in a certain condition. It terminates the loop. Also, we can use it inside the loop then it will first terminate the innermost loop.

Syntax

I. break

II. with for loop

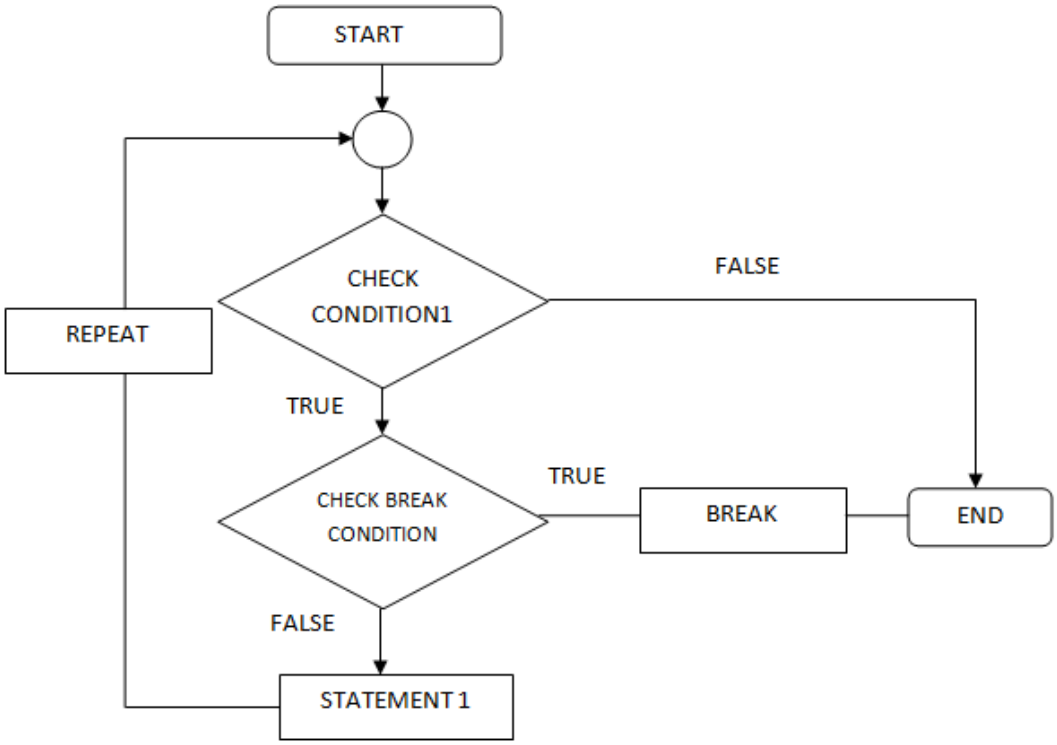
```
1. for value in sequence:
2.     ...body statement of for
3.     if(<condition>):
4.        break
5.     ...body statement of for loop
6.
7. ...body statement outside of for loop
```

III. with a while loop

```
while(<condition>):
    statement 1...
   if(<condition>):
        break
   ...Statement of while loop

....Statements outside while loop
```

Break statement Flow Chart



Example

```
In [13]: a=10
         while a>0:
             a-=1
             if (a!=5):
                 print(a)
             else:
                 break
```

```
9
8
7
6
```

```
In [14]: for a in "Python":
             if a == "h":
                 break
             print(a)

         print("Loop ends")
```

```
P
y
t
Loop ends
```

Continue Statement

A continue statement won't continue the loop, it executes statements until the condition matches True.

Syntax

I. continue

II. with for loop

```
1. for value in sequence:
2.     ...body statement of for
3.     if(<condition>):
4.         continue
5.     ...body statement of for loop
6.
7. ...body statement outside of for loop
```

