# Assignment

## Conceptual Questions:

Q1.What is the purpose of using control flow statements like if, else, and elif in Python?

Ans. The purpose of using control flow statements is to control the execution and execute a specific block if the condition is true and ignore the other conditions.

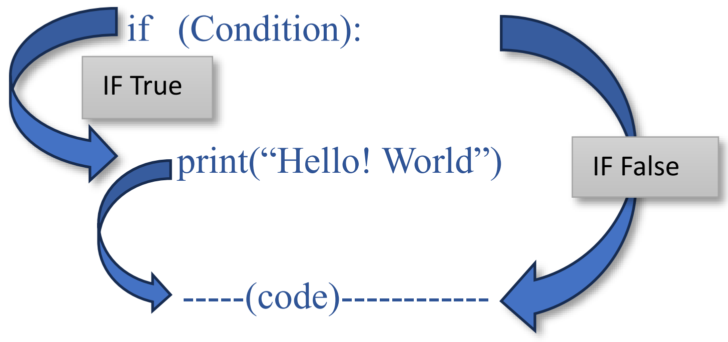
Q2. How does Python determine which block of code to execute in an if-else statement

Ans. First Python check the if condition. If it is true python interpreter go in if block and execute the code. If condition is false python interpreter directly execute the code written in else block.

Q3. Explain the difference between the if-elif-else and nested if-else structures

Ans.

1.IF:



IF:

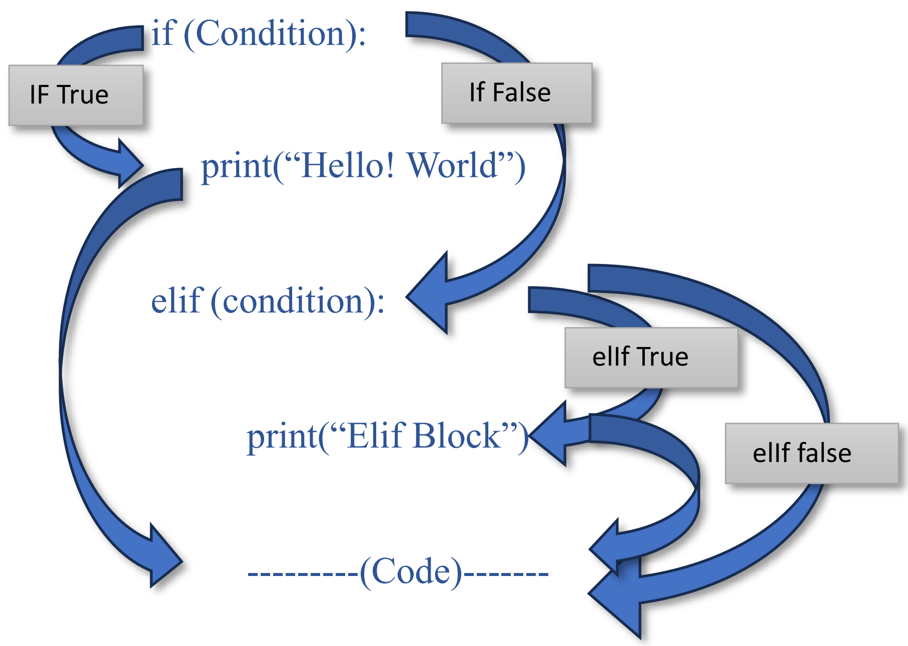
TRUE

If the condition is true, the interpreter enters the 'if' block and executes the code (prints 'Hello, World!') and then go to next line of code. This line of code is not dependent on the 'if' condition. It executes whether the 'if' condition is true or false.

FALSE

If the 'if' condition is false, the interpreter proceeds to the next line of code and execute it. This line of code is not dependent on the 'if' condition. It executes whether the 'if' condition is

