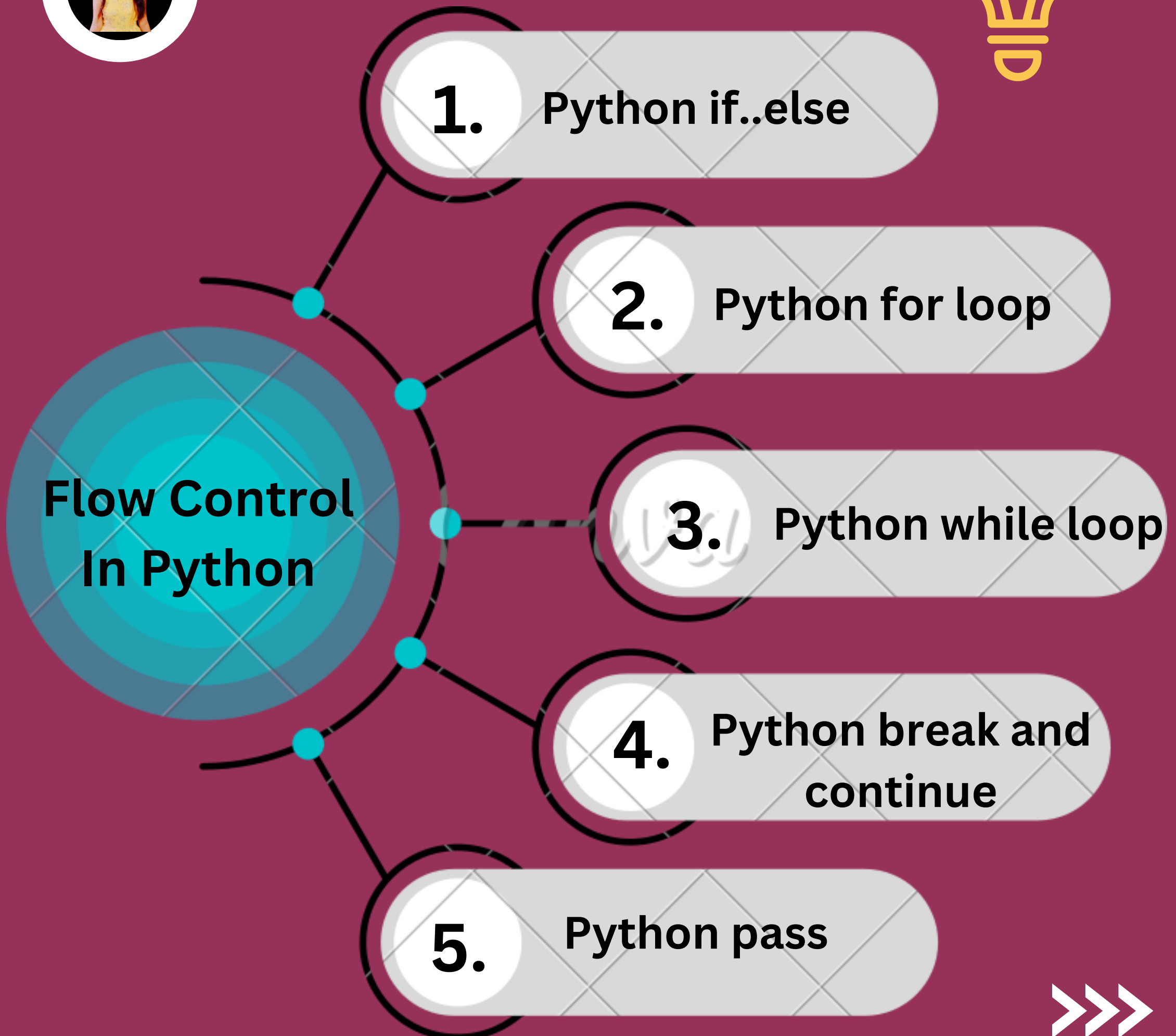




# Concepts on your tips -



# Python if...else Statement

There are basically three forms of the if...else statement.

1. if statement
2. if-else statement
3. if-elif-else statement

## if Statement in Python

- In control statements, The if statement is the simplest form. It takes a condition and evaluates to either True or False.
- If the condition is True, then the True block of code will be executed, and if the condition is False, then the block of code is skipped, and The controller moves to the next line
- **Syntax:-**      **if** condition:  
                    # body of if statement



## If – else statement

- The if-else statement checks the condition and executes the if block of code when the condition is True, and if the condition is False, it will execute the else block of code.

- **Syntax:-**  
**if** condition:  
    statement 1  
**else:**  
    statement 2

## If – else statement

- In Python, the if-elif-else condition statement has an elif blocks to chain multiple conditions one after another. This is useful when you need to check multiple conditions.

