Python: Flow Control Statements

Connect with me: Youtube | LinkedIn | WhatsApp Channel | Web | Facebook | Twitter

- Download PDF
- To access the updated handouts, please click on the following link: https://yasirbhutta.github.io/python/docs/control-flow.html

Flow control statements

Flow control statements in Python determine the order in which your code is executed. They allow you to make decisions, repeat actions, and control the program's flow based on specific conditions.

1. Conditional Statements (if, else, elif)
2. Looping Statements (for, while)
3. Loop Control Statements (break, continue, pass)
4. Match Statement (Python 3.10+)

Conditional Statements (if, else, elif)

- The `if` statement in Python is a conditional statement that allows you to execute a block of code only if a certain condition is met.
- Make decisions using `if`, `elif`, and `else` statements.

The general **syntax** of the if statement is as follows:

```
if condition:
    # code to execute if condition is True
```

The `condition` can be any logical expression. If the condition is evaluated to `true`, the block of `statements` is executed. Otherwise, the block of `statements` is skipped.

Here is a simple example of an if statement in MATLAB:

Example #:

```
x = 10

if x > 5:
    print('x is greater than 5.')
```

This code will print the message `x is greater than 5.` to the console.

You can also use `elif` statements to check for multiple conditions. The general **syntax** of the **elif statement** is as follows:

```
if condition1:
    # code to execute if condition1 is True
elif condition2:
    # code to execute if condition2 is True
```

If the `condition` for the if statement is evaluated to `false`, the python interpreter will check the `condition` for the first elif statement. If the condition for the elif statement is evaluated to `true`, the corresponding block of `statements` is executed. Otherwise, the python interpreter will check the `condition` for the next elif statement, and so on.

Here is an example of an if statement with an elif statement:

Example #1:

```
x = 3

if x > 5:
    print('x is greater than 5.')
elif x < 5:
    print('x is less than 5.')
```

This code will print the message "x is less than 5." to the console.

You can also use an else statement to check for all other conditions. The general syntax of the else statement is as follows:

```
if condition1:
    # code to execute if condition1 is True
elif condition2:
    # code to execute if condition2 is True
else:
    # code to execute if none of the conditions are True
```

If all of the conditions for the if and elseif statements are evaluated to `false`, the block of `statements` in the `else` statement is executed.

Here is an example of an if statement with an `elif` statement and an `else` statement:

```
x = 2

if x > 5:
    print('x is greater than 5.')
elif x == 5:
    print('x is equal to 5.')
else:
    print('x is less than 5.')
```

Output: This code will print the message "x is less than 5." to the console.

Example 2: Basic If-Else

```python
number = 10

if number > 0:
    print("The number is positive")
else:
    print("The number is not positive")
```

Explanation: This code checks if the variable `number` is greater than 0. If true, it prints "The number is positive", otherwise it prints "The number is not positive".

Example 3: Checking Even or Odd related video:

```python
number = 7

if number % 2 == 0:
    print("The number is even")
else:
    print("The number is odd")
```

Explanation: This code checks if the variable `number` is even or odd. If `number % 2` equals 0, it is even; otherwise, it is odd.

Example 4: Age Group Classification

```python
age = 25

if age < 18:
    print("Minor")
else:
    print("Adult")
```

Explanation: This code classifies a person as a "Minor" if their age is less than 18, and as an "Adult" otherwise.

Example 5: Grade Assignment [video]

```python
score = 85

if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Grade: F")
```

Explanation: This code assigns a grade based on the `score`. It uses multiple `elif` statements to check for different score ranges.

Example 6: Nested If-Else

```python
number = -5

if number >= 0:
    if number == 0:
        print("The number is zero")
    else:
        print("The number is positive")
else:
    print("The number is negative")
```

Explanation: This code uses nested `if-else` statements to check if the number is zero, positive, or negative.

Example 7: Temperature Check

```python
temperature = 30

if temperature > 30:
    print("It's a hot day")
else:
    print("It's not a hot day")
```

Explanation: This code checks if the temperature is greater than 30. If true, it prints "It's a hot day"; otherwise, it prints "It's not a hot day".

Example 8: Voting Eligibility [video]

```python
age = 17

if age >= 18:
    print("You are eligible to vote")
else:
    print("You are not eligible to vote")
```

Explanation: This code checks if a person is eligible to vote based on their age. If `age` is 18 or more, it prints "You are eligible to vote"; otherwise, it prints "You are not eligible to vote".

Example 9: Password Check

```
password = "password123"

if password == "password123":
    print("Access granted")
else:
    print("Access denied")
```

Explanation: This code checks if the `password` is correct. If it matches "password123", it prints "Access granted"; otherwise, it prints "Access denied".

Example 10: Maximum of Two Numbers

```
a = 15
b = 20

if a > b:
    print("a is greater than b")
else:
    print("a is not greater than b")
```

Explanation: This code checks which number is greater between `a` and `b`. If `a` is greater than `b`, it prints "a is greater than b"; otherwise, it prints "a is not greater than b".

Example 11: Checking String Length

```
string = "Hello, World!"

if len(string) > 10:
    print("The string is long")
else:
    print("The string is short")
```

Explanation: This code checks if the length of the `string` is greater than 10. If true, it prints "The string is long"; otherwise, it prints "The string is short".

Example #12: Using if-elif-else

```
x = 10

if x < 0:
    print("Negative")
elif x == 0:
    print("Zero")
else:
    print("Positive")
```

- Video: Use Walrus Operator with if-else (New)
- Video: How to Write Single-Line Code Instead of If-Else Statements

Example #:

- Python Quiz -IF

loops

- There are two ways to create loops in Python: with the for-loop and the while-loop.
- Repeat actions using `for` and `while` loops.

for loop

- A for loop in Python is a programming statement that repeats a block of code a fixed number of times.
- The for-loop is always used in combination with an iterable object[^1], like a list or a range.
- The Python for statement iterates over the members of a sequence in order, executing the block each time.

video: What is it and Why do we Use it? | Python For loop Tutorial

Syntax:

```
for item in iterable:
    # code block
```

`iterable` is a sequence of elements such as a list, tuple, dictionary, set, or string. item is a variable that takes on the value of each element in the sequence, one at a time. The code block is executed once for each element in the sequence.

range() function:

- We can use the `range()` function as an iterable in a for loop in Python. The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.
- We can also specify the starting value and the increment value of the sequence using the range() function. For example, range(2, 10, 2) returns a sequence of numbers starting from 2, incrementing by 2, and ending at 8. read more ...

