Week 3: Python Control Flow

1 Introduction

Control flow is a fundamental aspect of programming that allows developers to dictate the order in which statements are executed based on certain conditions or to repeat actions. Mastering control flow is crucial for building dynamic applications that can respond to user input and data effectively. This report will discuss the key components of control flow in Python, including conditional statements, loops, control statements, and exception handling, along with relevant examples.

2 Conditional Statements

Conditional statements enable the execution of different blocks of code based on specific conditions. In Python, the primary conditional statements are `if`, `elif`, and `else`.

2.1 Example

```python
temperature = 30

if temperature > 30:
    print("It's a hot day.")
elif temperature > 20:
    print("It's a nice day.")
else:
    print("It's a bit chilly.")
```

In this example, the program checks the value of `temperature` and prints a message based on the evaluated condition. If the temperature is above 30, it prints that it's a hot day; if it's above 20 but not more than 30, it prints that it's a nice day, and for all other values, it states it's chilly.

3 Loops

Loops allow a set of instructions to be executed repeatedly based on a condition. Python supports two types of loops: `for` loops and `while` loops.

3.1 For Loops

The `for` loop is used to iterate over a sequence (like a list or a string).

3.1.1 Example

```python
for i in range(5):
    print("Iteration:", i)
```

This loop will print "Iteration: 0" through "Iteration: 4", demonstrating how the loop iterates through a sequence of numbers.

3.2 While Loops

The `while` loop continues to execute as long as a specified condition remains true.

3.2.1 Example

```python
count = 0
while count < 5:
    print(count)
    count += 1
```

In this example, the `while` loop continues until `count` is no longer less than 5, printing values from 0 to 4.

4 Control Statements

Control statements manage the execution flow of loops, allowing for more complex behaviors.

4.1 Break Statement

The `break` statement immediately terminates the loop.

4.1.1 Example

```python
for num in range(10):
    if num == 5:
        break
    print(num)
```

This loop will print numbers 0 through 4 and exit when `num` reaches 5.

4.2 Continue Statement

The `continue` statement skips the current iteration and continues with the next iteration.

4.2.1 Example

```python
for num in range(5):
    if num % 2 == 0:
        continue
    print(num)
```

This code prints only the odd numbers from 0 to 4.

4.3 Pass Statement

A `pass` statement acts as a placeholder that does nothing; it's often used in situations where a statement is syntactically required but no action is needed.

4.3.1 Example

```python
for num in range(5):
    if num == 3:
        pass  # Do nothing for 3
    print(num)
```

This will print all numbers from 0 to 4, but will do nothing when `num` is 3.

5 Exception Handling

Exception handling allows a program to manage errors gracefully. In Python, this is achieved using `try`, `except`, and `finally` blocks.

5.1 Example

```python
try:
    x = int(input("Please enter a number: "))
    result = 10 / x
except ValueError:
    print("Invalid input; please enter a number.")
except ZeroDivisionError:
    print("You cannot divide by zero.")
finally:
    print("This executes no matter what.")
```

In this program, if the user inputs a non-numeric value, a `ValueError` is caught. If the user inputs zero, a `ZeroDivisionError` is caught. The `finally` block executes regardless of whether an exception occurred, ensuring important cleanup or finalization can take place.

6 Conclusion

Control flow is essential in Python programming, allowing developers to write programs that can make decisions and repeat actions based on varying conditions. Mastery of conditional statements, loops, control statements, and exception handling is crucial for developing robust and effective Python applications. Through the examples provided, students should now have a clearer understanding of how to implement these concepts in their programming projects.

Exercises: Python Control Flow Exercises with Solutions

1. **Positive or Negative:** Write a program that checks if a number entered by the user is positive, negative, or zero.

Solution:

```python
number = float(input("Enter a number: "))
if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")

```

2. **Largest of Three Numbers:** Create a program that takes three numbers as input from the user and prints the largest of the three.
Solution:

```python
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))
c = float(input("Enter third number: "))
largest = max(a, b, c)
print(f"The largest number is {largest}.")