2. ELIF:

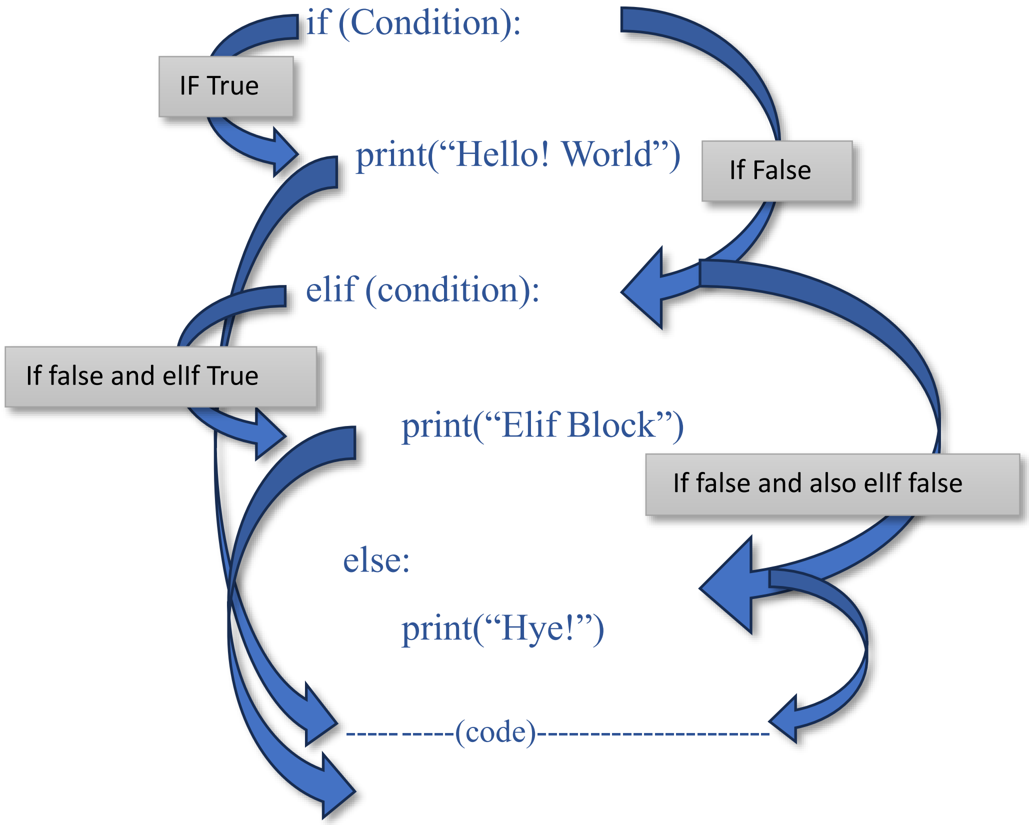


```
if (Condition):
    print("Hello! World")
elif (condition):
    print("Elif Block")
---------(Code)-------
```

ELIF:

the 'if' condition is true, the interpreter enters the 'if' block and executes the code (prints 'Hello, World'), skipping the 'elif' block. If the 'if' condition is false, the interpreter checks the 'elif' condition. If the 'elif' condition is true, the interpreter executes the code within the 'elif' block and then proceeds to the next line. If the 'elif' condition is false, the interpreter skips the 'elif' block and moves on to the next line of code."

3. ELSE:

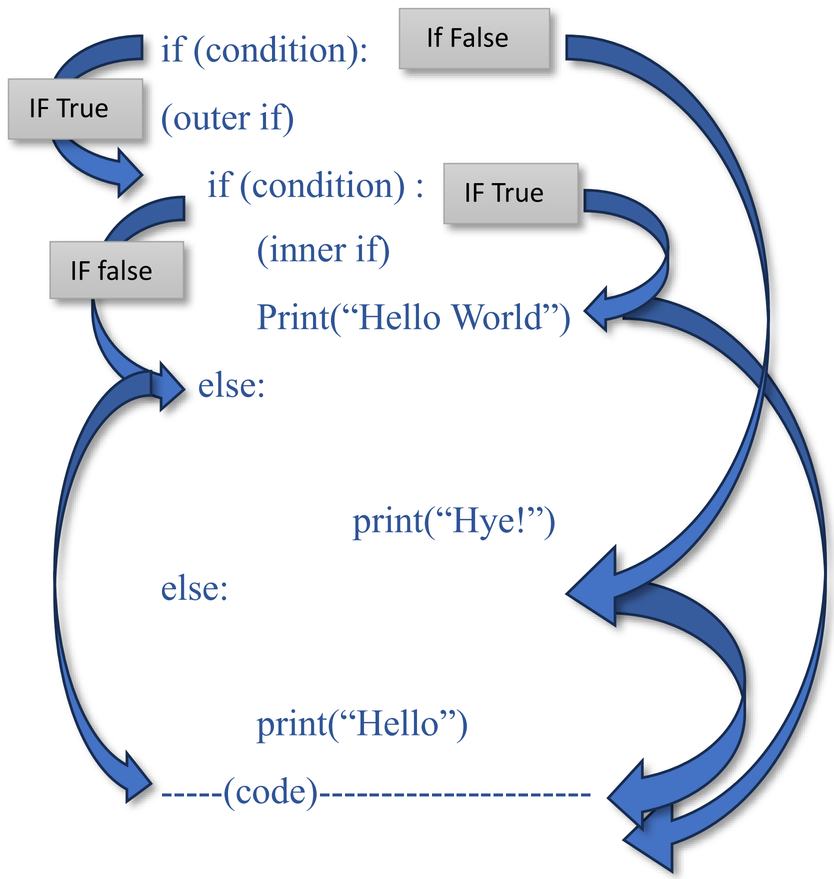


```
if (Condition):
    print("Hello! World")
elif (condition):
    print("Elif Block")
 else:
    print("Hye!")
----------(code)--------------------
```

