

CONDITIONAL STATEMENTS IN PYTHON

IF-ELIF-ELSE STATEMENT

FIELDS FOR EXAMPLES OF IF-ELIF-ELSE STATEMENT.

```
1 conditionOne = True
2 conditionTwo = False
```

'if' Statement:

- Description: The 'if' statement in Python is a fundamental control flow statement that allows you to execute a block of code based on a specified condition. It helps in making decisions in your program by evaluating whether a given condition is True or False.
- Syntax: if conditionOne:
 code to be executed if the conditionOne is True
- Example:

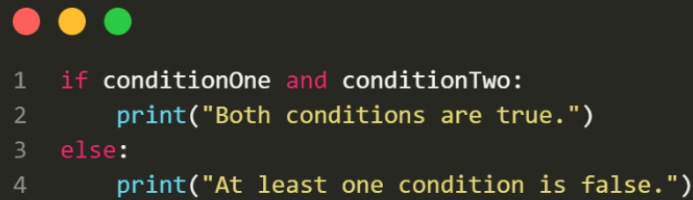
```
1 if conditionOne and conditionTwo:
2     print("Both conditions are true.")
```

- Explanation:
 - The 'if' keyword is followed by the condition that needs to be evaluated.
 - The indented block of code underneath is executed only if the conditions is 'True'.

'else' Statement:

- Description: The 'else' statement in Python is used in conjunction with conditional statements (such as 'if' and 'elif') to define a block of code that executes when the specified condition evaluates to False. If the preceding 'if' or 'elif' condition is not met, the code in the 'else' block is executed.
- Syntax: if conditionOne:
 code to be executed if the conditionOne is True
else
 code to be executed if the conditionOne is False
- Example:

CONDITIONAL STATEMENTS IN PYTHON



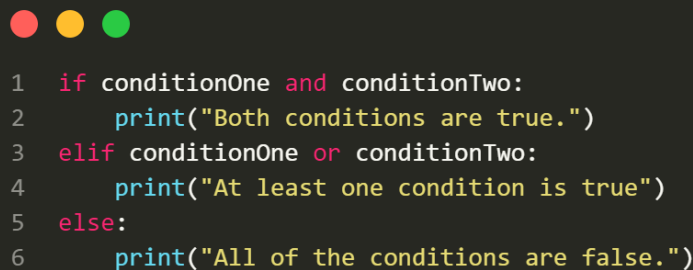
```
1 if conditionOne and conditionTwo:
2     print("Both conditions are true.")
3 else:
4     print("At least one condition is false.")
```

- Explanation:
 - Optionally, the 'else' keyword introduces another indented block of code that is executed when the condition in the 'if' statement is False.

'elif' (else if) Statement:

- Description: The 'elif' statement in Python is utilized within conditional structures to introduce an additional condition to be evaluated if the preceding 'if' statement is False. It allows for the creation of multiple branches of code, and the block beneath the first 'elif' whose condition evaluates to True will be executed. If none of the specified conditions are met, the code in the final 'else' block, if present, will be executed.
- Syntax:

```
if conditionOne:
    code to be executed if conditionOne is True
elif conditionTwo:
    code to be executed if conditionTwo is True
else:
    code to be executed if none of the conditions are True
```
- Example:



```
1 if conditionOne and conditionTwo:
2     print("Both conditions are true.")
3 elif conditionOne or conditionTwo:
4     print("At least one condition is true")
5 else:
6     print("All of the conditions are false.")
```

- Explanation:
 - The 'elif' keyword is used to introduce an additional condition that is evaluated only if the preceding 'if' condition is False.
 - The indented block of code underneath the 'elif' statement is executed only if the 'elif' condition is True.

Nested Conditional Statement:

- Description: Nested Conditional Statements in Python involves embedding one or more conditional statements within the indented block of another, allowing for multi-level

CONDITIONAL STATEMENTS IN PYTHON

decision-making. The indentation level signifies the nesting, creating a hierarchy of logical branches based on the evaluation of conditions within conditions.

- Syntax: if conditionOne:
 code to be executed for conditionOne being True
 if conditionTwo:
 code to be executed for conditionTwo being True
 else:
 code to be executed for conditionTwo being False
else:
 code to be executed for conditionOne being False
- Example:

```
1  if conditionOne:
2      if conditionTwo:
3          print("Both conditions are true.")
4      else:
5          print("Only first condition is true.")
6  else:
7      print("Terminated.")
```

- Explanation:
 - The 'if' keyword is followed by the outer condition that needs to be evaluated.
 - The indented block of code underneath the 'if' statement is executed only if the outer condition is True.
 - Within this block, another 'if' statement can be nested, introducing an inner condition.
 - The indented code underneath the nested 'if' statement is executed only if both the outer and inner conditions are True.
 - This nested structure allows for the creation of more complex decision-making logic based on multiple conditions.

Ternary Operator:

- Description: The Ternary Operator in Python, also known as the conditional expression, provides a concise way to express a conditional statement in a single line. It's often used to assign a value based on a condition.
- Syntax: value_if_true if conditionOne else value_if_false
- Example:

```
1  print("The condition is " + "true" if conditionOne else "false")
```

- Explanation:

CONDITIONAL STATEMENTS IN PYTHON

- `conditionOne` : The expression to evaluate.
- `"true"` : The value to return if the condition is true.
- `"false"` : The value to return if the condition is false.

