100 DAYS 100 PYTHON PROBLEMS

[Day 3: Control structures (if statements, loops)]

Control structures are essential elements in programming that allow you to control the flow of your code. They enable you to make decisions, repeat actions, and create complex algorithms. In this context, we will cover two fundamental control structures: if statements and loops.

if statement :-

- An if statement is used to make decisions in your code.
- It allows you to execute different code blocks based on conditions.
- Basic syntax of 'if' statement is –

```
if condition:
    # code to be executed if the condition is true
else:
    # code to be executed if the condition is false
```

➢ INDENTATION

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

Error full Code :

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

Error less Code :

```
a = 33
b = 200

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

100 DAYS 100 PYTHON PROBLEMS

[Day 3: Control structures (if statements, loops)]

elif statement :-

- The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

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

else statement :-

- The else keyword catches anything which isn't caught by the preceding conditions.

```
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")
```

Short hand :-

- If you have only one statement to execute, you can put it on the same line as the if statement.

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

```
a = 2
b = 330
print("A") if a > b else print("B")
```

100 DAYS 100 PYTHON PROBLEMS

[Day 3: Control structures (if statements, loops)]

And/ Or/ Not :-

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

Nested if :-

- You can have if statements inside if statements, this is called nested if statements.

```
x = 41

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

100 DAYS 100 PYTHON PROBLEMS

[Day 3: Control structures (if statements, loops)]

The pass statement :-

- if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
a = 33
b = 200

if b > a:
  pass
```

Loops in python :-

- Loops allow you to execute a block of code repeatedly. There are two primary types of loops:

a) FOR LOOP =

- For loops are used when you know how many times you want to repeat a certain block of code.
- They are typically used with a range of values or elements in an iterable (e.g., a list).

```
for variable in iterable:
    # Code to execute for each iteration
```

b) WHILE LOOP =

- While loops are used when you want to repeat a block of code until a certain condition is met.

```
while condition:
    # Code to execute while the condition is true
```

100 DAYS 100 PYTHON PROBLEMS

[Day 3: Control structures (if statements, loops)]

PRACTICE QUESTIONS :-

1. Write a Python program that checks if a given number is even or odd using an **if** statement.
2. Create a Python program that prints the numbers from 1 to 10 using a **for** loop.
3. Write a Python program that calculates the sum of all the even numbers from 1 to 50 using a **for** loop.
4. Develop a Python program that asks the user to input a number and then prints the multiplication table for that number using a **for** loop.
5. Create a Python program that counts down from 10 to 1 using a **while** loop.
6. Write a Python program that simulates a simple guessing game. The program generates a random number, and the user has to guess the number. It should provide feedback to the user (e.g., "Too low" or "Too high") and continue until the user guesses the correct number.
7. Create a Python program that calculates the factorial of a given number using a **for** loop.
8. Write a Python program that checks if a given string is a palindrome (reads the same backward as forward) using an **if** statement.
9. Develop a Python program that prints all the prime numbers between 1 and 100 using nested **for** loops.