Here are the common usages:

1. **Single Argument (`stop`):**

```
range(stop)
```

- Generates numbers from `0` up to, but not including, `stop`.

```
range(5)  # Output: 0, 1, 2, 3, 4
```

2. **Two Arguments (`start`, `stop`)**:

```
range(start, stop)
```

- Generates numbers from `start` up to, but not including, `stop`.

```
range(2, 5)  # Output: 2, 3, 4
```

3. **Three Arguments (`start`, `stop`, `step`)**:

```
range(start, stop, step)
```

- Generates numbers from `start` up to, but not including, `stop`, incrementing by `step`. The `step` can be positive or negative.

```
range(1, 10, 2)  # Output: 1, 3, 5, 7, 9
range(10, 1, -2) # Output: 10, 8, 6, 4, 2
```

Note: `range()` produces an immutable sequence type, which is often used in loops. To get a list of numbers, you can convert the `range` object to a list:

```
list(range(5))  # Output: [0, 1, 2, 3, 4]
```

- Video: The range() Function
- Video: Use of range() in for loop

Example #1: Print Numbers from 1 to 5 [video]

Question: Write a Python program to print the numbers from 1 to 5, using a for loop.

```python
for i in range(6):
    print(i)
```

Example #2: Printing "Building the future, one line at a time." 5 Times Using a for Loop

Question: Write a Python program to print the string "Building the future, one line at a time." 5 times, using a for loop.

```python
for i in range(5):
    print("Building the future, one line at a time.")
```

Example #3: Sum of Numbers from 1 to N

Question: Write a python program to calculate the sum of the first N natural numbers using a for loop.

video: Calculate the sum of the first N natural numbers | Python for loop example

Example #4: Print Even Numbers from 2 to 10

Question: Write a Python program to display the even numbers from 2 to 10, inclusive, using a for loop.

```python
sum = 0
for i in range(2,11):
    if i%2 == 0:
        sum +=i
        print(i)
print(f"Sum of even numbers: {sum}")
```

Example #5: video: String as an iterable

Example #6: video: How to Print Multiplication Tables in Python

Example #7: Lists as an iterable

```python
collection = ['python', 5, 'd']
for x in collection:
    print(x)
```

Example #7: Loop over Lists of lists

```python
list_of_lists = [ [1, 2, 3], [4, 5, 6], [7, 8, 9]]
for list in list_of_lists:
    for x in list:
        print(x)
```

- A **nested loop** is a loop inside another loop. It is a powerful programming technique that can be used to solve a wide variety of problems.

Example #:8 Nested Loops - Multiplication Tables

```
for i in range(1, 11):
    print(f"Multiplication table of {i}")
    for j in range(1, 11):
        print('%d * %d = %d' % (i, j, i*j))
```

Why We Use Variables in For Loops

In Python, a variable in a `for` loop is used to iterate over a sequence (like a list, tuple, string, or range) and access each element in that sequence one at a time. This variable is often called the "loop variable" or "iterator variable."

Understanding the Role of the Variable:

1. **Accessing Elements**: The loop variable allows you to access each element in the sequence during each iteration of the loop. This makes it possible to perform operations on each element.
2. **Dynamic Assignment**: The loop variable automatically takes the value of the next element in the sequence during each iteration. This saves you from having to manually update the variable's value.
3. **Readability and Simplicity**: Using a loop variable makes the code more readable and easier to understand. You can name the variable in a way that reflects the data it represents, making the code more self-explanatory.
4. **Control Over Iteration**: The loop variable gives you control over the loop’s execution. You can use it to control the flow, such as skipping certain elements, breaking out of the loop early, or performing specific actions based on the variable's value.

Example:

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

In this example, the loop variable `fruit` takes on the value of each element in the list `fruits` during each iteration. The output will be:

```
apple
banana
cherry
```

The variable `fruit` is used to access and print each element in the `fruits` list.

Use of underscore in 'For Loop'

- In Python, an underscore (_) is often used as a variable name in a for loop (or any other context) when the value of the variable is not needed.
- This is a common convention to indicate that the value is intentionally being ignored or discarded.

For example, if you want to repeat an action a certain number of times but don't need to use the loop variable, you can use `_`:

```
for _ in range(5):
    print("Hello")
```

Here, `_` is used instead of a variable name like `i` or `j` because the value is not important. The loop will simply print "Hello" five times without using the loop index. This helps to make the code more readable by signaling to other programmers that the loop variable is not used in the loop's body.

- video: Underscore to Ignore Values in for loop

while loop

- A while loop in python is a control flow statement that repeatedly executes a block of code until a specified condition is met.

Syntax:

```
while condition:
    # code block
```

Here, condition is a boolean expression that is evaluated before each iteration of the loop. If the condition is `True`, the code block is executed. The loop continues to execute as long as the condition remains True.

video: Python while loop example - Learn how to use while loop

Example #: Print numbers from 1 to 10 using while loop

Question: Write a Python program to print the numbers from 1 to 10, using a while loop?

```
count = 1  # Start counting at 1
while count <= 10:  # Keep counting as long as we're less than or equal to 10
    print(count)  # Print the current number
    count += 1  # Add 1 to the count for the next round
```

Example #: Print "Hello, world!" 5 times using while loop

Question: Write a Python program to print the string "Hello, world!" 5 times, using a while loop?

```
i = 1
while i <= 10:
    print('Hello, world!')
    i += 1
```

Example #: Sum of numbers from 1 to 100 using while loop

Question: Write a MATLAB program to calculates the sum of the numbers from 1 to 100 using a while loop.

```
i = 1
sum = 0

while i <= 100:
    sum += i
    i += 1

print(f'Sum = {sum}');
```

Example #: Sum of even numbers from 2 to 20 using while loop

Question: Write a Python program to calculate the sum of the even numbers from 2 to 20 using a while loop.

```
sum = 0  # Initialize a variable to store the sum
number = 1

while number <= 20:
    if number%2 == 0:
      sum += number  # Add the current number to the sum
    number += 1

print(f'The sum of even numbers from 1 to 20 is: {sum}')
```

Example: Square of numbers less than 5 using while loop

Question : Write a program that prints the sum of the squares of all the numbers from 1 to 4, using a while loop.

```
i = 1
while i < 5:
    square = i ** 2
    print(f'Square of {i} is {square}')
    i += 1
```

Output:

```
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
```

Example #:

Question: Write a Python program to prompt the user to enter lines of text until the user enters a blank line. The program should then display the message "You entered a blank line.".

```
inputStr = 'Start';

while inputStr != "":
  inputStr = input("Enter a line of text:")

print('You entered a blank line.')
```

Example: Sum of given numbers till the number entered is zero

Question: Write a Python program to add all the numbers entered by the user until the user enters zero. The program should display the sum of the numbers.

```
sum = 0; # Initialize the sum

# Prompt the user to enter a number
number = int(input('Enter a number: ')) # int() Convert string input to integer

# While the number entered is not zero, add the number to the sum and prompt the
user to enter another number
while number != 0:
    sum += number
    number = int(input('Enter another number: ')) # int() Convert string input to
integer

# Display the sum of the numbers
print(f'The sum of the numbers is: {sum}')
```

Example : While loop from 1 to infinity, therefore running forever.

```
x = 1
while True:
    print("To infinity and beyond! We're getting close, on %d now!" % (x))
    x += 1
```

while loop examples:

See also:

- Video: Learn how to use while loops
- Video: Learn how to use INFINITE while loop
- Video: Python while loop

video: Python Loops Performance Comparison: For vs. While | Which is Faster?