```

3. **Grade Evaluation:** Write a program that prompts the user to enter their score (0-100) and prints ”Excellent” if the score is above 85, ”Good” if it’s between 70 and 85, and ”Needs Improvement” if it’s below 70.
Solution:

```python
score = float(input("Enter your score (0-100): "))
if score > 85:
    print("Excellent")
elif score >= 70:
    print("Good")
else:
    print("Needs Improvement")

```

4. **Simple Interest Calculator:** Develop a program that calculates simple interest using the formula Interest = $P \times R \times T$ where P is the principal, R is the rate of interest, and T is the time in years.
Solution:

```python
P = float(input("Enter principal amount: "))
R = float(input("Enter rate of interest (in %): ")) / 100
T = float(input("Enter time (in years): "))
interest = P * R * T
print(f"The simple interest is: {interest}")

```

5. **Even Numbers in a Range:** Create a program that prints all even numbers between 1 and 100.
Solution:

```python
print("Even numbers between 1 and 100:")
for i in range(1, 101):
    if i % 2 == 0:
        print(i, end=' ')
print()

```

6. **Multiples of a Number:** Write a program that takes a number from the user and prints all its multiples up to 100.
Solution:

```python
number = int(input("Enter a number: "))
print(f"Multiples of {number} up to 100:")
for i in range(1, 101):
    if i % number == 0:
        print(i, end=' ')
print()

```

7. **Countdown from a Given Number:** Write a program that takes a number from the user and counts down to zero, printing each number.
Solution:

```python
countdown_start = int(input("Enter a number to start countdown: "))
while countdown_start >= 0:
    print(countdown_start)
    countdown_start -= 1

```

8. **Factorial Calculation:** Develop a program that computes the factorial of a given non-negative integer using a loop.
Solution:

```python
number = int(input("Enter a non-negative integer: "))
factorial = 1
for i in range(1, number + 1):
    factorial *= i
print(f"The factorial of {number} is {factorial}.")

```

9. **Sum of Natural Numbers:** Create a program that calculates the sum of all natural numbers up to a number provided by the user.
Solution:

```python
n = int(input("Enter a number: "))
total = sum(range(1, n + 1))
print(f"The sum of natural numbers up to {n} is {total}.")

```

10. **Prime Number Checker:** Write a program that checks whether a number entered by the user is prime.
Solution:

```python
number = int(input("Enter a number: "))
is_prime = True
if number > 1:
    for i in range(2, int(number ** 0.5) + 1):
        if number % i == 0:
            is_prime = False
            break
    if is_prime:
        print(f"{number} is a prime number.")
    else:
        print(f"{number} is not a prime number.")
else:
    print(f"{number} is not a prime number.")

```

11. **Fibonacci Sequence:** Create a program that prints the Fibonacci sequence up to n terms, where n is provided by the user.
Solution:

```python
n = int(input("Enter the number of terms in Fibonacci sequence: "))
a, b = 0, 1
print("Fibonacci sequence:")
for _ in range(n):
    print(a, end=' ')
    a, b = b, a + b
print()

```

12. **Vowel or Consonant:** Write a program that checks if a character entered by the user is a vowel or a consonant.
Solution:

```python
char = input("Enter a character: ").lower()
if char in 'aeiou':
    print(f"{char} is a vowel.")
else:
    print(f"{char} is a consonant.")

```

13. **Guess the Number Game:** Create a simple number guessing game where the program selects a random number between 1 and 100, and the user has to guess it.
Solution:

```python
import random

number_to_guess = random.randint(1, 100)
guess = None

while guess != number_to_guess:
    guess = int(input("Guess a number between 1 and 100: "))
    if guess < number_to_guess:
        print("Too low!")
    elif guess > number_to_guess:
        print("Too high!")
print("Congratulations! You've guessed the number.")

