Decision Making Statements (if,if-else,if-elif-else)

Statements	Syntax	Example	Definition

if	if condition:
	    statement1
	statement2
		i = 10
		if (i > 15):
		   print ("10 is less than 15")
		print ("I am Not in if")
		
		Output:
		`I am Not in if`
			if statement is the most simple decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not

If - else	if (condition):
	    statement1
	else:
	    statement2
		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:
		i is greater than 15
		i'm in else Block
		i'm not in if and not in else Block
			We can use the else statement with if statement to execute a block of code when the condition is false.

nested-if	if (condition1):
	    statement
	      if (condition2):
	        statement
	      # if Block is end here
	# if Block is end here
		i = 10
		if (i == 10):
		    if (i < 15):
		      print ("i is smaller than 15")
		     if (i < 12):
		      print ("i is smaller than 12 too")
		     else:
		      print ("i is greater than 15")
		
		Output:
		i is smaller than 15
		i is smaller than 12 too
			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.

if-elif-else	if (condition):
	        statement
	elif (condition):
	        statement
	        .
	        .
	else:
	        statement
		i = 20
		if (i == 10):
		   print ("i is 10")
		elif (i == 15):
		   print ("i is 15")
		elif (i == 20):
		   print ("i is 20")
		else:
		   print ("i is not present")
		
		Output:
		`i is 20`
			Here, a user can decide among multiple options. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed.

1. Predict the output:

```
num = 22
if num % 2 == 0:
    print("Even Number")
else:
    print("Odd Number")
```

1. Output:

```
Even Number
```

2. Predict the output:

```
i = 10
if i < 15: print("i is less than 15")
```

2. Output:

```
i is less than 15
```

3. Predict the output:

```
i = 10
print(True) if i < 15 else print(False)
```

3. Output:

```
True
```

4. Predict the output:

```
i = 20;
if (i < 14):
    print ("i is smaller than 14")
    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")
```

4. Output:

```
i is greater than 15
i'm in else Block
i'm not in if and not in else Block
```

5. Predict the output:

```python
num = 1122
if 9 < num < 99:
    print("Two digit number")
elif 99 < num < 999:
    print("Three digit number")
elif 999 < num < 9999:
    print("Four digit number")
else:
    print("number is <= 9 or >= 9999")
```

5. Output:

Four digit number

6. Predict the output:

```python
num = -99
if num > 0:
    print("Positive Number")
else:
    print("Negative Number")
    #nested if
    if -99<=num:
        print("Two digit Negative Number")
```

6. Output:

```
Negative Number
Two digit Negative Number
```

Loop Statements (for,while)

Statements	Syntax	Example	Meaning

while	while (Condition):
	        statement(s)
		count = 0
		while (count < 3):
		        count = count+1
		        print("Hello Geek")
		Output:
		Hello Geek
		Hello Geek
		Hello Geek
			while loop is used for iterators

for	for iterator_var in sequence:
	        statements(s)
		l = ["bennett", "for", "bennetians"]
		for i in l:
		    print(i)
		Output:
		bennett
		for
		bennetians
			for can be used to iterate over iterators and a range.

nested-for	for iterator_var in sequence:
	    for iterator_var in sequence:
	        statements(s)
	        statements(s)
		for i in range(1, 5):
		    for j in range(i):
		        print(i, end=' ')
		    print( )
		Output:
		        1
			Python programming language allows to use for loop inside another for loop.

		2 2 3 3 3 4 4 4 4	

nested-while	while expression:
	    while expression:
	        statement(s)
	statement(s)
		`i = 1`
		`j = 5`
		`while i < 4:`
		`    while j < 8:`
		`        print(i, ",", j)`
		`        j = j + 1`
		`        i = i + 1`
		Output:
		`1 , 5`
		`2 , 6`
		`3 , 7`
			Python programming language allows to use while loop inside another while loop.