Loop Control Statements (break, continue, pass)

These statements modify the behavior of loops.

break: Terminates the loop entirely. **continue:** Skips the current iteration and moves to the next one. **pass:** Does nothing, often used as a placeholder.

`break`

Exits the loop prematurely.

```
for item in sequence:
    if some_condition:
        break  # exit the loop
```

`continue`

Skips the current iteration and proceeds to the next iteration of the loop.

```
for item in sequence:
    if some_condition:
        continue  # skip the rest of the code in this iteration
    # code to execute if some_condition is False
```

`pass`

A null statement, used as a placeholder.

```
if condition:
    pass  # do nothing
```

Example #:

```
for x in range(3):
    if x == 1:
        break
```

Example #:

```
for i in range(10):
    if i == 5:
        break  # exit the loop when i is 5
    if i % 2 == 0:
        continue  # skip even numbers
    print(i)  # print only odd numbers less than 5
```

- Video: How to Effectively Use Break and Continue Statements
- Video: Using Python break statement with a while loop

The else Clauses on Loops

In Python, the `else` clause can be used with loops (`for` and `while`). This may be surprising at first since most people associate `else` with `if` statements. However, in loops, the `else` clause has a unique behavior:

- The `else` block is executed **only if the loop completes all its iterations without encountering a `break` statement**.
- If the loop is exited early because of a `break`, the `else` block is skipped.

The `else` clause in loops (`for` and `while`) in Python is a bit unusual because most people associate `else` with `if` statements. In the context of loops, the `else` clause is executed only when the loop finishes normally, meaning it wasn't interrupted by a `break` statement.

How It Works:

1. **With a `for` loop**:
    - The `else` block runs if the loop completes all its iterations without hitting a `break`.
    - If the loop is terminated by a `break`, the `else` block is skipped.
2. **With a `while` loop**:
    - The `else` block runs if the `while` loop condition becomes `False` naturally.
    - If the loop is terminated by a `break`, the `else` block is skipped.

Example with a `for` loop

Let's say we're searching for a specific number in a list.

```
# List of numbers
numbers = [1, 2, 3, 4, 5]

# Number we want to find
target = 6

# Iterate over the list
for num in numbers:
    if num == target:
        print("Found the target!")
        break
else:
    print("Target not found in the list.")
```

Explanation:

- The loop checks each number to see if it matches the `target`.
- If the number is found, the loop breaks, and the `else` clause is skipped.
- If the loop finishes without finding the target (i.e., without a `break`), the `else` block runs, printing "Target not found in the list."

Example with a `while` loop

```
# Counter
i = 1

# Loop condition
while i <= 5:
    if i == 3:
        print("Breaking the loop")
        break
    print(i)
    i += 1
else:
    print("Loop finished without breaking.")
```

Explanation:

- The loop runs while `i` is less than or equal to 5.
- If `i` equals 3, the loop breaks.
- Since the loop is broken before it naturally ends, the `else` block is skipped.

Why Use the `else` Clause with Loops?

Using `else` with loops can be helpful when you're performing a search or some operation where you want to know if the loop completed successfully or was interrupted by a `break`. It's a clean way to handle scenarios where the loop might end early.

Example #: for..else

```
for x in range(3):
    print(x)
else:
    print('Final x = %d' % (x))
```

Match Statement (Python 3.10+)

The match statement offers a more concise way to handle multiple comparison cases.

```
def get_fruit_color(fruit):
    match fruit:
        case "apple":
            return "red"
        case "banana":
            return "yellow"
        case _:
            return "unknown"
```

Key Terms

- for
- while
- range
- pass
- else
- match
- break
- continue
- f-string

Fix the errors in Python

Sure! Here are more examples of Python code challenges with mistakes in control flow statements. Each one is followed by an explanation and the corrected code.

1. **Mistaken Use of `elif` Instead of `if`**

Code:

```
x = 7

if x > 5:
    print("x is greater than 5")
elif x > 0:
    print("x is positive")
```

Mistake:
Using `elif` after the first condition is true means the second condition (`x > 0`) will never be checked.

Corrected Code:

```
x = 7

if x > 5:
    print("x is greater than 5")
if x > 0:
    print("x is positive")
```

- Now, both conditions are checked independently.

2. **Infinite `while` Loop Due to Missing Update Statement**

Code:

```
n = 10

while n > 0:
    print(n)
```

Mistake:
The loop will run infinitely because `n` is never updated, so `n > 0` is always true.

Corrected Code:

```
n = 10

while n > 0:
    print(n)
    n -= 1  # Decrease n by 1 each iteration
```

- The loop now terminates when `n` becomes 0.

3. **Incorrectly Using `for` Loop with `break`**

Code:

```
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num == 3:
        print("Found 3!")
        break
    print(num)
print("Loop finished")
```

Mistake:
While this code is technically correct, it can be misleading because it suggests that the loop will continue after finding `3`, but it actually stops.

Improved Code:

```python
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num == 3:
        print("Found 3!")
        break
    print(num)
else:
    print("3 was not found")

print("Loop finished")
```

- The `else` statement would run if the loop completes without breaking, indicating `3` wasn't found.

4. `continue` Misused in `while` Loop

Code:

```python
i = 0

while i < 5:
    if i == 2:
        continue
    print(i)
    i += 1
```

Mistake:
The `continue` statement skips the `i += 1` statement, causing an infinite loop when `i == 2`.

Corrected Code:

```python
i = 0

while i < 5:
    if i == 2:
        i += 1  # Move this before the continue to avoid skipping it
        continue
    print(i)
    i += 1
```

- Now, `i` is correctly incremented before `continue`, preventing the infinite loop.

5. **Logical Error in `if-else` Statements**

Code:

```
age = 20

if age < 18:
    print("Minor")
elif age >= 18:
    print("Adult")
else:
    print("Invalid age")
```

Mistake:
The `else` block here is unnecessary and misleading, as the condition is fully handled by `if-elif`.

Improved Code:

```
age = 20

if age < 18:
    print("Minor")
else:
    print("Adult")
```

- The code is more concise and still covers all cases.

6. **Using `pass` in an Incorrect Context**

Code:

```
x = 10

if x > 0:
    pass
    print("x is positive")
else:
    pass
    print("x is not positive")
```

Mistake:
The `pass` statement is unnecessary and misleading here, as it suggests there might be some unimplemented logic.

Corrected Code:

```
x = 10

if x > 0:
    print("x is positive")
else:
    print("x is not positive")
```

- The `pass` statement is removed, clarifying the logic.

8. **Incorrect Loop Boundaries in `for` Loop**

Code:

```
for i in range(1, 5):
    print(i)
```

Mistake:
The loop runs from `1` to `4`, not including `5`. This might be misunderstood if you expect `5` to be printed.

Clarification:

```
for i in range(1, 6):  # Adjust the range to include 5
    print(i)
```

- Now, the loop includes `5` as intended.

9. **Using `break` After `else` in Loop**

Code:

```
for i in range(5):
    print(i)
else:
    break  # SyntaxError: 'break' outside loop
```

Mistake:
You can't use `break` after the `else` block of a loop, as it doesn't make sense.

