Chapter 6: Mastering Control Structures in Python

Introduction

Control structures are the backbone of programming logic, allowing us to dictate the flow of our programs. In Python, control structures like conditional statements and loops empower developers to write efficient, dynamic, and versatile code. Whether you're a beginner looking to grasp the essentials or an experienced coder polishing your skills, understanding these concepts is crucial for any programming journey.

In this chapter, we'll dive into control structures, focusing on conditional statements (`if`, `elif`, `else`), loops (`for`, `while`), and the powerful technique of list comprehensions. By the end, you'll have a solid understanding of how to control the flow of your Python programs, making them more efficient and easier to manage.

Conditional Statements: Making Decisions in Code

Conditional statements allow your program to make decisions based on certain conditions. Think of them as the "if this, then that" logic of your code.

The `if` Statement :-The `if` statement is the most basic form of a conditional in Python. It allows you to execute a block of code only if a specified condition is true. In this example, the program checks if `age` is 18 or more. If the condition is met, it prints "You are an adult."

```
age = 20
if age >= 18:
    print("You are an adult.")
```

The `elif` Statement :- What if you have multiple conditions to check? This is where `elif` (short for "else if") comes in. It allows you to test additional conditions if the previous `if` statement is false. Here, the program first checks if `age` is 18 or more. If not, it checks if `age` is 13 or more. If neither condition is true, the program executes the `else` block.

The `else` Statement :-The `else` statement provides an alternative block of code that runs if none of the previous conditions are met. In the previous example, if `age` is less than 13, the `else` block executes, printing "You are a child."

```
age = 16

if age >= 18:
    print("You are an adult.")
elif age >= 13:
    print("You are a teenager.")
else:
    print("You are a child.")
```

Loops: Repeating Actions in Code

Loops allow you to repeat a block of code multiple times. This is incredibly useful for tasks that require repetitive actions, like iterating over a list of items.

The `for` Loop :- The `for` loop is used to iterate over a sequence (like a list, tuple, or string) and execute a block of code for each item in that sequence.

```
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

This loop will print each fruit in the list:

```
apple
banana
cherry
```

The `while` Loop :-The `while` loop runs as long as a specified condition is true. It's useful when the number of iterations isn't predetermined.

```
count = 1

while count <= 5:
    print("Count:", count)
    count += 1
```

This loop will print the count from 1 to 5:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

List Comprehensions: Create Lists

List comprehensions offer a concise way to create lists in Python. They can be used to generate a new list by applying an expression to each item in an existing list, optionally filtering items with a conditional.

Basic List Comprehension

Here's a simple example that creates a list of squares:

```
squares = [x**2 for x in range(10)]
print(squares)
```

This will output:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

List Comprehension with Conditionals

You can also add a condition to filter items. For instance, to create a list of even squares:

```
even_squares = [x**2 for x in range(10) if x % 2 == 0]
print(even_squares)
```

This will output:

```
[0, 4, 16, 36, 64]
```

Here are some coding questions related to control structures,

1. Question: FizzBuzz Challenge

Write a Python program that prints the numbers from 1 to 50. For multiples of three, print "Fizz" instead of the number, and for the multiples of five, print "Buzz". For numbers that are multiples of both three and five, print "FizzBuzz".

Solution:

```
for num in range(1, 51):
    if num % 3 == 0 and num % 5 == 0:
        print("FizzBuzz")
    elif num % 3 == 0:
        print("Fizz")
    elif num % 5 == 0:
        print("Buzz")
    else:
        print(num)
```

2. Question: Calculate Factorial

Write a Python function to calculate the factorial of a number using a `while` loop.

Solution:

```
def factorial(n):
    result = 1
    while n > 0:
        result *= n
        n -= 1
    return result

# Test the function
print(factorial(5))  # Output: 120
```

3. Question: List of Even Numbers

Using list comprehension, write a Python program that generates a list of even numbers between 1 and 100.

Solution:

```
even_numbers = [x for x in range(1, 101) if x % 2 == 0]
print(even_numbers)
```

4. Question: Grade Calculator

Write a Python program that takes a numerical score as input and prints the corresponding letter grade based on the following scale:

- A: 90-100
- B: 80-89
- C: 70-79
- D: 60-69
- F: Below 60

Solution:

```
def get_grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"

# Test the function
score = 85
print(f"Score: {score}, Grade: {get_grade(score)}")  # Output: Score: 85, Grade: B
```

5. Question: Sum of Digits

Write a Python program that takes a number and returns the sum of its digits. Use a `while` loop to achieve this.

Solution:

```
def sum_of_digits(num):
    total = 0
    while num > 0:
        total += num % 10
        num //= 10
    return total
```

```
# Test the function
print(sum_of_digits(1234))  # Output: 10
```

6. Question: Multiplication Table

Write a Python program to print the multiplication table for a given number using a `for` loop.

Solution:

```
def multiplication_table(n):
    for i in range(1, 11):
        print(f"{n} x {i} = {n * i}")

# Test the function
multiplication_table(5)
```

7. Question: Filter Words by Length

Write a Python program using list comprehension to filter a list of words and return only those that are longer than a given length.

Solution:

```
words = ["apple", "banana", "cherry", "date", "elderberry", "fig", "grape"]
length = 5

filtered_words = [word for word in words if len(word) > length]
print(filtered_words)  # Output: ['banana', 'cherry', 'elderberry']
```

8. Question: Check Prime Number

Write a Python program that checks whether a given number is prime. Use a `for` loop and conditional statements.

Solution:

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# Test the function
print(is_prime(11))  # Output: True
print(is_prime(25))  # Output: False
```

9. Question: Find Common Elements

Write a Python program to find the common elements between two lists using list comprehension.

Solution:

```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]

common_elements = [x for x in list1 if x in list2]
print(common_elements)  # Output: [4, 5]
```

10. Question: Reverse a String

Write a Python program to reverse a string using a `for` loop.

Solution:

```
def reverse_string(s):
    reversed_s = ""
    for char in s:
        reversed_s = char + reversed_s
    return reversed_s

# Test the function
print(reverse_string("hello"))  # Output: "olleh"
```