1. Predict the output:

```
for char in "Python":
    if (char == "y"):
        print("Current character:", char)
```

1. Output:

```
Current character: y
```

2. Predict the output:

```
while True:
   print("hello")
```

2. Output:

Infinite hello

3. Predict the output:

```
num = 1
while num<5:
 print(num)
```

3. Output:

print ‘1’ indefinitely

4. Predict the output:

```
num = 10
while num > 6:
   print(num)
   num = num-1
else:
   print("loop is finished")
```

4. Output:

```
10
9
8
7
loop is finished
```

5. Predict the output:

```
numbers = [1, 2, 4, 6, 11, 20]
sq = 0
for val in numbers:
       sq = val * val
print(sq)
```

5. Output:

```
400
```

6. Predict the output:

```
sum = 0
for val in range(1, 6):
        sum = sum + val
print(sum)
```

6. Output:

```
15
```

7. Print the following pattern

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

7. Output:

```python
print("Second Number Pattern ")
lastNumber = 6
for row in range(1, lastNumber):
    for column in range(1, row + 1):
        print(column, end=' ')
    print("")
```

8. Print First 10 natural numbers using while loop

Expected output:

```
0
1
2
3
4
5
6
7
8
9
10
```

8. Solution

```python
i = 0
while i <= 10:
    print(i)
    i += 1
```

9. Reverse the following list using for loop

list1 = [10, 20, 30, 40, 50]

Expected output:
50
40
30
20
10

9. Solution

```
list1 = [10, 20, 30, 40, 50]
start = len(list1) - 1
stop = -1
step = -1
for i in range(start, stop, step) :
    print(list1[i])
```

10. Python program to display all the prime numbers within a range
Note: A Prime Number is a whole number that cannot be made by multiplying other whole numbers

Examples:

6 is not a Prime Number because it can be made by 2×3 = 6
37 is a Prime Number because no other whole numbers multiply together to make it.

Given:
start = 25
end = 50

Expected output:

```
Prime numbers between 25 and 50 are:
29
31
37
41
43
47
```

10. Solutions

```
start = 25
end = 50
print("Prime numbers between", start, "and", end, "are:")

for num in range(start, end + 1):
    # all prime numbers are greater than 1
    # if number is less than or equal to 1, it is not prime
    if num > 1:
        for i in range(2, num):
            # check for factors
            if (num % i) == 0:
                # not a prime number so break inner loop and
                # look for next number
                break
        else:
            print(num)
```

Control Statements (Continue,Break,Pass)

Statements	Example	Meaning

Continue	for char in 'Pythn':
	    if (char == 'y'):
	        continue
	    print("Current character: ", char)
	Output:
	Current character: P
	Current character: t
	Current character: h
	Current character: n
		When the program encounters continue statement, it will skip the statements which are present after the continue statement inside the loop and proceed with the next iterations.

break	for char in 'Python':
	    if (char == 'h'):
		The break statement is used to terminate the loop

	break print("Current character: ", char) **Output:** Current character: P Current character: y Current character: t	containing it, the control of the program will come out of that loop.

pass	for char in 'Python':
	if (char == 'h'):
	pass
	print("Current character: ", char)
	Output:
	Current character: P
	Current character: y
	Current character: t
	Current character: h
	Current character: o
	Current character: n
		Pass statement is python is a null operation, which is used when the statement is required syntactically.

1. Predict the output:

```
for num in [11, 9, 88, 10, 90, 3, 19]:
   print(num)
   if(num==88):
           print("The number 88 is found")
           print("Terminating the loop")
           break
```

1. Output:

```
11
9
88
The number 88 is found
Terminating the loop
```

2. Predict the output:

```
for num in [20, 11, 9, 66, 4, 89, 44]:
        if num%2 == 0:
        continue
        print(num)
```

2. Output:

```
11
9
89
```

3. Predict the output:

```
for num in [20, 11, 9, 66, 4, 89, 44]:
    if num%2 == 0:
        pass
    else:
        print(num)
```

3. Output:

```
11
9
89
```

4. Given a list iterate it and display numbers which are divisible by 5 and if you find number greater than 150 stop the loop iteration
list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]

Expected output:
15
55
75
150

4. Solution:

```
list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]
for item in list1:
    if (item > 150):
        break
    if(item % 5 == 0):
        print(item)
```

5. Display Fibonacci series up to 10 terms

Expected output:

```
Fibonacci sequence:
0  1  1  2  3  5  8  13  21  34
```

5. Solutions:

```
terms = 10
# first two terms
num1, num2 = 0, 1
count = 0

print("Fibonacci sequence:")
while count < terms:
    print(num1, end="   ")
    temp = num1 + num2
    # update values
    num1 = num2
    num2 = temp
    count += 1
```