Corrected Code:

```
for i in range(5):
    if i == 2:
        break
    print(i)
```

```
else:
    print("Loop completed without break")
```

- Now, `break` is used correctly within the loop, and the `else` block handles the case where the loop completes.

10. **Incorrect Use of Multiple `elif` Conditions**

Code:

```python
score = 75

if score >= 90:
    print("A")
elif score >= 80:
    print("B")
elif score >= 70:
    print("C")
elif score >= 60:
    print("D")
elif score < 70:
    print("F")
```

Mistake:
The last `elif` block (`elif score < 70`) is redundant because the `elif score >= 70` block already covers it.

Improved Code:

```python
score = 75

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

- The `else` block now correctly handles any score below `60`.

Which of the following will NOT cause a syntax error in Python?

A)

```
1st_variable = 10
```

B)

```
if x == 10:
print("x is 10")
```

C)

```
print("Hello World!"
```

D)

```
print "Hello World!)
```

1. **Incorrect `if` Statement Syntax**

Code:

```
x = 10

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

Mistake:
The `if` statement is missing a colon (`:`) at the end.

Corrected Code:

```
x = 10

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

2. **Incorrect Indentation in `if-else` Statement**

Code:

```
y = -3
```

```
if y > 0:
    print("y is positive")
    print("This is always printed")
else:
print("y is not positive")
```

Mistake:
The `else` block is not properly indented.

Corrected Code:

```
y = -3

if y > 0:
    print("y is positive")
    print("This is always printed")
else:
    print("y is not positive")
```

3. `while` Loop with Incorrect Condition

Code:

```
count = 10

while count > 10:
    print("This will never print")
    count -= 1
```

Mistake:
The loop condition `count > 10` is false at the start, so the loop will never run.

Corrected Code:

```
count = 10

while count >= 0:
    print("Count:", count)
    count -= 1
```

4. `for` Loop with Incorrect Range

Code:

```
for i in range(1, 10, -1):
    print(i)
```

Mistake:
The step value in `range(1, 10, -1)` is negative, but the start value is less than the stop value, so this loop will not run.

Corrected Code:

```
for i in range(10, 0, -1):
    print(i)
```

- This prints numbers from 10 to 1.

5. **Using `break` Incorrectly in a Loop**

Code:

```
for i in range(5):
    if i == 2:
        break
    else:
        print(i)
```

Mistake:
The `else` block is not correctly used here. It runs as part of each iteration, not after the loop.

Corrected Code:

```
for i in range(5):
    if i == 2:
        break
    print(i)
else:
    print("Loop completed without a break")
```

- The `else` block will only run if the loop completes without a `break`.

6. **`try` Block with Unhandled Exception**

Code:

```
try:
    print(5 / 0)
```

```
except TypeError:
    print("A TypeError occurred")
```

Mistake:
The code raises a `ZeroDivisionError`, but the `except` block is only catching a `TypeError`.

Corrected Code:

```
try:
    print(5 / 0)
except ZeroDivisionError:
    print("You can't divide by zero!")
```

7. **Misuse of `continue` Statement**

Code:

```
for i in range(5):
    if i == 3:
        continue
        print("This will never be printed")
    print(i)
```

Mistake:
The `print` statement after `continue` will never be executed.

Corrected Code:

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

- The code correctly skips printing `3` and continues with the next iteration.

8. **Logical Error with `if-elif` Statements**

Code:

```
z = 20

if z > 30:
    print("z is greater than 30")
elif z > 10:
    print("z is greater than 10")
```

```
else:
    print("z is less than 10")
```

Mistake:
The code will print `"z is greater than 10"` even though `z` is also greater than 30. This is logically correct, but not what might be intended if you want to check for ranges.

Corrected Code:

```
z = 20

if z > 30:
    print("z is greater than 30")
elif z > 20:
    print("z is greater than 20 but less than 30")
elif z > 10:
    print("z is greater than 10")
else:
    print("z is less than or equal to 10")
```

- Now, this checks for more specific conditions.

9. **Incorrect Usage of `pass` in Control Flow**

Code:

```
for i in range(5):
    if i == 2:
        pass
        print("This should not print")
    print(i)
```

Mistake:
The `pass` statement does nothing, but the `print("This should not print")` will still run.

Corrected Code:

```
for i in range(5):
    if i == 2:
        pass
    print(i)
```

- Now, the loop will simply skip doing anything when `i` is `2`, without any unintended behavior.

10. **Improper Use of `finally` in Exception Handling**

Code:

```
try:
    result = 10 / 0
finally:
    print("This will run no matter what.")
except ZeroDivisionError:
    print("You can't divide by zero!")
```

Mistake:
The `finally` block must come after `except`, not before.

Corrected Code:

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("You can't divide by zero!")
finally:
    print("This will run no matter what.")
```

These challenges are designed to highlight common mistakes that occur when working with control flow statements in Python. Working through these examples will help reinforce proper syntax and logical structure.

True/False (Mark T for True and F for False)

Multiple Choice (Select the best answer)

if statement (MCQs)

What is the output of the following code Python Quiz #16

```
x = 10
y = 5

if x == y * 2:
    print("True")
else:
    print("False")
```

```
- A) True
- B) False
- C) Error
- D) Nothing
```

Watch this video for the answer: https://youtube.com/shorts/ExDu2lwjd3c?si=VVq47sCNgRWd7l_i

What is the output of the following code? Python Quiz #31

```
x = 3
y = 5

if x == 3 and y != 5:
    print("True")
else:
    print("False")
```

```
- A) True
- B) False
- C) Syntax error
- D) Name error
```

Watch the video for the answer: https://youtube.com/shorts/7uAgT3_P4Oc?si=evBHSymYx2kVaEEk

13. **What will be the output of the following code?** Python Quiz #39

```
x = 10
if  5 < x < 15:
    print("x is between 5 and 15")
else:
    print("x is not between 5 and 15")
```

```
- A) x is between 5 and 15
- B) x is not between 5 and 15
- C) Error
- D) No output
```

Watch the video for the answer:https://youtu.be/TkmqlDK8oCg

Which of the following correctly fixes the syntax error in the code below? Python Quiz #40

```
# fixes the error
x = 10
if x == 10
    print("x is 10")
```

```
- A) Remove the comment.
- B) Add a colon after `if x == 10`.
- C) Add parentheses around `x == 10`.
- D) Indent the print statement correctly.
```

