Python Lecture 3: Nested Branching, Looping, Break/Continue, and Examples

September 02, 2025

1 Revision of Lecture 2 Programs

To reinforce concepts from Lecture 2, we revise three programs with slight modifications for clarity. These cover arithmetic operators, relational/logical operators, and basic branching.

1.1 Revised Program 1: Basic Arithmetic (from Lecture 2, Program 1)

Write a program that takes two numbers and computes their sum, difference, product, and division, with added error handling for division by zero.

.

```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
sum_result = num1 + num2
diff_result = num1 - num2
prod_result = num1 * num2
print("Sum:", sum_result)
print("Difference:", diff_result)
print("Product:", prod_result)
if num2 != 0:
    div_result = num1 / num2
    print("Division:", div_result)
else:
    print("Division: Cannot divide by zero")
```

Example Input: 10, 2
Output: Sum: 12.0
Difference: 8.0
Product: 20.0
Division: 5.0

1.2 Revised Program 2: Even or Odd (from Lecture 2, Program 5.2.1)

Write a program that determines if a number is even or odd, with added input validation.

.

```
try:
    num = int(input("Enter a number: "))
    if num % 2 == 0:
        print("The number is even.")
    else:
```

```
        print("The number is odd.")
except ValueError:
    print("Invalid input; please enter an integer.")
```

Example Input: 4
Output: The number is even.
Example Input: invalid
Output: Invalid input; please enter an integer.

1.3 Revised Program 3: Grading System (from Lecture 2, Program 5.3.1)

Write a program that assigns a grade based on score, with range check.

.

```
score = float(input("Enter score (0-100): "))
if score < 0 or score > 100:
    print("Invalid score; must be between 0 and 100.")
else:
    if score >= 90:
        print("Grade: A")
    elif score >= 80:
        print("Grade: B")
    elif score >= 70:
        print("Grade: C")
    else:
        print("Fail")
```

Example Input: 85
Output: Grade: B
Example Input: 101
Output: Invalid score; must be between 0 and 100.

2 Nested If Statements

Nested if statements involve placing an if-else structure inside another if or else block. This allows for more complex decision-making by checking multiple conditions hierarchically. Indentation is crucial; each nested level adds 4 spaces.

Structure:

```
if condition1:
    # code if condition1 True
    if condition2:
        # code if condition1 and condition2 True
    else:
        # code if condition1 True but condition2 False
else:
    # code if condition1 False
```

Write a program to check if a number is positive and even.

.

```
num = int(input("Enter a number: "))
if num > 0:
    if num % 2 == 0:
```

```
        print("Positive and even.")
    else:
        print("Positive but odd.")
else:
    print("Non-positive.")
```

Example Input: 4
Output: Positive and even.
Example Input: -1
Output: Non-positive.

3 Control Statements by Repetition (Looping)

Looping allows code to repeat until a condition is met. Python has two main loops: while and for.

3.1 While Loop

Executes as long as a condition is true. Useful for unknown iteration counts.

Structure:

```
while condition:
    # code to repeat
```

Example: Print numbers 1 to 5.

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

Output: 1
2
3
4
5

3.2 For Loop

Iterates over a sequence (e.g., range, list). Useful for known iteration counts.

Structure:

```
for variable in sequence:
    # code to repeat
```

Example: Print numbers 1 to 5.

```
for i in range(1, 6):
    print(i)
```

Output: 1
2
3
4
5

4 Break and Continue

- **break**: Exits the loop immediately. - **continue**: Skips the rest of the current iteration and goes to the next.

Example with break:

```
for i in range(1, 10):
    if i == 5:
        break
    print(i)
```

Output: 1
2
3
4

Example with continue:

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

Output: 1
2
4
5

5 Example Programs

Below are 10 programs demonstrating nested branching, loops, break, and continue step by step, each with expected output.

5.1 Program 1: Nested If for Age Group

Write a program that classifies age into child, teen, or adult.

