Conditional Statements (31-40)

31. What are the conditional statements in Python?

→ Conditional statements in Python are used to make decision in your code - they allow your program to execute different action based on conditions.

<u>key conditional statements:</u>

→ if
→ if-else
→ if - elif - else.

32. What is the syntax of an if statement in Python?

<u>Syntax:</u>

```
if condition:
```

<u>Key points:</u>

→ The if keyword is followed by a condition
→ Ends with colon (:)
→ The code block under the if must be indented

33. What is the difference b/w if and if-else?

if	if-else.
* Executes a block only if the condition is true.	* Executes one block if true, another if false.
* When only care about one condition	* When you want two possible outcomes.

age = 20
if age >= 18 :
	age = 15
	if age >= 18:

```
print ("You are an adult.")
```

if age>=18. is True, it prints the message.
If False, it does nothing.

```
    print ("You are an adult.")
else:
    print ("You are a minor:")
```

Executes one block if condition is true
Executes the else block if condition is false.

34. What is the use of elif in Python?

→ The elif keyword stands for else-if. It is used to check multiple conditions after the initial if statement.

35. Can you use multiple elif blocks in a condition?

Yes, you can use multiple elif blocks in a condition in Python. This is useful when you want to check several different conditions one after another

Example:

```
score = 82
if score >= 90:
    print ("Grade : A")
elif score >= 80:
    print ("Grade : B")
elif score >= 70:
    print ("Grade : C")
elif score >= 60:
    print ("Grade : D")
else:
    print ("Grade : F")
```

```
// output:
    Grade : B
```

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

→ It depends on whether you include an else block.

With else:

If none of the if or elif conditions are true, the else block will run.

```
x = 5
if x > 10:
    print("Greater than 10")
elif x == 10:
    print("Equal to 10")
else:
    print("Less than 10")
```

// output

```
Less than 10.
```

Without else:

If there's no else and none of the conditions are true, nothing happens. the program just skips the block.

```
x = 5
if x > 10:
    print("Greater than 10")
elif x == 10:
    print("Equal to 10")
```

// output

(Nothing is printed)

37. Can we use if inside another if? Explain it?

→ Yes. you can use an if inside another if - this is called a nested if statement.

→ It allows you to test more than one condition in a hierarchial or step-by-step manner

Example:

```
age = 20
has_id = True
if age >= 18:
    if has_id:
        print ("Entry allowed:")
                                      // output:
    else:
        print ("Id required:")              Entry allowed.
else:
    print (" You must be 18 or older.")
```

38. How is an identation important in writing conditionals in Python?

→ In Python, indentation is not just style - it's syntax. It tells Python which code belongs to which block, especially in conditionals like if, else and elif.

39. How do you check multiple conditions using and / or?

→ In Python, you can check multiple conditions in a single if statement using the logical operators and & or

and - All conditions must be True

```
age = 20
has_id = True
if age >= 18 and has_id:
    print("Entry allowed:")
```

// output:
Entry allowed.
Only if both conditions are True.

Or - At least one condition must be True.

```
age = 16
has_ticket = True.
if age >= 18 or has_ticket:
    print("You may enter.")
```

// output:
You may enter.
(Runs even though age is less than 18, because has_ticket is True)

40. What is the output of if " " or if 0 in Python? Why?

→ Both " " (an empty String) & 0 (zero) are considered false values in Python.

```
if " ":
  print ("This will print")
else:
  print ("This won't print")
```

// output:
This wont print
Because " " is False, so the if block is skipped and the else runs

```
if 0:
    print ("Number is non-zero")
else :
    print ("Number is zero:")
```

output: Number is Zero.

Because 0 is also False, the if block is skipped.

For Loop in Python (41-50)

41. What is for loop in Python and how is it used?

-> A for loop in Python is used to iterate over a sequence like a list, string, tuple or range and execute a block of code for each item in the sequence.