- **Syntax:-**  
**if** condition1:  
    # code block 1  
**elif** condition2:  
    # code block 2  
**else:**  
    # code block 3

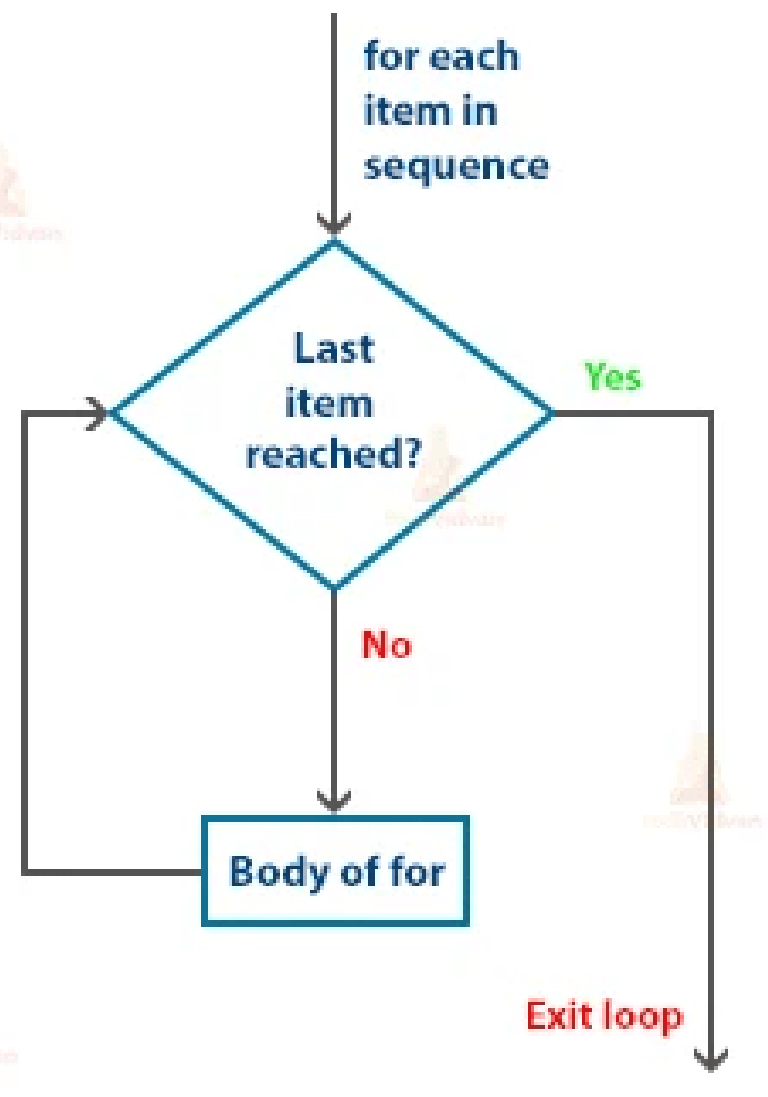


# Python for Loop

- Using for loop, we can iterate any sequence or iterable variable. The sequence can be string, list, dictionary, set, or tuple.
- **Syntax:-**  
for val in sequence:  
    # statement(s)

```
Python Tutorial.py
1 numbers = [10, 23, 43, 54, 65, 64, 87, 43, 21, 11]
2
3 for number in numbers: #2
4     print(number)      #3
   for number in numbers
Run: new_visual x Python Tutorial x
10
23
43
54
65
64
87
43
21
```

