PYTHON

LOOP AND CONDITIONAL STATEMENTS

PRESENTED BY JASPER ANGELES

PRE-TEST

Python Conditional and Loop Statements

Instruction: Students are to answer the questions to test their prior knowledge about the topic to be discussed today.

Pre-Test

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

A. To execute a set of statements as long as a condition is true.
B. To perform a specific action repeatedly until a condition is met.
C. To execute a set of statements once for each item in an iterable.
D. To execute a set of statements for a specific number of times.

2. Which of the following statements is true regarding the break statement in Python?

A) It is used to skip the current iteration of a loop.
B) It terminates the loop and transfers execution to the next statement after the loop.
C) It is used to create an infinite loop.
D) It can only be used with while loops.

3. What is the purpose of the elif statement in an if-elif chain?

A) To perform an action if the **if** condition is true.

B) To skip the current iteration of the loop.

C) To introduce an additional condition to check if the previous **if** condition was not met.

D) To execute a set of statements unconditionally.

4. In Python, what is the role of the else block in an if-elif-else statement?

A) It is used to skip the current iteration of the loop.

B) It introduces an additional condition after **elif**.

C) It is executed when none of the conditions in the **if** and **elif** blocks are true.

D) It is optional and can be omitted in all cases.

5. What is the difference between a for loop and a while loop in Python?

A) **for** loops are used for iterating over sequences, while **while** loops execute a block of statements as long as a condition is true.
B) **for** loops are used for arithmetic calculations, while **while** loops are used for logical operations.
C) **for** loops can only be used with lists, while **while** loops work with all data types.
D) There is no difference; they are interchangeable and can be used interchangeably.

5. What is the difference between a for loop and a while loop in Python?

A) **for** loops are used for iterating over sequences, while **while** loops execute a block of statements as long as a condition is true.
B) **for** loops are used for arithmetic calculations, while **while** loops are used for logical operations.
C) **for** loops can only be used with lists, while **while** loops work with all data types.
D) There is no difference; they are interchangeable and can be used interchangeably.

OBJECTIVES

Python Conditional and Loop Statements

01 DISCUSS THE PYTHON CONDITIONS AND LOOPS;

02 EXECUTE PROGRAMS USING PYTHON CONDITIONS AND LOOPS; AND

03 GENERATE PYTHON PROGRAMS USING CONDITIONS AND LOOPS.

1.PYTHON CONDITION AND IF STATEMENT

Conditional Statements (if, else, and elif) are fundamental programming constructs that allow you to control the flow of your program based on conditions that you specify. They provide a way to make decisions in your program and execute different code based on those decisions.

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

TYPES OF CONDITIONAL STATEMENT

IF STATEMENT

IF..ELSE STATEMENT

IF-ELIF STATEMENT

NESTED IF

if Statement

- These conditions can be used in several ways, commonly in "**if** statements" and loops.
- The **if** statement is the simplest decision-making statement. It is used to decide whether a statement or block of statements will be executed.
- An "**if** statement" is written by using the if keyword

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

if-elif Statement

- *The **if-elif statement** is a shortcut of the **if..else chain.** While using the **if-elif** statement at the end else block is added which is performed if none of the above **if-elif** statement is true.*
- ***Elif** is short for **"else if"** and is used to introduce an additional condition to check if the previous **if** condition was not met.*
- *It comes after an initial if statement and allows you to specify another condition that is evaluated only if the preceding if condition is False.*
- *You can have multiple **elif** blocks in a sequence, and the first one that evaluates to True will be executed.*

if-elif Statement

Example

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

if..else Statement

- IN A CONDITIONAL **IF** STATEMENT THE ADDITIONAL BLOCK OF CODE IS MERGED AS **ELSE** STATEMENT WHICH IS PERFORMED WHEN **IF** CONDITION IS FALSE.
- **else** is used in conjunction with an initial **if** statement and provides a fallback or default code block to execute when the preceding **if** and **elif** conditions are all False.

if..else Statement

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Nested If

- YOU CAN HAVE **IF** STATEMENTS INSIDE **IF** STATEMENTS, THIS IS CALLED NESTED **IF** STATEMENTS.
- We can have an **if…elif…else** statement inside another **if…elif…else** statement. This is called nesting in computer programming. Any number of these statements can be nested inside one another. Indentation is the only way to figure out the level of nesting. This can get confusing, so it must be avoided if we can.

Nested If

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

2. PYTHON “FOR” LOOPS

A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the keyword in other programming languages and works more like an iterator method as found in other object-orientated programming languages.

With the for loop, we can execute a set of statements, once for each item in a list, tuple, set, etc.

