© Shehab Abdel-Salam

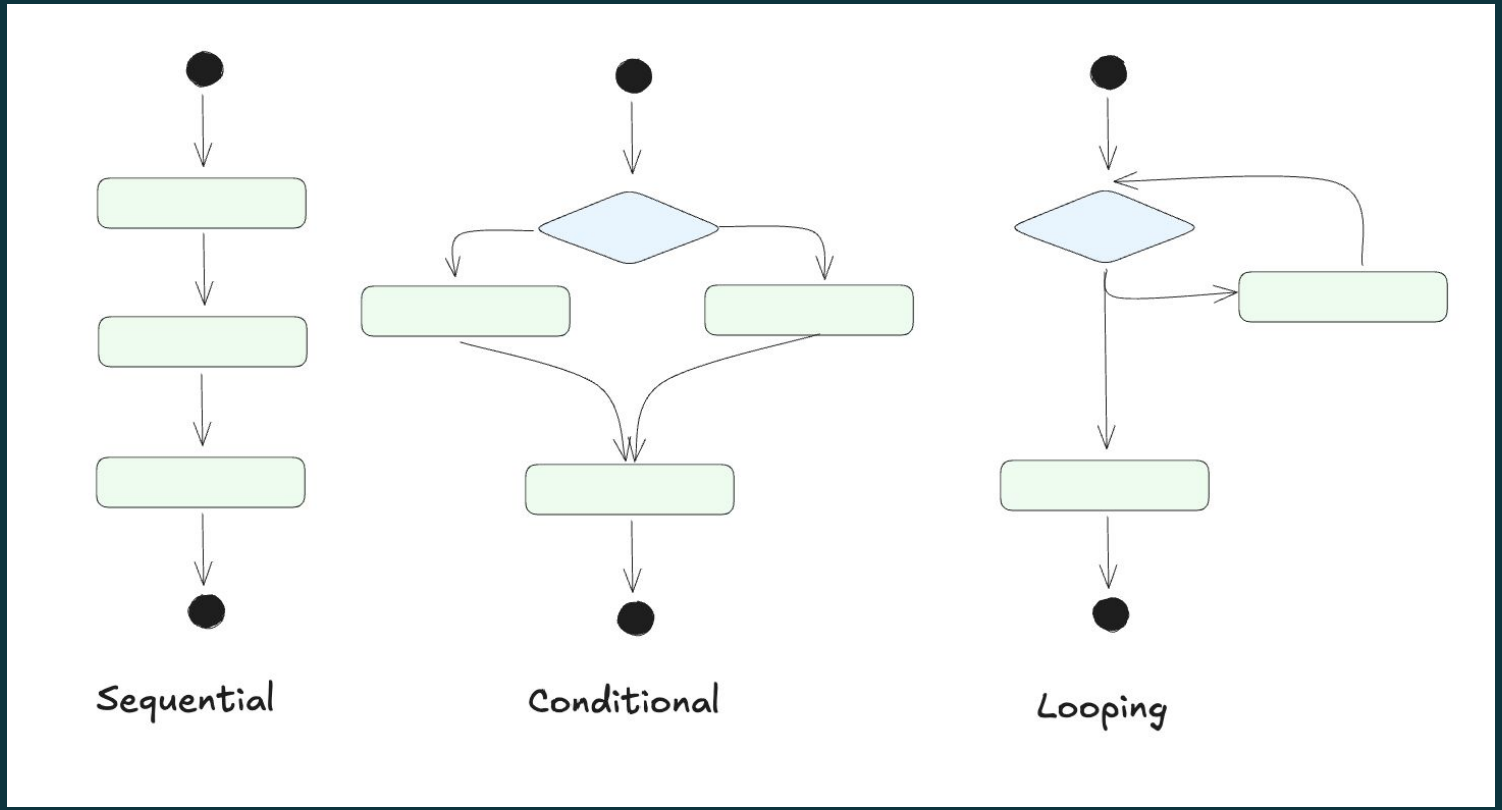
04 - Control Flow

© Shehab Abdel-Salam

What is a control flow?

© Shehab Abdel-Salam

What is a control flow?



© Shehab Abdel-Salam

What is a control flow?

Control flow refers to the order in which individual statements or code blocks are executed.