Watch the video for the answer: https://youtu.be/3010E1WuHd8

1. **Which of the following correctly represents the syntax of an if statement in Python?**
    - A) if condition { block of code }
    - B) if(condition) { block of code }
    - C) if condition: block of code
    - D) None
2. **What is the syntax for a simple if statement in Python?**
    - A) if x == 10:
    - B) for x in range(10):
    - C) while x < 10:
    - D) if (x == 10) then:
3. **What is the purpose of the else block in an if statement?**
    - A) To execute a code block when the if condition is True
    - B) To execute a code block when the if condition is False
    - C) To create an infinite loop
    - D) To define a function
4. **What happens when none of the conditions in an if-elif-else chain are True, and there is no else block?**
    - A) The program raises an error.
    - B) The program executes the first if block.
    - C) The program executes the last elif block.
    - D) The program does nothing and continues to the next statement.
5. **What will be the output of the following code?**

```python
x = 10
y = 5
if x < y:
    print("x is greater than y")
```

```
- A) "x is greater than y"
- B) "x is less than y"
- C) Nothing will be printed
- D) An error will occur
```

6. **What happens if none of the conditions in an if-elif chain are True, and there is no else block?**
   - A) The program raises an error
   - B) The program executes the first if block
   - C) The program does nothing and continues to the next statement
   - D) The program executes the last elif block
7. **What will be the output of the following code?**

```
number = 10
if number % 2 == 0:
    print("Number is even")
else:
    print("Number is odd")
```

```
- A) Number is odd
- B) Number is even
- C) Nothing
- D) Error
```

8. **Which of the following is the correct way to compare if two variables are equal in an if statement?**
   - A) if x equals 5:
   - B) if x == 5:
   - C) if x = 5:
   - D) if x := 5:
9. **How can you check if a value is NOT equal to 10 in an if statement?**
   - A) if x != 10:
   - B) if x <> 10:
   - C) if x =! 10:
   - D) if x not 10:
10. **What does the following code snippet do?**

```
if x > 0:
    print("Positive")
elif x < 0:
    print("Negative")
else:
    print("Zero")
```

```
- A) Prints "Positive" if x is greater than 0, "Negative" if x is less than 0, and
"Zero" if x is 0.
- B) Prints "Positive" if x is less than 0, "Negative" if x is greater than 0, and
"Zero" if x is 0.
- C) Prints "Positive" if x is 0, "Negative" if x is greater than 0, and "Zero" if
x is less than 0.
- D) Causes an error because the conditions are conflicting.
```

11. **What is the correct syntax for an if-else statement?**

- A) if condition: code block
- B) if condition: code block else: code block
- C) if condition: code block elif condition: code block
- D) if condition: code block else: code block elif condition: code block

12. **What happens if none of the conditions in an if-elif-else block are true?**

    - A) The program terminates
    - B) The else block is executed
    - C) An error occurs
    - D) The program continues without executing any block

13. **What will be the output of the following code?**

```
num = 7
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

```
- A) Even
- B) Odd
- C) Error
- D) No output
```

15. **What will be the output of the following code?**

```
score = 85
if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
else:
    print("Grade: F")
```

```
- A) Grade: A
- B) Grade: B
- C) Grade: C
- D) Grade: F
```

16. **What is the output of the following code?**

```
x = 5
if x > 10:
    print("Greater than 10")
elif x > 5:
    print("Greater than 5 but not greater than 10")
else:
    print("5 or less")
```

```
- A) Greater than 10
- B) Greater than 5 but not greater than 10
- C) 5 or less
- D) No output
```

Answer: C

17. **Which of the following is true about the `elif` statement?**

    - A) `elif` stands for "else if" and is used to check additional conditions after the `if` statement
    - B) `elif` is the same as `else`
    - C) `elif` must be followed by an `else` statement
    - D) You can have multiple `elif` statements in one block

Answer: A, D

18. **What will be the output of the following code?**

```
y = 20
if y < 20:
    print("Less than 20")
elif y == 20:
    print("Equal to 20")
```

```
else:
    print("Greater than 20")
```

```
- A) Less than 20
- B) Equal to 20
- C) Greater than 20
- D) None of the above
```

Answer: B

19. **What will be the output of the following code snippet?**

```
a = 10
b = 5
if a > b:
    if a > 0:
        print("a is positive")
    else:
        print("a is non-positive")
else:
    print("a is less than or equal to b")
```

```
- A) a is positive
- B) a is non-positive
- C) a is less than or equal to b
- D) No output
```

Answer: A

20. **Which of the following statements is false about `if-elif-else` statements in Python?**

    - A) `elif` statements are optional in an `if-elif-else` construct
    - B) An `if` statement can be followed by multiple `elif` statements
    - C) An `else` statement is mandatory in an `if-elif-else` construct
    - D) An `if` statement can exist without `elif` or `else`

Answer: C

21. **What will be the output of the following code?**

```
x = 0
if x:
    print("True")
```

```
else:
    print("False")
```

```
- A) True
- B) False
- C) No output
- D) Error
```

Answer: B

Truthy and Falsy Values in Python In Python, certain values are considered False when evaluated in a boolean context (like in an if statement). These values include:

- None
- False
- 0 (zero of any numeric type, including 0.0)
- Empty sequences and collections (e.g., '', (), [], {})
- Any object that implements **bool** or **len** to return False or 0, respectively

22. **Which of the following expressions can be used in an `if` condition to check if a number `n` is between 1 and 100 inclusive?**
    - A) `if n > 1 and n < 100:`
    - B) `if n >= 1 and n <= 100:`
    - C) `if 1 <= n <= 100:`
    - D) `if n == 1 or n == 100:`

Answer: B, C

23. **What will be the output of the following code?**

```
z = 7
if z < 5:
    print("Less than 5")
elif z == 10:
    print("Equal to 10")
elif z > 5:
    print("Greater than 5 but not 10")
else:
    print("Something else")
```

```
- A) Less than 5
- B) Equal to 10
- C) Greater than 5 but not 10
- D) Something else
```

Answer: C

24. **What is the purpose of the `else` statement in an `if-elif-else` structure?**
    - A) To check another condition if the previous ones were false
    - B) To execute a block of code if none of the previous conditions were true
    - C) To execute a block of code if the previous conditions were true
    - D) To end the `if-elif-else` structure

Answer: B

25. **What will be the output of the following code?**

```
x = 10
if x > 5:
    if x == 10:
        print("x is 10")
    else:
        print("x is greater than 5 but not 10")
else:
    print("x is 5 or less")
```

```
- A) x is 10
- B) x is greater than 5 but not 10
- C) x is 5 or less
- D) No output
```

Answer: A

26. **Which of the following is correct syntax for an `if-elif-else` statement in Python?**

A)

```
if x > y:
print("x is greater")
elif x == y:
print("x is equal to y")
else:
print("x is less")
```

B)

```
if x > y:
    print("x is greater")
elif x == y:
    print("x is equal to y")
```

```
else:
    print("x is less")
```

C)

```
if (x > y):
    print("x is greater")
elif (x == y):
    print("x is equal to y")
else:
    print("x is less")
```

D)

```
if x > y:
    print("x is greater")
elif x == y
    print("x is equal to y")
else
    print("x is less")
```

Answer: B, C

27. **What will be the output of the following code?**

```
a = 3
b = 4
if a > b:
    print("a is greater")
elif a == b:
    print("a and b are equal")
else:
    print("b is greater")
```

```
- A) a is greater
- B) a and b are equal
- C) b is greater
- D) No output
```

Answer: C

28. **Which of the following statements will execute if `n = 7`?**