```

14. **Multiplication Table:** Develop a program that generates the multiplication table for a number entered by the user.
Solution:

```python
number = int(input("Enter a number for multiplication table: "))
print(f"Multiplication table for {number}:")
for i in range(1, 11):
    print(f"{number} x {i} = {number * i}")

```

15. **Count Occurrences of a Character:** Write a program that counts how many times a specific character appears in a string entered by the user.
Solution:

```python
string = input("Enter a string: ")
char = input("Enter the character to count: ")
count = string.count(char)
print(f"The character '{char}' appears {count} times in the string.")

```

16. **Check for Palindrome:** Create a program that checks if a string entered by the user is a palindrome.
Solution:

```python
string = input("Enter a string: ")
if string == string[::-1]:
    print(f"{string} is a palindrome.")
else:
    print(f"{string} is not a palindrome.")

```

17. **Simple Calculator:** Write a program that acts as a simple calculator that can perform addition, subtraction, multiplication, and division based on user input.
Solution:

```python
def calculator():
    num1 = float(input("Enter first number: "))
    operator = input("Enter an operator (+, -, *, /): ")
    num2 = float(input("Enter second number: "))

    if operator == '+':
        result = num1 + num2
    elif operator == '-':
        result = num1 - num2
    elif operator == '*':
        result = num1 * num2
    elif operator == '/':
        if num2 != 0:
            result = num1 / num2
        else:
            return "Division by zero is not allowed."
    else:
        return "Invalid operator."

    return f"The result is: {result}"

print(calculator())

```

18. **Pattern Printing:** Develop a program that prints a pyramid pattern of asterisks based on a height provided by the user.
Solution:

```python
height = int(input("Enter the height of the pyramid: "))
for i in range(height):
    print(' ' * (height - i - 1) + '*' * (2 * i + 1))

```

19. **Sum of Odd Numbers:** Write a program that calculates the sum of all odd numbers from 1 to a given number.
Solution:

```python
n = int(input("Enter a number: "))
total = sum(i for i in range(1, n + 1) if i % 2 != 0)
print(f"The sum of odd numbers from 1 to {n} is {total}.")

```

20. **Days in a Month:** Create a program that asks the user for a month (as a number) and checks how many days are in that month, considering leap years.
Solution:

```python
month = int(input("Enter month number (1-12): "))
year = int(input("Enter year: "))

if month == 2:
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        print("February has 29 days.")
    else:
        print("February has 28 days.")
elif month in [4, 6, 9, 11]:
    print(f"The month has 30 days.")
else:
    print(f"The month has 31 days.")

```

Problems to be Solved

1. **Check Divisibility:** Write a program that takes an integer input from the user and checks if it is divisible by both 3 and 5. If it is, print "Divisible by 3 and 5"; otherwise, print "Not divisible by 3 and 5".
2. **Sum of Digits:** Create a program that takes a positive integer from the user and calculates the sum of its digits. For example, if the user enters 123, the program should return 6.
3. **Count Even and Odd Numbers:** Develop a program that prompts the user to enter a series of numbers (the input should continue until the user enters 0). The program should then count how many of those numbers are even and how many are odd.
4. **Factorial Calculation:** Write a program that takes a positive integer from the user and calculates its factorial. Use a loop to perform the calculation, and print the result.
5. **Multiplication Table:** Create a program that generates a multiplication table for a number entered by the user, up to 12.
6. **Largest Number in a List:** Write a program that prompts the user to enter a list of numbers (the input should continue until the user enters -1). The program should then find and print the largest number in the list.
7. **Prime Number Checker:** Develop a program that checks whether a number provided by the user is a prime number. If it is prime, print "Prime"; otherwise, print "Not Prime".
8. **Grade Calculator:** Create a program that takes a student's score as input and outputs the corresponding grade according to the following scale:
   - A: 90-100
   - B: 80-89
   - C: 70-79
   - D: 60-69
   - F: Below 60
9. **Count Vowels:** Write a program that takes a string input from the user and counts how many vowels (a, e, i, o, u) are present in the string.
10. **Simple Banking System:** Develop a program that simulates a simple banking system. The user should be able to perform the following actions:
    - Check balance
    - Deposit money
    - Withdraw money

    Ensure to validate the withdrawal amount to ensure the user does not withdraw more than the available balance.