We can control the execution flow of a Python program using **conditional statements** or **loops**.

© Shehab Abdel-Salam

What is a control flow?

Control flow refers to the order in which individual statements or code blocks are executed.

We can control the execution flow of a Python program using **conditional statements** or **loops**.

By default, a Python program runs in sequential control flow.

We'll explore different ways of changing the flow:

1. If-else statements - (i.e. branching)
2. Looping - (i.e. repetition)

© Shehab Abdel-Salam

Conditional statements

© Shehab Abdel-Salam

Conditional statements

Conditional statements help us to create different branches.

Conditional statements use **`if`**, **`elif`**, and **`else`** to execute different blocks of code based on the evaluation of Boolean expressions

© Shehab Abdel-Salam

Conditional statements

Conditional statements help us to create different branches.

Conditional statements use **`if`**, **`elif`**, and **`else`** to execute different blocks of code based on the evaluation of Boolean expressions

```
# Branching can happen in form of:

# 1. If
# 2. If-else
# 3. If-elif-else
# 4. Nested If
```

© Shehab Abdel-Salam

if statement

Executes a block of code if a specified condition is true.

```
grade = 85

if grade >= 90:
    print("You got an A.")
```

© Shehab Abdel-Salam

if-else statement

Executes one block of code if a condition is true, and another block if it is false.

```
grade = 85

if grade >= 90:
    print("You got an A.")
else:
    print("You got below A.")
```

© Shehab Abdel-Salam

if-elif-else statement

Execute different code blocks for multiple conditions, with an else statement for when none of the conditions are true.

```
grade = 82

if grade >= 90:
    print("You got an A.")

elif grade >= 80:
    print("You got a B.")

else:
    print("You got below B.")
```

© Shehab Abdel-Salam

Nested if statement

Executes an **if statement** inside another **if statement** to evaluate multiple conditions.

```
grade = 90
extra_credit = True

if grade >= 90:
    if extra_credit:
        print("You got an A with extra credit.")
    else:
        print("You got an A.")
```

© Shehab Abdel-Salam

Structural Pattern matching

```
match instruction:
    case “start”:
        print(“Starting…”)
    case “quit”:
        print(“Quitting…”)
    case _:
        print(“Unknown”)
```

© Shehab Abdel-Salam

Structural Pattern matching

Structural pattern matching introduces the **`match`** / **`case`** statement and the pattern syntax to Python.

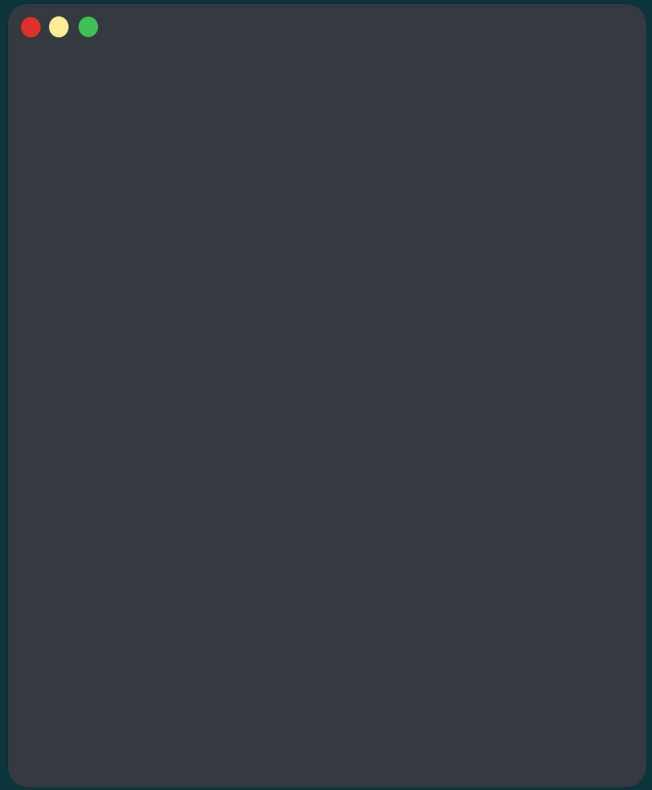
It takes an object, tests the object against one or more match patterns, and executes the code block where a match is found.