```
if n < 5:
    print("n is less than 5")
elif n < 10:
    print("n is less than 10")
else:
    print("n is 10 or more")
```

```
- A) n is less than 5
- B) n is less than 10
- C) n is 10 or more
- D) None of the above
```

Answer: B

29. **Which keyword is used to check an additional condition if the initial `if` statement is `False`?**

    - A) else
    - B) elif
    - C) elseif
    - D) if

Answer: B

30. **What is the result of the following code?**

```
x = 0
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

```
- A) Positive
- B) Zero
- C) Negative
- D) No output
```

Answer: B

31. **How many `elif` clauses can be included in an `if-elif-else` statement?**

    - A) One
    - B) Two

    - C) Unlimited
    - D) None

Answer: C

32. **What will be the output of the following code?**

```
age = 20
if age < 18:
    print("Minor")
elif age >= 18 and age < 65:
    print("Adult")
else:
    print("Senior")
```

```
- A) Minor
- B) Adult
- C) Senior
- D) No output
```

Answer: B

for loop (MCQs)

What is the output of the following code snippet in Python? Python Quiz #3

```
for i in range(10):
    if i % 2 == 0:
        print(i)
```

A) 0 1 2 3 4 5 6 7 8 9 B) 0 2 4 6 8 C) 2 4 6 8 D) IndentationError: expected an indented block

Watch the video for answer: https://bit.ly/3WKH9wE

What is the correct syntax for a for loop in Python?

- A) for (int i = 0; i < 10; i++):
- B) for i in range(10):
- C) for i = 0 to 9:
- D) for i in 10:

watch the video for the answer: https://youtu.be/2mhrDgBEp10

What will be the output of the following code? Python Quiz #36

```
for i in range(5):
  print(i * 2)
```

```
- A) 0 1 2 3 4
- B) 2 4 6 8 10
- C) 10 8 6 4 2
- D) 0 2 4 6 8
```

Watch this video for the answer: https://bit.ly/3WqjjEP

How many times is the print statement executed? Python Quiz #37

```
for i in range(3):
    for j in range(2):
        print(f"i = {i}, j = {j}")
```

```
- A) 3 times
- B) 5 times
- C) 6 times
- D) 9 times
```

Watch this video for the answer: https://youtu.be/CYeZI3uCiTI

What is the primary purpose of a for loop in Python?

```
- A) To define a function
- B) To iterate over a sequence
- C) To create a conditional statement
- D) To perform mathematical operations
```

In Python, what does the range() function do when used in a for loop? - A) Generates a sequence of numbers - B) Defines a list - C) Calculates the average - D) Determines the length of a string

How can you create a nested loop in Python?

```
- A) by using a loop inside another loop with proper indentation
- B) by using a loop inside another loop with parentheses
- C) by using a nested keyword before the inner loop
- D) by using a colon after the outer loop and before the inner loop
```

Answer: A

Question 4

Consider the following code:

```
total = 0
for i in range(1, 6):
    total += i
print(total)
```

What does this code accomplish?

A) It prints numbers from 1 to 5.

B) It calculates the sum of numbers from 1 to 5.

C) It prints the sum of numbers from 1 to 6.

D) It calculates the sum of numbers from 0 to 5.

Explanation: The loop iterates over the range from 1 to 5 (inclusive), summing up the values. The `total` variable accumulates this sum.

Answer: B

Question 6

How can a `for` loop be used in a real-life scenario involving data processing?

A) To count the number of words in a large document.

B) To open a web browser.

C) To create a new file on the desktop.

D) To turn off a computer.

Explanation: A `for` loop can be used to iterate through the words in a document to count them, making it useful for data analysis tasks.

Answer: A

Why might you use a `for` loop instead of manually performing repeated tasks?

```
- A) To reduce the chance of human error.
- B) To make the program run slower.
- C) To avoid using variables.
- D) To limit the program to one iteration.
```

Explanation: Using a `for` loop automates repetitive tasks, which helps prevent errors and saves time, especially when processing large datasets or performing repetitive calculations.

Answer: A

What does the following code print?

```
for x in range(5, 8):
    print(x)
```

```
- A) 5 6 7
- B) 5 6 7 8
- C) 4 5 6 7
- D) 5 6 7 8 9
```

Explanation: The `range(5, 8)` function generates numbers starting from 5 up to, but not including, 8.

Answer: A

What does the following code output?

```
for i in range(1, 4):
    for j in range(1, 3):
        print(i, j)
```

A) 1 1 1 2 2 1 2 2 3 1 3 2

B) 1 2 2 3 3 4

C) 1 3 2 3 3 3

D) 1 1 2 2 3 3

Explanation: This is a nested loop, where the outer loop runs from 1 to 3 (inclusive) and the inner loop runs from 1 to 2 (inclusive). It prints all combinations of `i` and `j`.

Answer: A

What will be the output of this code?

```
for i in range(3):
    print(i * i)
```

A) 0 1 4

B) 0 1 2

C) 1 4 9

D) 0 2 4

Explanation: The loop iterates over the range 0, 1, 2. For each iteration, it prints the square of the current index `i`.

Answer: A

What will the following code output?

```
for i in range(5, 10, 2):
    print(i)
```

A) 5 7 9

B) 5 6 7 8 9

C) 5 7 9 11

D) 5 7 8

Explanation: The `range(5, 10, 2)` generates numbers starting from 5 up to, but not including, 10 with a step of 2, resulting in 5, 7, and 9.

Answer: A

while loop (MCQs)

1. **What is the output of the following code snippet in Python?** Python Quiz #7

```
i = 1
while i < 10:
    print(i)
    i += 2
```

```
- A) 1 2 3 4 5 6 7 8 9
- B) 1 3 5 7 9
- C) 0
- D) IndentationError: expected an indented block
```

Watch the video for answer: https://youtube.com/shorts/zdLNmwO1u8Y

2. **What is the output of the following code snippet in Python?** Python Quiz #4

```
i = 0
while i < 5:
    print(i)
    i += 1
else:
    print("Done")
```

```
- A) 0 1 2 3 4 Done
- B) 0 1 2 3 Done
- C) SyntaxError: invalid syntax
- D) IndentationError: expected an indented block
```

Watch this video for the answer: https://youtube.com/shorts/9Zw-LuNX9h0

4. **What is the output of the following code?** Python Quzi #38

```
x = 10
while x > 0:
    print(x)
    x -= 2
```

```
- A) 9 7 5 3 1
- B) 10 8 6 4 2
- C) 10 9 8 7 6
- D) The code will run indefinitely.
```

Watch this video for the answer: https://bit.ly/3AdTqka

What is the output of the following code?

```
count = 0
while count < 3:
    print(count)
    count += 1
```

```
- A) 0 1 2
- B) 0 1
- C) 1 2 3
- D) The code will run indefinitely.
```

What will happen if you try to modify the loop variable within the body of a while loop?

(a) The loop will continue as normal. (b) The loop will terminate immediately. (c) The loop may behave unexpectedly, depending on how the variable is modified. (d) The loop will always run indefinitely.

