Python Fundamentals

Conditional Statements

Conditional statements in Python are used to control the flow of a program based on specific conditions or criteria. They allow you to execute different blocks of code depending on whether a given condition is true or false. Python provides several constructs for handling conditional statements, including if, elif (short for "else if"), and else. Here's an explanation of these constructs:

if Statement:

The if statement is the most fundamental conditional statement in Python. It allows you to specify a condition, and if that condition is true, the code block inside the if statement is executed. If the condition is false, the code block is skipped.

Syntax

if condition:

```
Code to execute if the condition is true
it can one line or multiple lines
```

Example

```
x = 10
if x > 5:
    print("x is greater than 5")

x is greater than 5
```

elif Statement:

The elif statement is used in conjunction with an if statement to test multiple conditions sequentially. If the preceding if condition is false, the code inside the elif block is checked. If the condition specified in elif is true, the code block inside that elif statement is executed. You can have multiple elif statements after an if, and only one block of code among them will execute (the first one with a true condition).

Syntax

if condition1:

```
# Code to execute if condition1 is true
```

elif condition2:

```
# Code to execute if condition2 is true
```

elif condition3:

```
# Code to execute if condition3 is true
```

Example

```
x = 10
if x > 20:
    print("x is greater than 20")
elif x > 15:
    print("x is greater than 15 but not 20")
elif x > 5:
    print("x is greater than 5 but not 15")
else:
    print("x is not greater than 5")
```

```
x is greater than 5 but not 15
```

else Statement:

The else statement is used in conjunction with an if statement to specify a block of code to execute when the if condition is false. It does not have a condition of its own and is typically used as the last part of a conditional statement.

Syntax

if condition:

```
# Code to execute if the condition is true
```

else:

```
# Code to execute if the condition is false
```

Example

```python
age = 17
if age >= 18:
    print("You are an adult")
else:
    print("You are a minor")
```

```
You are a minor
```

These conditional statements allow you to make decisions in your code and execute different code blocks based on those decisions. You can also nest if, elif, and else statements within each other to create more complex decision-making logic.

Some more Examples

```python
# Ask the user to enter their exam score
score = 65

# Determine the grade based on the score
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"

# Provide feedback to the user based on their grade
if grade == "A":
    feedback = "Excellent!"
elif grade == "B":
    feedback = "Good job!"
elif grade == "C":
    feedback = "Not bad, but you can do better."
elif grade == "D":
    feedback = "You need improvement."
else:
    feedback = "Sorry, you failed."
```

```
print("Your grade is:", grade)
print("Feedback:", feedback)

Your grade is: D
Feedback: You need improvement.
```

Nested Conditions

Nested conditions in programming involve placing one or more if statements within another if, elif, or else block. They enable the evaluation of multiple conditions in a structured hierarchy, allowing for complex decision-making. When one condition is satisfied, additional checks can be performed within that branch. Nested conditions facilitate branching logic and handling intricate scenarios based on multiple criteria. Proper code indentation is crucial for readability when using nested conditions.

Example

```
# Ask the user for a number
number = 25

# Check if the number is greater than 0
if number > 0:
    print('Number is greater than 0.')

    # Check if the number is positive
    if number % 2 == 0:
        print('Number is positive and even.')
    else:
        print('Number is positive and odd.')

# Check if the number is less than 0
elif number < 0:
    print('Number is less than 0.')

    # Check if the number is negative
    if number % 2 == 0:
        print('Number is negative and even.')
    else:
        print('Number is negative and odd.')

# If neither condition is met, the number is 0
else:
    print('Number is equal to 0.')

Number is greater than 0.
Number is positive and odd.
```

Two conditions in one if statement

```
    refers to the practice of combining two separate conditions using
logical operators like and or or within a single if statement. This
allows you to evaluate both conditions simultaneously, and the if
block will execute if both conditions are true.
```

For example, consider the following code snippet:

if condition1 and condition2:

```
# Code to execute if both condition1 and condition2 are true
```

In this case:

If condition1 is true, and condition2 is also true, then the code block inside the if statement will execute. If either condition1 or condition2 (or both) is false, the code block will not execute. Using two conditions in one if statement allows you to create more complex decision logic by requiring multiple criteria to be met simultaneously for the code within the if block to run.

Example

This example takes a student's test score and the number of absences as input. It then determines the student's grade based on two conditions within each grade range: a specific score range (e.g., 80-89) and a condition that checks if the number of absences is less than 5. If both conditions are met, the student receives a corresponding grade (e.g., 'A'). Otherwise, they receive an 'F'. The result is printed as the final grade.

```
# Ask the user for their test score and the number of absences
score = 87
absences = 3

# Determine the grade based on the score and absences
if (80 < score < 90) and absences < 5:
    grade = 'A'
elif (70 <= score < 80) and absences < 5:
    grade = 'B'
elif (60 <= score < 70) and absences < 5:
    grade = 'C'
elif (50 <= score < 60) and absences < 5:
    grade = 'D'
else:
    grade = 'F'

print('Your grade is:', grade)
```

```
Your grade is: A
```

More Examples

```
# Ask for product information
price = 250
available = "yes"

# Categorize the product based on price and availability
if price >= 100 and available == 'yes':
    category = 'Expensive and available'
elif price >= 100 and available == 'no':
    category = 'Expensive but not available'
elif price < 100 and available == 'yes':
    category = 'Affordable and available'
else:
    category = 'Affordable but not available'

print('Product category:', category)
```

```
Product category: Expensive and available
```

This code collects product price and availability information and categorizes the product into four categories based on whether it's expensive (price >= 100) and available ("yes" or "no"). The determined category is then displayed based on these conditions, providing insights into the product's status and pricing.

Conditional Programming with Boolean Variables

Boolean variables, expressed as True or False, are pivotal for conditional programming. They enable decision-making by evaluating whether a condition holds or not. Using logical operators, you can combine and manipulate Boolean variables to control program flow and respond to specific scenarios.

Example: Nested Conditions with Boolean Variable Updates

```
# Boolean variables representing membership and payment status
is_member = True
has_paid = False

if is_member:
    if has_paid:
        print("Thank you for your payment.")
    else:
        print("Please make a payment to renew your membership.")
        # Update the has_paid variable
        has_paid = True
else:
    print("Join our membership program for exclusive benefits.")

# Check the updated has_paid variable
if has_paid:
    print("Your membership is active.")
```

Example: Event Eligibility Checker with Conditional Statements and Boolean Operators

```
# Ask the user for their age and whether they have a ticket
age = 23
has_ticket = "yes"

# Check eligibility for entry to an event
if age >= 18 and has_ticket == 'yes':
    print('You are eligible to enter the event.')
elif age < 18 or not has_ticket == 'yes':
    print('You are not eligible to enter the event.')
else:
    print('Please provide valid information.')
```