Syntax:

```
for variable in sequence:
```

42. What is the syntax of for Loop?

Syntax:

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

Components:

* for - the keyword that starts the loop.
* Variables - takes the value of each item in the sequence.
* in - used to link the variable and the sequence.
* sequence - a list, string, tuple, dictionary or range()
* : - colon ends the for statement.
* Indented block - code that runs on each loop.

43. How does the range() function work with loops?

→ The range() function is commonly used in for loops to generate a sequence of numbers for looping a specific number of times.

Basic syntax of range():

* range (stop)
* range (start, stop)
* range (start, stop, step)

44. Can you loop over strings and lists using for?

→ Yes, you can loop over strings and lists using a for loop in Python.

→ Both strings and lists are iterable, meaning they contain elements you can loop through one by one.

Looping over a string:

```
text = "hello"
for char in text:
    print (char)
```

```
// output
h
e
l
l
o
```

Each character in a string is accessed one at a time.

Looping over a list:

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

```
// output
apple
banana
cherry.
```

Each item in the list is processed in order.

45. What is the use of break and continue inside a loop?

-> In Python, break and continue are loop control Statements that let you change how a loop behaves.

* break. Stops the loop completely.
Exits the loop immediately, even if the condition hasn't finished.
Used when a certain condition is met, you want to exit early!

```
for num in range (1,6):                Output
    if num == 3:                       1
        break                          2
    print(num)                         Stop when num == 3
```

* continue - Skips the current iteration.

-> Skips the rest of the loop only for that iteration
-> Then moves on to the next item.

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

Output:

```
1
2
4
5
```

Skips printing 3, but continues the loop.

46. How do you print only even numbers b/w 1 & 20 using a loop?

→ You can use a for loop with an if condition to print only even numbers.

Method 1: Using if & continue

```
for num in range(1, 21):
    if num % 2 != 0:
        continue
    print(num)
```

Output:

```
    2
    4
    6
    8
    10
```

Method 2: Using range() with step.

```
for num in range(2, 11, 2):
    print(num)
```

Output:

```
    2
    4
    6
    8
    10
```

47. What is the use of else with a for loop?

→ In Python, you can attach an else block to a for loop.

→ The else block runs only if the for loop completes normally (ie. no break is hit)

→ If the loop is interrupted by a break, the else block is skipped.

Example:
```
for i in range(5):
    print(i)
else:
    print("Loop finished successfully.")
```

output:
```
0
1
2
3
4
Loop finished successfully.
```

Example 2: Loop is broken early

```
for i in range(5):
    if i == 3:
        break
    print(i)
else:
    print("Loop finished successfully")
```

Output:
```
0
1
2
```

else does not run because break was used.

48. What does enumerate() do in a for loop?

→ The enumerate () function in Python is used to loop over a sequence (like a list or string) & get both the index and the item at the same time.

Syntax: for index , item in enumerate (sequence):

Without enumerate ():

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

Output:

```
apple
banana
Cherry.
```

You get only items, no index.

With enumerate ():

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

Output:

```
0 apple
1 banana
2 cherry.
```

You get both index and item in each loop.

49. What is the nested loop? Provide an example.

-> A nested loop is a loop inside another loop.

-> In Python, you can nest any type of loop (for or while) within another loop.

Syntax:

```
for i in outer_sequence :
        for j in inner_sequence:
```

Example: Print a number pattern.

```
for i in range (1,4):
        for j in range (1, i+1):
            print (j, end=" ")
        print ()
```

Output:

```
1
1 2
1 2 3
```

50. Can we use for loops with dictionaries? If yes, how?

-> Dictionaries are collection of key value pairs, and Python allows you to loop through them in several useful ways.

1. Loop through keys (Default Behavior):

```
student = {"name": "Alice", "age": 18, "grade": "A"}
```

```
for key in student:
    print(key)
```

Output:

```
name
age
grade.
```

2. Loop through values:

```
for value in student.values():
    print(value)
```

Output:

```
Alice
18
A
```

3. Loop through key-value pairs:

```
for key, value in student.items():
    print(key, "->", value)
```

Output:

```
name -> Alice
age -> 18
grade -> A.
```