What is the correct syntax for a while loop in Python?

A) while (condition): B) while condition {} C) while condition: D) while (condition) {}

What is the purpose of the else clause in a while loop?

(a) To execute a block of code if the loop condition is never true. (b) To execute a block of code if the loop completes without being terminated by a break statement. (c) To execute a block of code if the loop encounters an error. (d) The else clause cannot be used with a while loop.

What is the difference between pass and return statements in Python?3

A) pass does nothing and return exits the function B) pass exits the function and return does nothing C) pass and return both do nothing D) pass and return both exit the function Answer: A

How can you create an if-elif-else statement without using elif keyword in Python?2

A) by using nested if-else statements B) by using logical operators with if-else statements C) by using multiple if statements with indentation D) none of the above Answer: A

What is the difference between break and continue statements in Python?3

A) break terminates the loop and continue skips the current iteration B) break skips the current iteration and continue terminates the loop C) break and continue both terminate the loop D) break and continue both skip the current iteration Answer: A

What is the benefit of using control structures in programming languages?3

A) to make the code more modular and reusable B) to control the flow of execution based on certain parameters or conditions C) to implement various algorithms and logic in the code D) all of the above Answer: D

How can you create an if-else statement in one line in Python?2

A) by using a ternary operator with condition and values B) by using a lambda function with condition and values C) by using a colon after if block and before else block D) by using parentheses around if block and else block Answer: A

Loop Control Statements (break, continue, pass) (MCQs)

1. **What is the output of the following code snippet in Python?** [Python Quiz #5]

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

```
- A) 0 1 2 3
- B) 0 1 2 4
- C) SyntaxError: invalid syntax
- D) IndentationError: expected an indented block
```

Watch this video for answer: https://youtube.com/shorts/x7CIxqoqccY

What is the output of the following code snippet in Python? [Python Quiz #6]

```python
for i in range(5):
    if i == 3:
        break
    print(i)
```

```
- A) 0 1 2
- B) 0 1 2 3
- C) 1 2 3
- D) 0 1 2 3 4
```

Watch this video for answer: https://youtube.com/shorts/XNLL6j-P61A

How can you create a nested if statement in Python?2

A) by using elif keyword B) by using else keyword C) by indenting the inner if block under the outer if block D) by using parentheses around the inner if block Answer: C What is the purpose of a pass statement in Python?3

A) to skip a block of code that is not needed B) to indicate that a block of code is empty or not implemented yet C) to exit a loop or a function prematurely D) to pass a value or an argument to another function Answer: B

What is the syntax of a while loop in Python?1

A) while condition: statement B) while (condition): statement C) while condition do statement D) while (condition) do statement Answer: A

What is the difference between a break statement and a continue statement in Python?4

A) break terminates the loop, continue skips to the next iteration B) break skips to the next iteration, continue terminates the loop C) break exits the program, continue pauses the program D) break resumes the program, continue exits the program Answer: A

What are the three types of control structures in Python?

A) Sequential, Selection and Repetition B) Conditional, Looping and Branching C) Function, Class and Module D) Input, Output and Processing Answer: A

What is the keyword used to make a selection statement in Python?2

A) select B) if C) switch D) case Answer: B

What are the two types of iteration statements in Python?1

A) while and do-while B) for and foreach C) while and for D) do-while and foreach Answer: C

What are the three basic types of control structures in Python? a) sequence, selection, repetition b) input, output, processing c) function, class, module d) list, tuple, dict Answer: a) sequence, selection, repetition

Which control structure is used to execute a block of code only if a certain condition is true? a) if statement b) for loop c) while loop d) break statement Answer: a) if statement

Which control structure is used to execute a block of code for each item in an iterable object? a) if statement b) for loop c) while loop d) break statement Answer: b) for loop

#51 In Python, What is the output of this code?

name = "Ahmad" if name == "Ahmad": print("Hello, Ahmad") elif name == "Ali": print("Hello, Ali") else: print("Hello, stranger")

Ahmad Hello, Ahmad Hello, stranger Syntax Error

#39 Which keyword is used to break out of a loop in Python?

Watch the Video Tutorial for the Answer: https://youtu.be/LfF9CsyVRgU

#python #pythonpoll #MCQsTest #yasirbhutta

a) stop b) break c) end d) exit

Answer: b) break

#37 What is the output of the following code?

i = 0 while i < 5: i += 1 if i == 3: break print(i) else: print("Loop completed")

Watch the Video Tutorial for the Answer: https://youtu.be/LfF9CsyVRgU

#python #pythonpoll #MCQsTest #yasirbhutta

a) 1 2 b) 1 2 Loop completed c) 1 2 3 d) 1 2 3 Loop completed

#34 What is the output of the following code?

for i in range(5): if i == 3: continue print(i)

#python #pythonpoll #MCQsTest #yasirbhutta

a) 0 1 2 4 b) 0 1 2 3 c) 1 2 4 d) 0 1 2 3 4

Answer: a) 0 1 2 4

Which loop is best suited for iterating over a range of numbers?

a) for loop b) while loop c) do-while loop d) foreach loop Answer: a) for loop

Which of the following loops is guaranteed to execute at least once? a) for loop b) while loop c) do-while loop d) foreach loop Answer: c) do-while loop

#36 Which keyword is used to skip to the next iteration of a loop in Python?

Watch the Video Tutorial for the Answer: https://youtube.com/shorts/lf6SYOMIJv8?feature=share

#python #pythonpoll #MCQsTest #yasirbhutta

a) skip b) continue c) next d) go Answer: b) continue

Which of the following is not a looping statement in Python? a) for b) while c) repeat d) do-while

Exercises

if statement

Certainly! Here are 30 exercises to help you practice using `if-elif-else` statements in Python:

1. **Check Positive, Negative, or Zero**

   ```
   # Write a program to check if a number is positive, negative, or zero.
   number = int(input("Enter a number: "))
   ```

2. **Even or Odd**

   ```
   # Write a program to check if a number is even or odd.
   number = int(input("Enter a number: "))
   ```

3. **Largest of Three Numbers**

   ```
   # Write a program to find the largest of three numbers.
   a = int(input("Enter first number: "))
   b = int(input("Enter second number: "))
   c = int(input("Enter third number: "))
   ```

4. **Grade Assignment**

   ```
   # Write a program to assign grades based on the score.
   score = int(input("Enter your score: "))
   ```

5. **Leap Year Check**

```python
# Write a program to check if a year is a leap year or not.
year = int(input("Enter a year: "))
```

6. **Age Group Classification**

```python
# Write a program to classify age into child, teenager, adult, and senior.
age = int(input("Enter your age: "))
```

7. **Temperature Description**

```python
# Write a program to describe the temperature (hot, warm, cold).
temperature = float(input("Enter the temperature: "))
```

8. **Voter Eligibility**

```python
# Write a program to check if a person is eligible to vote.
age = int(input("Enter your age: "))
```

9. **Password Strength Checker**

```python
# Write a program to check the strength of a password.
password = input("Enter your password: ")
```