EXAMPLE OF “FOR LOOP”

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

LOOPING THROUGH A STRING

Even strings are iterable objects, they contain a sequence of characters:

```
for x in "banana":
  print(x)
```

CONTROL STATEMENT

break Statement

With the **break** statement, you can stop the loop before it has looped through all the items:
Terminates the loop statement and transfers execution to the statement immediately following the loop.

continue Statement

With the **continue** statement we can stop the current iteration of the loop, and continue with the next:
Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

pass Statement

for loops cannot be empty, but if you for some reason have a **for** loop with no content, put in the **pass** statement to avoid getting an error.
The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute

2. WHILE LOOP

- used to execute a block of statements repeatedly until a given condition is satisfied. And when the condition becomes false, the line immediately after the loop in the program is executed.

- With the while loop we can execute a set of statements as long as a condition is true.

WHILE LOOP

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

3. NESTED LOOPS

A nested loop is a loop inside a loop. The "inner loop" will be executed one time for each iteration of the "outer loop":

NESTED LOOPS

Print each adjective for every fruit:

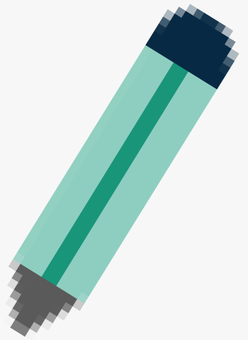
```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

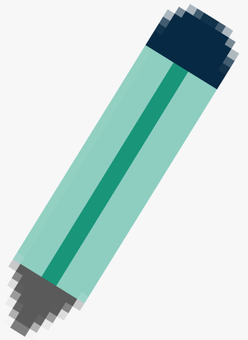
PYTHON CONDITIONAL AND LOOP STATEMENTS

Takeaways?

QUIZ TIME



MULTIPLE CHOICE: CHOOSE THE BEST ANSWER.



1. What is the primary purpose of using loops in programming?

A. To create graphical user interfaces.
B. To perform mathematical calculations.
C. To automate repetitive tasks.
D. To display text on the screen.

2. What is a "for" loop in Python used for??

A. Displaying data on the screen.
B. Iterating over a sequence.
C. Defining functions.
D. Storing values in variables.

3. How can you stop a loop before it has looped through all the items using the break statement?

A. By deleting the loop.
B. By using the stop keyword.
C. By writing a stop statement.
D. By using the break statement.

4. What is the purpose of the continue statement in a loop?

A. To skip the loop entirely.

B. To stop the loop.

C. To skip the remainder of the loop's body and retest the loop condition.

D. To add an additional condition to the loop

5. Which logical condition can be used to check if two variables a and b are equal in Python?

A. a is b

B. a equals b

C. a == b

D. a <> b

6. What is the purpose of the continue statement in a loop in Python?

A. It terminates the loop.
B. It skips the current iteration of the loop and proceeds with the next iteration.
C. It executes the loop's body repeatedly.