ELSE:

If the 'if' condition is true, the interpreter enters the 'if' block and executes the code (prints 'Hello, World'), skipping the 'elif' block. If the 'if' condition is false, the interpreter checks the 'elif' condition. If the 'elif' condition is true, the interpreter executes the code within the 'elif' block and then proceeds to the next line. If the 'elif' condition is false, the interpreter skips the 'elif' block and moves to the else and execute it'. "else" block only executes if "if" and "elif" both condition is false if any one of them is true then else block will not execute.

4.NESTED IF-ELSE:



```
if (condition):
(outer if)
    if (condition) :
        (inner if)
        Print(“Hello World”)
    else:

            print(“Hye!”)
else:

        print(“Hello”)
-----(code)---------------------
```

NESTED IF ELSE:

In nested if else we add multiple condition in one if. If condition in outer” if “ is true then interpreter check inner “if “condition if outer “if” condition false then interpreter not check the inner if condition and then execute the else block

Q4. How can you use logical operators (and, or, not) with if-else statements in Python?

Ans.

1. and:

   You can use 'and' to check if all conditions in the 'if' statement are correct. If all conditions specified in the 'if' statement are satisfied, the interpreter will execute the code within the 'if' block.

   Example,

```
        if 2<3 and 3<5:
                Print(“2<3<5”)
        else:
                print(“Not valid”)
```

2. or:

   You can use “or” to check if any one condition in the ‘if’ statement is correct. If any one condition specified in the ‘if’ statement is satisfied, the interpreter will execute the code within the ‘if’ block.

   Example,
   ```
   if 2<3 or 3<5:
           print(“2<3<5”)
   else:
           print(“Not valid”)
   ```
3. not:

   In “not” if any conditional expression returns true then “not” reverse it and vice versa. So, you can use “not” to reverse the true into false

   Example,
   ```
   if not(3>7 or 5>7):    #or return false and not change it in true So if block execute
           print(“3 and 5 are smallest number”)
   else:
           print(“3 and 5 are largest number”)
   ```

Q5. Describe scenarios where nested if-else statements are preferred over if-elif-else structures.

Ans. Nested if-else preferred when you want to check multiple conditions are satisfied by one expression, value etc.

Scenario 1: Prompt the user to enter a year. Check whether the number is positive or not. If the number is positive, then check if the given year is a leap year or not.

checking whether a given year is a leap year, first verify that the given number is greater than zero. If the number is positive, proceed to the next condition to determine if the given year is a leap year or not. If the number is negative, the leap year condition cannot be executed. If the number is

positive but the year is not a leap year, the 'else' condition within the outer 'if' block will be executed

Code:

```
user=int(input("Enter Year: "))
if user>0:
    if (user%4==0 and user%100!=0) or (user%400==0):
        print(f"{user} is a Leap year.")
    else:
        print(f"{user} is not a Leap year.")
else:
    print("Cheack the value")
```

Scenario 2: You really like biryani. Here's what happens:

1. No biryani, no eating for you.
2. If there's biryani but no cold drink, and there's salad, you eat.
3. But if there's a cold drink, you prefer to eat there.

In this scenario, first, you check if there is biryani or not. If there is no biryani, you simply print 'I will not eat.' But if there is biryani, then you check if there is a cold drink or not. If there is a cold drink, you simply print 'I will eat food.' However, if there is no cold drink, then you check if there is salad or not. If there is no salad, you don’t eat. But if there is salad, you eat food.

Code:

```
biryani=input("Is there biryani or not? (y or n): ")
if biryani=="y":
    drink=input("Is there a cold drink or not? (y or n): ")
    if drink=="y":
        print("I will eat food :)")
    else:
        salad=input("Is there salad or not? (y or n): ")
        if salad=="y":
            print("I will eat food :)")
        else:
            print("I will not eat food :(")
else:
    print("I will not eat food :(")
```

Q6. How does Python handle multiple conditions in an if-elif-else ladder?

Ans.

Step 1:

If the "if" condition is true, python interpreter execute the block associated with it and skip the “elif” and “else” blocks

Step 2:

If the "if" condition is false, python interpreter check the "elif" conditions one by one. If a true "elif" condition is found, execute the block associated with it and skip the "else" block.

Step 3:

If none of the conditions (if and elif) are true,  python interpreter execute the block associated with the "else" statement

Q7. Why is it important to indent properly when using control flow statements in Python?

Ans. It is important to indent properly when using control flow statements in Python because of python syntax.