10. **Simple Calculator**

```python
# Write a simple calculator program using if-elif-else.
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
operation = input("Enter operation (+, -, *, /): ")
```

11. **BMI Calculator**

```python
# Write a program to calculate BMI and categorize it (underweight, normal,
overweight, obese).
height = float(input("Enter your height in meters: "))
weight = float(input("Enter your weight in kilograms: "))
```

12. **Triangle Type Checker**

```python
# Write a program to check the type of a triangle (equilateral, isosceles,
scalene).
a = float(input("Enter first side: "))
b = float(input("Enter second side: "))
c = float(input("Enter third side: "))
```

13. **Day of the Week**

```python
# Write a program to print the day of the week based on a number (1 for
Monday, 7 for Sunday).
day_number = int(input("Enter a number (1-7): "))
```

14. **Letter Grade Assignment**

```python
# Write a program to convert a numerical score into a letter grade.
score = int(input("Enter your score: "))
```

15. **Month Days**

```python
# Write a program to print the number of days in a given month.
month = int(input("Enter the month number (1-12): "))
```

16. **Discount Calculator**

```python
# Write a program to calculate the discount on a product based on the price.
price = float(input("Enter the price of the product: "))
```

17. **Character Checker**

```python
# Write a program to check if a character is a vowel or consonant.
char = input("Enter a character: ").lower()
```

18. **Valid Triangle Checker**

```python
# Write a program to check if three given lengths can form a triangle.
a = float(input("Enter first side: "))
b = float(input("Enter second side: "))
c = float(input("Enter third side: "))
```

19. **Multiples of Five**

```python
# Write a program to check if a number is a multiple of five.
number = int(input("Enter a number: "))
```

20. **Alphabet Case Checker**

```python
# Write a program to check if an alphabet is uppercase or lowercase.
char = input("Enter a character: ")
```

21. **Login System**

```python
# Write a simple login system that checks username and password.
username = input("Enter username: ")
password = input("Enter password: ")
```

22. **Divisibility Check**

```python
# Write a program to check if a number is divisible by 3 and 5.
number = int(input("Enter a number: "))
```

23. **Prime Number Checker**

```python
# Write a program to check if a number is prime.
number = int(input("Enter a number: "))
```

24. **Digit or Alphabet Checker**

```python
# Write a program to check if a character is a digit or an alphabet.
char = input("Enter a character: ")
```

25. **Fuel Efficiency**

```python
# Write a program to categorize fuel efficiency (excellent, good, average,
poor).
mpg = float(input("Enter miles per gallon: "))
```

26. **Odd or Even List**

```python
# Write a program to classify numbers in a list as odd or even.
numbers = [int(x) for x in input("Enter numbers separated by space: ").split()]
```

27. **Quadratic Equation Solver**

```python
# Write a program to solve a quadratic equation and determine the nature of the roots.
a = float(input("Enter coefficient a: "))
b = float(input("Enter coefficient b: "))
c = float(input("Enter coefficient c: "))
```

28. **User Age Validator**

```python
# Write a program to validate the age input by the user (must be between 0 and 120).
age = int(input("Enter your age: "))
```

29. **Student Result System**

```python
# Write a program to check if a student passed or failed based on marks in three subjects.
marks1 = int(input("Enter marks in subject 1: "))
marks2 = int(input("Enter marks in subject 2: "))
marks3 = int(input("Enter marks in subject 3: "))
```

30. **Shopping Cart Total**

```python
# Write a program to calculate the total cost of items in a shopping cart and apply discount if applicable.
item1 = float(input("Enter price of item 1: "))
item2 = float(input("Enter price of item 2: "))
item3 = float(input("Enter price of item 3: "))
```

For loop Exercises

Basic Exercises

1. **Print numbers 1 to 10**: Write a `for` loop to print numbers from 1 to 10. Python Exercise Solution

2. **Print even numbers 2 to 20**: Write a `for` loop to print all even numbers from 2 to 20.

3. **Print each character in a string**: Write a `for` loop to iterate through the string "Hello, World!" and print each character.
4. **Sum of the first 10 natural numbers**: Use a `for` loop to calculate the sum of the first 10 natural numbers. Solution
5. **Find the factorial of a number**: Write a `for` loop to calculate the factorial of a given number, e.g., 5! = 5 × 4 × 3 × 2 × 1.
   - **Watch the Solution Now ✨**
6. **Squares of numbers 1 to 5**: Write a Python for loop that prints the square of each number from 1 to 5.
   - **Watch the Solution Now ✨**
7. **Count down from 10 to 1**: Use a `for` loop to print numbers from 10 down to 1. Solution
8. **Print "Python" 5 times**: Write a `for` loop to print the word "Python" five times. Solution **Intermediate Exercises**
9. **Print the first 10 Fibonacci numbers**: Write a `for` loop to generate the first 10 numbers in the Fibonacci sequence.
10. **Check if a number is prime**: Write a `for` loop to check if a given number is prime.
11. **Sum of digits of a number**: Write a `for` loop to calculate the sum of the digits of a given number, e.g., 1234.
12. **Print a pattern of stars**: Use nested `for` loops to print the following pattern:

```
*
**
***
****
*****
```

13. **Print odd numbers between 1 and 20**: Write a `for` loop to print all odd numbers between 1 and 20.
14. **Count vowels in a string**: Use a `for` loop to count the number of vowels in the string "Hello, World!".

Advanced Exercises

17. **Print the multiplication table for 7**: Write a `for` loop to print the multiplication table for the number 7.
18. **Generate a list of even numbers**: Use a `for` loop to generate a list of even numbers between 1 and 20.
19. **Print numbers divisible by 3 and 5 between 1 and 50**: Use a `for` loop to print all numbers between 1 and 50 that are divisible by both 3 and 5.
20. **Generate a list of the first 10 square numbers**: Write a `for` loop to create a list of the first 10 square numbers.
21. **Print a right-angle triangle of numbers**: Use nested `for` loops to print the following pattern:

```
1
12
123
1234
12345
```

22. **Print numbers that are multiples of 3 up to 30**: Use a `for` loop to print all multiples of 3 up to 30.
23. **Find numbers divisible by 7 and 11 between 1 and 100**: Use a `for` loop to print all numbers between 1 and 100 that are divisible by both 7 and 11.
24. **Calculate the harmonic sum of n numbers**: Write a

While loop Exercises

1. How to Count Digits in a Positive Integer using Python Solution
2. How to Count Occurrence of a Specific Digit in a Number using Python. Solution

Review Questions

for loop:

- What is the difference between a for loop and a while loop in Python?
- Can you use a for loop to iterate over a string?

if statement:

- Can you have multiple elif blocks in an if-elif-else statement?
- What is the purpose of nesting if statements?
-

References and Bibliography

[^1]: In Python, an iterable object is an object that you can loop over using a "for" loop. It's any object that can return its elements one at a time. [2] “ForLoop - Python Wiki,” Python.org, 2017. https://wiki.python.org/moin/ForLoop