D. It is used to introduce additional conditions.

7. In Python, which of the following statements is true about elif blocks in an if-elif-else chain?

A. Only one elif block can be used.
B. elif blocks are optional and can be omitted.
C. elif blocks are executed unconditionally.
D. Multiple elif blocks can be used, and the first one that evaluates to True is executed.

8. What happens if you omit the else block in an if-else statement in Python?

A. It will result in a syntax error.
B. The program will run without any issues.
C. The if block will be executed unconditionally.
D. The else block will be automatically generated by Python.

9. Which of the following logical conditions is correct in Python to check if a number is not equal to 10?

A. num == 10
B. num != 10
C. num > 10
D. num < 10

10. What is the key difference between a for loop and a while loop in Python?

A. for loops are used for definite iteration, while while loops are used for indefinite iteration.
B. for loops can only be used with numbers, while while loops work with strings.
C. for loops can only iterate over lists, while while loops can iterate over any data type.
D. for loops and while loops serve the same purpose and can be used interchangeably.

STOP ANSWERING

PYTHON CONDITIONAL AND LOOP

Statements

Ready to code?

TEST YOUR SKILLS:

Create a program based on the requirements given in the problem.

Problem A. Write a Python program to count the number of letters, digits, and other characters in a user-provided phrase. The program will need to show the results to the user.

Example: SAsags213S$(#FS_

Number of letters: 9
Number of digits: 3
Number of symbols: 4
Total number of characters: 16

Problem B. Write a Python program that takes a numerical grade as input and provides a message based on the grade range.

1.0 – Excellent
1.1 – 1.25 – Very Good
1.26 – 1.50 – Good
1.51 – 1.75 – Fair
1.76 – 2.0 – Passed
Below 2.0 – Failed
If not in the list – Invalid Grade

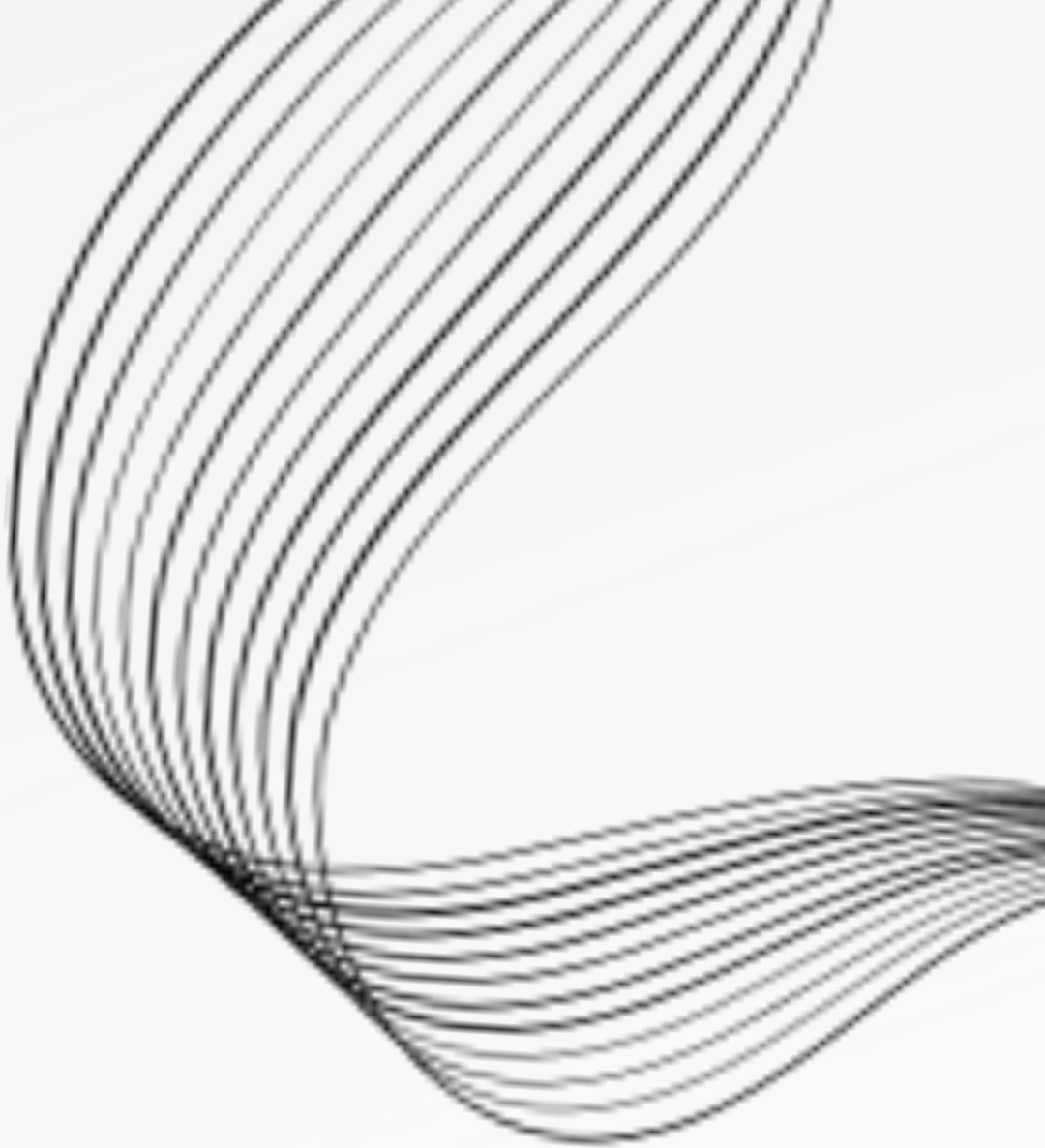
PYTHON CONDITIONAL AND LOOP Statements

ASSIGNMENT

ASSIGNMENT

Create a number guessing game where the computer selects a random
number between 1 and 100, and the player has to guess the number.
The game provides feedback to the player after each guess,
indicating whether the guess is too high, too low, or correct. The
game should also handle the case when the player decides to quit
by entering 0.

THANK'S FOR LISTENING



REFERENCES

- https://www.w3schools.com/python/python_conditions.asp
- https:/www.geeksforgeeks.org/python3-if-if-else-nested-if-if-elif-statements/
- https:/www.tutorialspoint.com/python/python_loops.htmhttps:/www.w3schools.com/python/python_for_loops.asp
- https:/www.freecodecamp.org/news/how-to-use-conditional-statements-if-else-elif-in-python/#:~:text=Conditional%20statements%20(if%2C%20else%2C,code%20based%20on%20those%20decisions.