.

```
age = int(input("Enter age: "))
if age > 0:
    if age < 13:
        print("Child")
    elif age < 20:
        print("Teen")
    else:
        print("Adult")
else:
    print("Invalid age")
```

Example Input: 15
Output: Teen
Example Input: -1
Output: Invalid age

5.2 Program 2: Nested If for Login Check

Write a program that simulates simple login with username and password check.

.

```
username = input("Enter username: ")
if username == "admin":
    password = input("Enter password: ")
    if password == "1234":
        print("Login successful")
    else:
        print("Invalid password")
else:
    print("Invalid username")
```

Example Input: admin, 1234
Output: Login successful
Example Input: user, 1234
Output: Invalid username

5.3 Program 3: While Loop for Sum

Write a program that sums numbers until user enters 0.

0.

```
total = 0
num = int(input("Enter number (0 to stop): "))
while num != 0:
    total += num
    num = int(input("Enter number (0 to stop): "))
print("Sum:", total)
```

Example Input: 5, 3, 0
Output: Sum: 8

5.4 Program 4: For Loop with Range

Write a program that prints even numbers from 0 to 10.

0 10.

```
for i in range(0, 11, 2):
    print(i)
```

Output: 0
2
4
6
8
10

5.5 Program 5: While with Break

Write a program that finds first even number greater than input.

.

```
num = int(input("Enter starting number: "))
while True:
    num += 1
    if num % 2 == 0:
        print("First even:", num)
        break
```

Example Input: 3
Output: First even: 4

5.6 Program 6: For with Continue

Write a program that prints numbers 1-10, skipping multiples of 3.

1-10 3.

```
for i in range(1, 11):
    if i % 3 == 0:
        continue
    print(i)
```

Output: 1
2
4
5
7
8
10

5.7 Program 7: Nested Loop (For in For)

Write a program that prints a 3x3 grid.

3x3.

```
for row in range(1, 4):
    for col in range(1, 4):
        print(row * col, end=" ")
    print()
```

Output: 1 2 3
2 4 6
3 6 9

5.8 Program 8: While with Nested If

Write a program that counts positive/negative numbers until 0.

/ 0.

```
positive = 0
negative = 0
num = int(input("Enter number (0 to stop): "))
while num != 0:
    if num > 0:
        positive += 1
    else:
```

```
        negative += 1
    num = int(input("Enter number (0 to stop): "))
print("Positive:", positive, "Negative:", negative)
```

Example Input: 5, -3, 0
Output: Positive: 1 Negative: 1

5.9 Program 9: For with Break in Nested Loop

Write a program that breaks out of inner loop when condition met.

.

```
for i in range(1, 4):
    for j in range(1, 4):
        print(i, j)
        if j == 2:
            break
```

Output: 1 1
1 2
2 1
2 2
3 1
3 2

5.10 Program 10: Infinite Loop with Break and Continue

Write a program that echoes input until "quit", skips "skip".

"quit" "skip".

```
while True:
    text = input("Enter text (quit to stop): ")
    if text == "quit":
        break
    if text == "skip":
        continue
    print("Echo:", text)
```

Example Input: hello, skip, world, quit
Output: Echo: hello
Echo: world

6 Assignments

Below are 5 assignments to test your understanding of nested branching, loops, break, and continue. Solutions are not provided; practice by writing and testing the code.

6.1 Assignment 1: Nested If for Eligibility

Write a program that checks if a person is eligible for a job: age >=18 and experience >2 years (nested inside age check). Print appropriate messages.

: >=18 >2 ( ). .

6.2 Assignment 2: While Loop for Factorial

Write a program to calculate factorial of a number using while loop. Handle invalid input.

while. .

6.3 Assignment 3: For Loop with Continue

Write a program to print numbers 1-20, skipping those divisible by 5 using continue.

1-20 5 continue.

6.4 Assignment 4: Nested Loop Pattern

Write a program to print a triangle pattern using nested for loops (e.g., 1
22
333).

for (1
22
333).

6.5 Assignment 5: Game Loop with Break/Continue

Write a program to simulate a simple guessing game: guess a number 1-10, continue until correct or "quit" to break.

: 1-10 "quit" .