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

| Statements | Syntax | Example | Definition |
|---|---|---|---|
| if | if condition:<br>    statement1<br>statement2 | i = 10<br>if (i > 15):<br>   print ("10 is less than 15")<br>print ("I am Not in if")<br><br>**Output:**<br>`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):<br>    statement1<br>else:<br>    statement2 | i = 20;<br>if (i < 15):<br>   print ("i is smaller than 15")<br>   print ("i'm in if Block")<br>else:<br>   print ("i is greater than 15")<br>   print ("i'm in else Block")<br>print ("i'm not in if and not in else Block")<br><br>**Output:**<br>i is greater than 15<br><br>i'm in else Block<br><br>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):<br>   statement<br>    if (condition2):<br>     statement<br>    # if Block is end<br>     here<br>  # if Block is end<br>here | i = 10<br>if (i == 10):<br>    if (i < 15):<br>    print ("i is smaller than 15")<br>    if (i < 12):<br>    print ("i is smaller than 12 too")<br>    else:<br>    print ("i is greater than 15")<br><br>**Output:**<br>i is smaller than 15<br><br>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):<br>    statement<br>elif (condition):<br>    statement<br>     .<br>     .<br>else:<br>    statement | i = 20<br>if (i == 10):<br>   print ("i is 10")<br>elif (i == 15):<br>   print ("i is 15")<br>elif (i == 20):<br>   print ("i is 20")<br>else:<br>   print ("i is not present")<br><br>**Output:**<br>`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<br>False otherwise |
| != | a != b | Not equal to | True if a is not equal to b<br>False otherwise |
| < | a < b | Less than | True if a is less than b<br>False otherwise |
| <= | a <= b | Less than or equal to | True if a is less than or equal to b<br>False otherwise |
| > | a > b | Greater than | True if a is greater than b<br>False otherwise |
| >= | a >= b | Greater than or equal to | True if a is greater than or equal to b<br>False otherwise |

## Logical Operator

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

| and | x and y | True if both x and y are True<br>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 = 12

if a > 6:
    print ("Value in a is", a)

print("End of program.")
```

2. Predict the output:

```
if 2==1:
    print("True")
else:
    print("False")
```

3. Complete the fill in the blank "---" in the "if" statement to make a complete program to verify whether "y" is leap year or not:

```
print("Enter the Year: ")
y = int(input())

if y%4==0 and ---:
    print("\nIt is a leap Year")
elif y%400==0:
    print("\nIt is a leap Year")
else:
    print("\nIt is not a leap Year")
```

4. Write an algorithm that perform simple grading scheme according to the given below table.

| LETTER GRADE S | RANGE OF NUMERIC GRADE |
|---|---|
| A | All grades above 89 |
| B | All grades above 79 and below 90 |
| C | All grades above 69 and below 80 |

| D | All grades above 59 and below 70 |
|---|---|
| F | All grades below 60 |

5.  You have your grades (any number between 0 to 100). Instructor will increase your grades by 4 if your existing grades is even and it is divisible by 4 too. Instructor will increase your grades by 3 if your existing grades is odd and "grades -1" is divisible by 4. Otherwise, Instructor will not change you grades. Write a program to perform this operation

> *Enter your grades marks: 68*
> *New grades = 72 (because 68 is even and it is divisible by 4 too)*
>
> *Enter your grades marks: 67*
> *New grades = 67 (because 67 is odd and "67 - 1" is not divisible by 4)*
>
> *Enter your grades marks: 69*
> *New grades = 72 (because 69 is odd and "69 - 1" is divisible by 4)*

6. Enter the radius of a circle. If the radius is greater than 0, calculate and print the area and circumference of the circle. If user enter A, P and D, outputs are area, parameters and diameter, respectively

7. Find the output of the following:
   a)
```
a=20
a*=5==5 and 6>=23>>3
print(a)
```
   b)
```
a=20
a*=5!=5 and 6>=100>>2
print(a)
```
   c)
```
a=20
a*=5!=5 or 6>=10>>2
print(a)
```
   d)
```
a=10
a/=5!=5 and 6>=100>>2
```

```
    print(a)

e)
    a=50<=55 and 6>=10 and 2
    print(a)
f)
    a=20
    a = 50 or 55 and 6<=10 and 20%2
    print(a)

g)
    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"]
    Thislist[1] = "blackcurrant"
    print(Thislist)
```

b)
```
    Thislist = ["Apple", "Banna", "Cherry"]
    Thislist.insert(2, "orange")
    print(Thislist)
```

c)
```
    Thislist = ["Apple", "Banna", "Cherry"]
    mylist = Thislist.copy()
    print(Thislist)
    print(mylist)
```
d)
```
    Thislist = ['B', 'E', 'N', 'N', 'E','T','T']
    print("Initial List: ", Thislist)
    Sliced_Thislist = Thislist[:5]
    print(Sliced_Thislist)
```

e)

```
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.