New feature available from **Python 3.10**

```
match instruction:
    case “start”:
        print(“Starting…”)
    case “quit”:
        print(“Quitting…”)
    case _:
        print(“Unknown”)
```

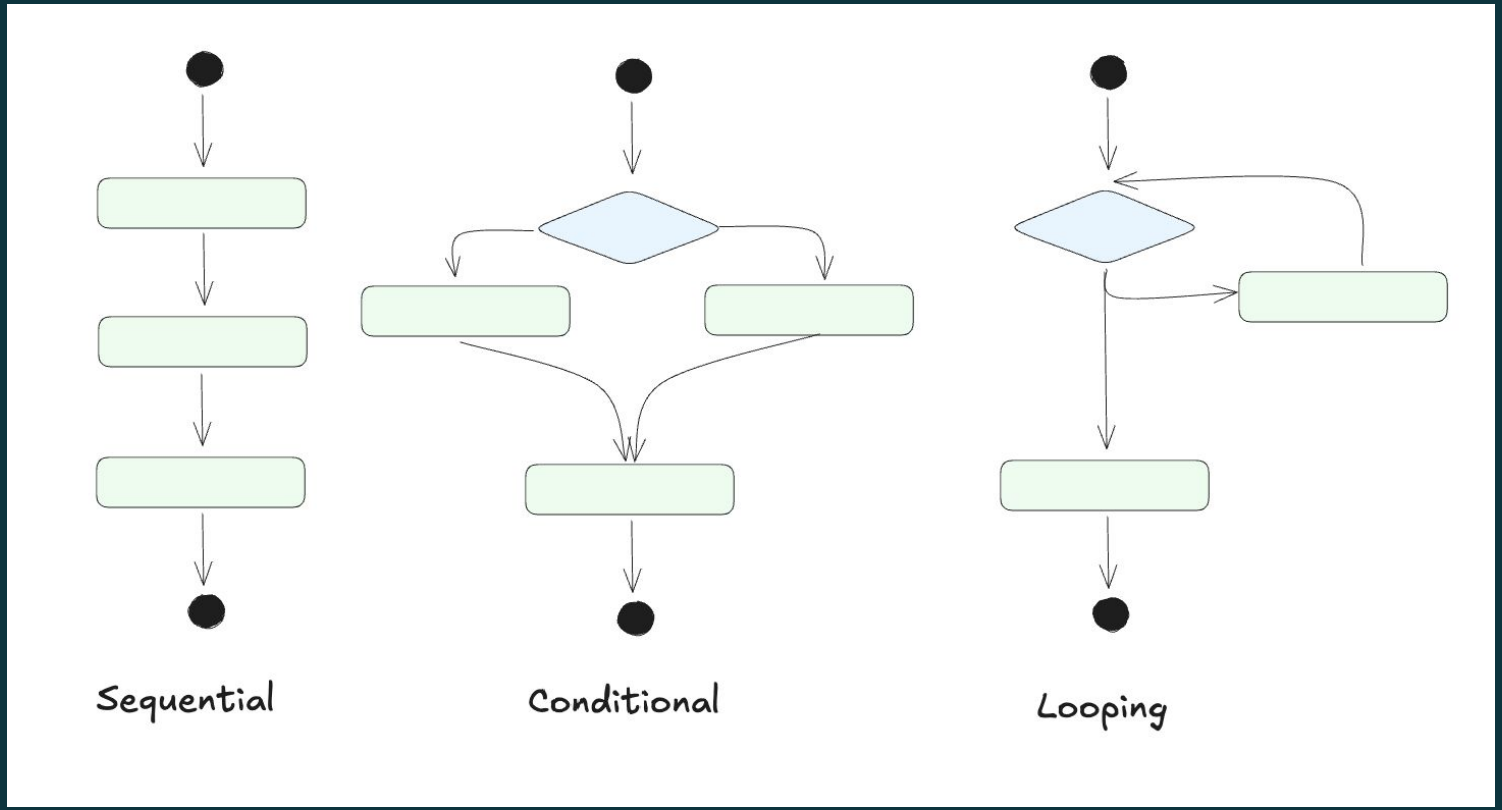
© Shehab Abdel-Salam

Loops



© Shehab Abdel-Salam

Loops



© Shehab Abdel-Salam

Loops

Loops is a way of **repeating** the execution of a specific code block.

```
# Print ‘Hello world’ 5 times.
for i in range(5):
    print(“Hello world”)
```