## Operation of for Loop



# Python while Loop

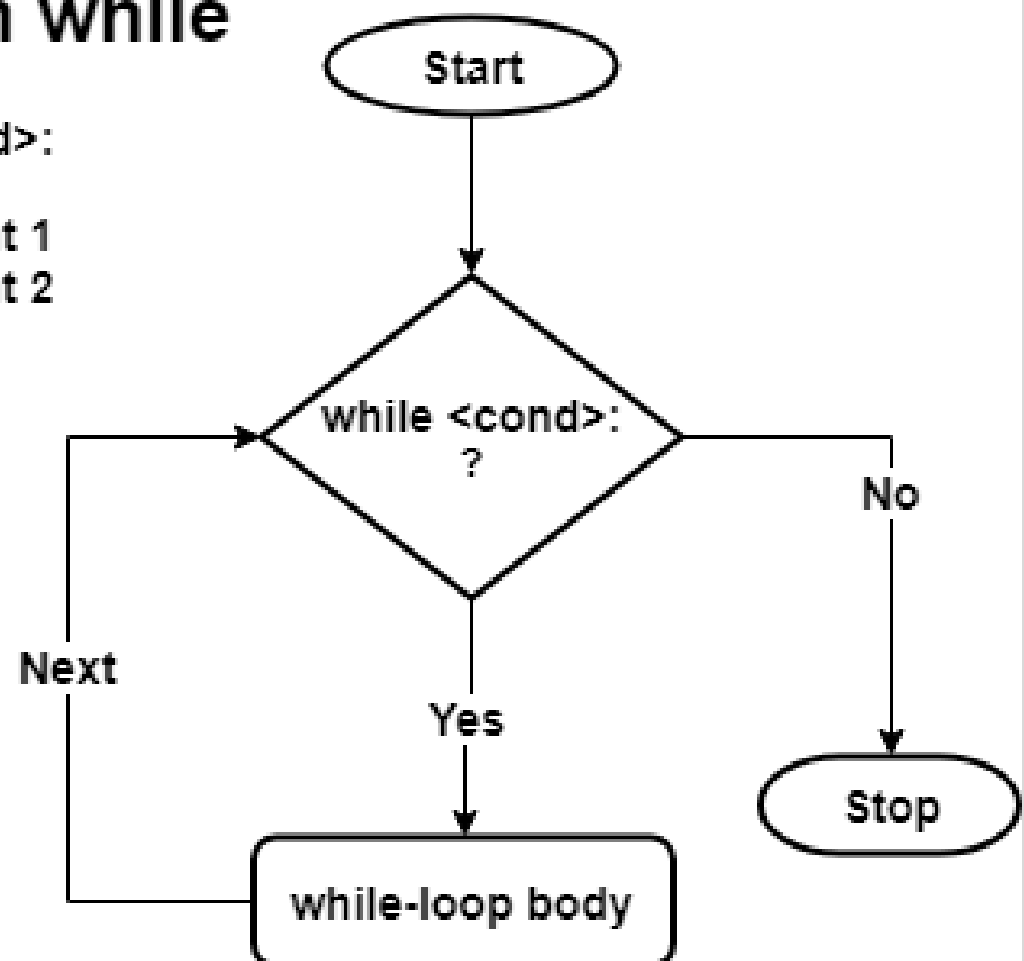
- In Python, The while loop statement repeatedly executes a code block while a particular condition is true.
- **Syntax:-**  
while condition:  
    # body of while loop

```
i = 5
while i > 0:
    i = i - 1
    if i == 2:
        continue
    print("inside loop", i)
else:
    print('Inside else')
```

```
inside loop 4
inside loop 3
inside loop 1
inside loop 0
Inside else
```

## Python while

```
while <cond>:
    statement 1
    statement 2
    ...
```



# Python break & continue

- The break statement is used to terminate the loop immediately when it is encountered.
- The continue statement is used to skip the current iteration of the loop and the control flow of the program goes to the next iteration

**for** var in sequence :

....  
statement(s)

....  
if (test\_expression)

break **true**

←  
*out of the loop*

**while** (test\_expression1) :

....  
statement(s)

....  
if (test\_expression2) :

break **true**

←  
*out of the loop*



## break

## continue



top of the loop

**for** test\_expression

statement1

....

if (condition)

continue **true**

....

statement2

top of the loop

**while** (test condition)

....  
statement1

....

if (condition)

continue **true**

....

statement2



# Python pass

- In Python programming, the pass statement is a null statement which can be used as a placeholder for future code.
- A pass statement is a Python null statement. When the interpreter finds a pass statement in the program, it returns no operation. Nothing happens when the pass statement is executed.
- It is useful in a situation where we are implementing new methods or also in exception handling. It plays a role like a placeholder.



The screenshot shows a Python script named `pass.py` in a text editor. The script defines a string `string1 = "Stechies"` and uses a `for` loop to iterate over its characters. Inside the loop, an `if` statement checks if the character is `'e'`. If it is, a `pass` statement is used as a placeholder. Otherwise, the character is printed. A red box highlights the `pass` statement, and a blue callout points to it with the text "pass statement". Below the script, a command prompt window shows the execution of `python pass.py`, resulting in the output: `Value: S`, `Value: t`, `Value: c`, `Value: h`, `Value: i`, and `Value: s`. A red box highlights this output, and a blue callout points to it with the text "Required Output".

```
pass.py - D:/python/pass.py (3.7.4)
File Edit Format Run Options Window Help

string1 = "Stechies"

# Use of pass statement
for value in string1:
    if value == 'e':
        pass
    else:
        print("Value: ",value)

C:\Windows\system32\cmd.exe

D:\python>python pass.py
Value: S
Value: t
Value: c
Value: h
Value: i
Value: s

D:\python>
```





If you really like my content  
please don't forget to like ,  
comment and share

*Thank You* 😊



Vaishnavi Pandey