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.

Comparison Operators (==,<,>,<=,>=)

Operator	Example	Meaning	Result
==	a == b	Equal to	True if the value of a is equal to the value of b False otherwise
!=	a != b	Not equal to	True if a is not equal to b False otherwise
<	a < b	Less than	True if a is less than b False otherwise
<=	a <= b	Less than or equal to	True if a is less than or equal to b False otherwise
>	a > b	Greater than	True if a is greater than b False otherwise
>=	a >= b	Greater than or equal to	True if a is greater than or equal to b False otherwise

Logical Operator

Operator	Example	Meaning
not	not x	True if x is False False if x is True (Logically reverses the sense of x)
or	x or y	True if either x or y is True False otherwise

and	x and y	True if both x and y are True False otherwise
not in	x not in y	x not in y, here not in results in a 1 if x is not a member of sequence y
in	x in y	x in y, here in results in a 1 if x is a member of sequence y

Bit-wise Operator

Operator	Example	Meaning	Result
<<	x << y	**bits shifted to the left**	Returns x with the bits shifted to the left by y places
>>	x >> y	**bits shifted to the right**	Returns x with the bits shifted to the right by y places
&	x & y	**bitwise and**	Each bit of the output is 1 if the corresponding bit of x AND of y is 1, otherwise it's 0
|	x | y	**bitwise or**	Each bit of the output is 0 if the corresponding bit of x AND of y is 0, otherwise it's 1
~	~ x	**complement of x**	Returns the complement of x - the number you get by switching each 1 for a 0 and each 0 for a 1
x ^ y	x ^ y	**Bitwise XOR operator**	Each bit of the output is the same as the corresponding bit in x if that bit in y is 0, and it's the complement of the bit in x if that bit in y is 1.

Others:

Data Type	Meaning
Booleans	**Boolean in Python can have two values – True or False**
Numbers	The numbers in Python are classified using the following keywords: **int, float, and complex**.
Strings	A sequence of one or more characters enclosed within either single quotes ‘ or double quotes ” is considered as String in Python. Any letter, a number or a symbol could be a part of the sting.
Lists	Lists in Python can be declared by placing elements inside **square brackets separated by commas**.
Tuples	A tuple is a heterogeneous collection of Python objects, using enclosing parentheses ( ) having its elements separated by commas inside.
Sets	A set is an unordered collection of unique and immutable objects. Its definition starts with enclosing braces { } having its items separated by commas inside.
Dictionaries	**Python syntax for creating dictionaries use braces { } where each item appears as a pair of keys and values.**

1. Predict the output:

```
a, b = 1, 2
if a==1:
         if b==2:
               print("a is", a, "and b is", b)
```

2. For the given below program:

```
print(“Enter the Number: “)
num = int(input())
R = num%2
if num%2==0:
    print(“Bennett”)
    print(R)
else:
    print(“University”)
    print(R)
```

if the input is -13, then what will be the output

3. Predict the output:

```
#Let's Play a game!!

First_Person = int(input("Enter a number between 1 and 10: "))
Second_Person = int(input("Enter a number between 1 and 10: "))
if (First_Person >= 1) and (First_Person <= 10):
    if (Second_Person >= 1) and (Second_Person <= 10):
        print("Your secret number is: ", First_Person * Second_Person)
    else:
        print("Second person entered value is out of range!")
else:
    print("First person entered value is out of range!")
```

if `First_Person entered` input is -1 and `Second_Person entered` input is 11, then what will be the output

4. Write an algorithm that perform simple grading scheme according to the given below table. Now, accept only valid inputs for our grade conversion script and displays an error message otherwise. That is, if input is greater than 100 or less than 0, then display error message.

LETTER GRADE S	RANGE OF NUMERIC GRADE
A+	All grades above 95
A	All grades above 90 and below 96
B	All grades above 80 and below 91
C	All grades above 70 and below 81
D	All grades above 60 and below 71
E	All grades above 50 and below 61
F	All grades below 51

5. Enter the number, check whether the given input is divisible by 2 and 4 both.

6. Enter three values as a=10, b=20 and c=30 and calculate the average of these three numbers. Then, check the following conditions:

I. If the average value is greater than the individual value
II. If the average value is greater than a and b
III. If the average value is greater than a and c
IV. If the average value is greater than b and c

7. Find the output of the following:

a)

```
a=20
a*=5!=5 or 6>=10>>2
print(a)
```

b)

```
a=10
a/=5!=5 and 6>=100>>2
print(a)
```

c)

```
a=50<=55 and 6>=10 and 2
print(a)
```

d)

```
a=20
a = 50 or 55 and 6<=10 and 20%2
print(a)
```

e)

```
a=20
a = int(5^5 or 55 and 6>=10 and 2*2)
print(a)
```

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

a)

```
Thislist = ["Apple", "Banna", "Cherry"]
mylist = Thislist.copy()
print(Thislist)
print(mylist)
```

b)

```
Thislist = ['B', 'E', 'N', 'N', 'E','T','T']
print("Initial List: ", Thislist)
Sliced_Thislist = Thislist[:5]
print(Sliced_Thislist)
```

c)

```
Thislist = ['B', 'E', 'N', 'N', 'E','T','T']
print("Initial List: ", Thislist)
Sliced_Thislist = Thislist[:-5]
print(Sliced_Thislist)
```

d)

```
lst=[[1,2,3],'hello',[3,4,5,6]]
print(lst[1][1])
print(lst[2][1])
print(lst[1])
print(lst[0])
lst[2][:2]=[1,1]
print(lst)
print(lst[0][1])
```

9. What will be output of the following statements?

```
a = 10 b = 4
print(a & b)
print(a | b)
print(~a)
print(a ^ b)
print(a >> 2)
print(a << 2)
```

Explain all the output.