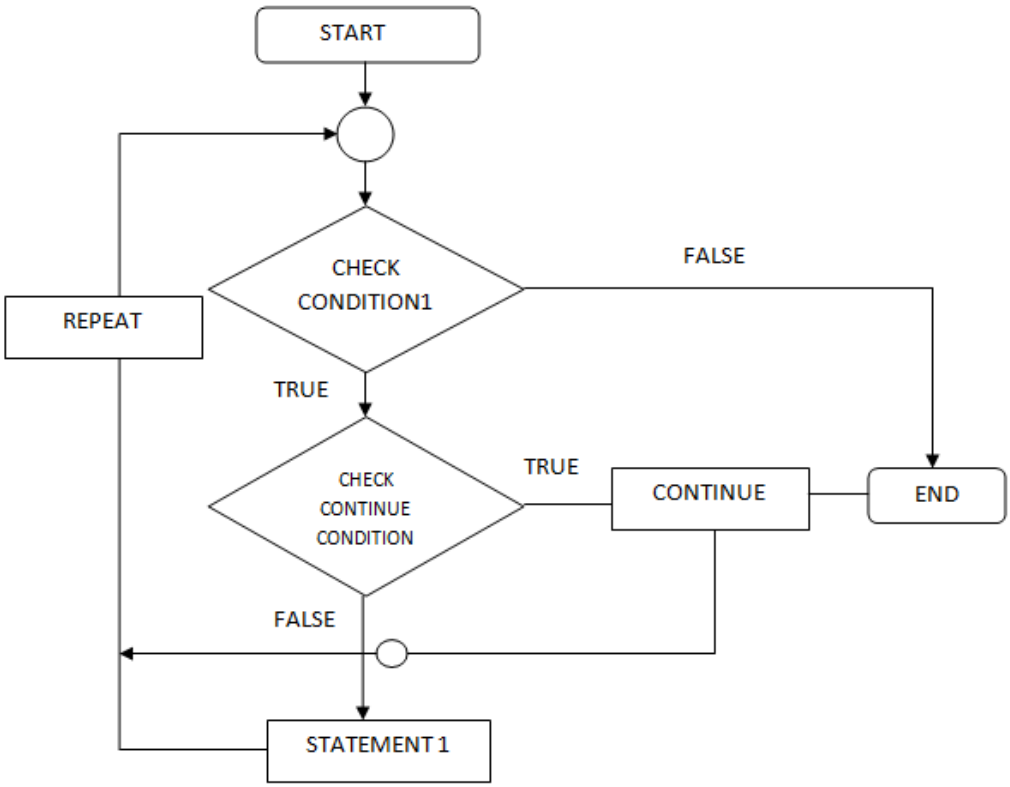
III. with the while loop

```
1. while(<condition>):
2.     statement 1...
```

```
    if(<condition>):
        continue
    ...Statement of while loop

...Statements outside while loop
```

Continue statement Flow Chart



Example

In [17]:
```
a=10
while a>0:
    a-=1
    if (a!=5):
        print(a)
    else:
        continue  #skip 5 and loop is on iteration
```

```
9
8
7
6
4
3
2
1
0
```

In [19]:
```
for a in "Python":
    if a == "h":
        continue
    print(a)

print("Loop ends")
```

```
P
y
t
o
n
Loop ends
```

One more additional Flow statement is PASS.

PASS

In Python, pass, and comment both are quite similar. The pass contains a Null value. The Python interpreter ignores the comment statement, but it's not possible to ignore a pass statement. There is nothing is going to execute when a pass is used. It is used as a Place Holder in a loop or a function.

Example

```
In [21]: v1 = {'p', 'a', 's', 's'}
         for v in v1:
             pass
```

It executes nothing.

2. Write a Program to Prompt for a Score between 0.0 and 1.0. If the Score is out of range, print an error. If the score is between 0.0 and 1.0, print a grade using the following table

```
score =
input("Enter
Score: ")
                s =  float(score)
                x = 'Error'
                if s >= 0.9:
                        x = 'A'
                elif s >=0.8:
                        x='B'
                elif s >=0.7:
                        x='C'
                elif s >= 0.6:
                        x='D'
                elif s < .6:
                        x ='F'
                else:
                        x ="Out of Range"
                print (x)
```

3. Write a program to display the fibonacci sequences up to nth term where n is provided by the user.

```
1. # take input from the user
2. nterms = int(input("How many terms? "))
3.
4. # first two terms
5. n1 = 0
6. n2 = 1
7. count = 2
8.
9. # check if the number of terms is valid
10.if nterms <= 0:
11.   print("Plese enter a positive integer")
12.elif nterms == 1:
13.   print("Fibonacci sequence:")
14.   print(n1)
15.else:
```

```
print("Fibonacci sequence:")
print(n1,",",n2,end=', ')
while count < nterms:
    nth = n1 + n2
    print(nth,end=' , ')
    # update values
    n1 = n2
    n2 = nth
    count += 1
```

3. **Write a program to repeatedly check for the largest number until the user enters "done"**

```
largest =
None
                smallest = None

                while True:
                    try:
                        num = raw_input("Enter a number: ")
                        if num == 'done':
                            break;
                        n = int(num)
                        largest = num if largest < num or largest == None else
largest
                        smallest = num if smallest > num or smallest == None else
smallest
                    except:
                        print "Invalid input"

                print "Maximum number is ", largest
                print "Minimum number is ", smallest
```

o/p:

#Enter 7,2,bob,10,4 u will get desired output