© Shehab Abdel-Salam

Loops

Loops is a way of **repeating** the execution of a specific code block.

```
# Print ‘Hello world’ 5 times.
for i in range(5):
   print(“Hello world”)
```

In Python, we have two types of loops:

- for-loop
- while-loop

© Shehab Abdel-Salam

Loops (for-loop)

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

```
# For-loop example 1
for i in range(5):
   print(“Hello world”)

# For-loop example 2
fruits = [“apple”, “orange”]
for fruit in fruits:
   print(fruit)

# For-loop example 3
for index, fruit in enumerate(fruits):
   print(index, fruit)
```

© Shehab Abdel-Salam

Loops (for-loop)

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

You can use for-loops for:

1. Iterating using `range()`
2. Iterating over a list, dictionary, etc.
3. Iterating with an Index

```
# For-loop example 1
for i in range(5):
   print(“Hello world”)

# For-loop example 2
fruits = [“apple”, “orange”]
for fruit in fruits:
   print(fruit)

# For-loop example 3
for index, fruit in enumerate(fruits):
   print(index, fruit)
```

© Shehab Abdel-Salam

Loops (while)

The while loop is used to execute a block of code as long as the condition is true.

```
counter = 1

while counter <= 5:
    print(counter)
    counter += 1
```

© Shehab Abdel-Salam

Loops (while) cont’d

Use `break` to exit the loop when a certain condition is met.

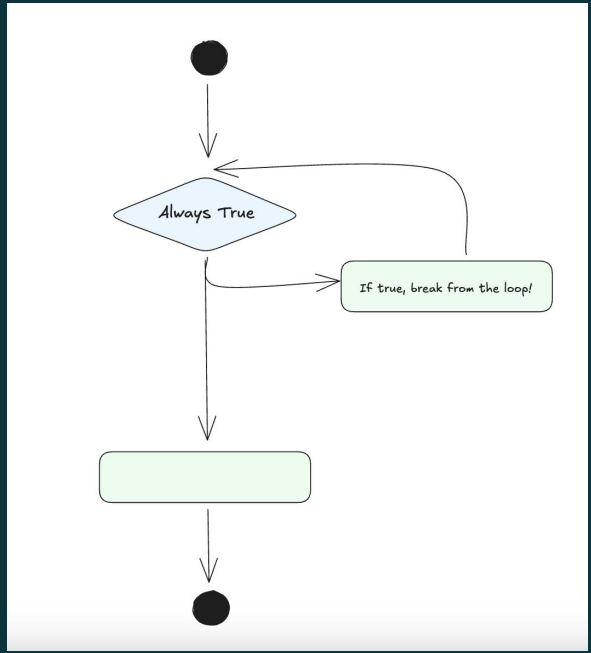
```
counter = 1

while True:
    print(counter)
    if counter == 5:
      break
    counter += 1
```

© Shehab Abdel-Salam

Loops (while) cont'd

Use `break` to exit the loop when a certain condition is met.




```
counter = 1

while True:
    print(counter)
    if counter == 5:
      break
    counter += 1
```

© Shehab Abdel-Salam

Loops (while) cont’d

Use `continue` to skip the current iteration and continue with the next one.

```
counter = 0

while counter < 10:
    counter += 1
    if counter % 2 == 0:
        continue # skip
    print(counter)
```

© Shehab Abdel-Salam

Summary

Other statements used in loops:

- **`break`** - for exiting the current execution loop.
  - Useful for stopping the current loop based on some condition.
  - Exists the loop prematurely.
- **`continue`** - for skipping the current iteration and go directly into the next iteration of the loop.
  - Skips the current iteration and continues with the next iteration.
- **`pass`** - used for doing nothing.
  - Useful as a placeholder for writing code, or for suggestion no action to be taken based on a valuated condition.
  - Put in the pass statement to avoid getting an error.

© Shehab Abdel-Salam

Conclusion

- Control flows
- Conditional statements
  - if, if-else, if-elif-else, and nested if statements.
- Structural pattern matching
- Loops
  - for-loop and while-loop
- Other statements used in repetition
  - break
  - continue
  - pass