

Lecture 04: Control Flow in Python

February 16, 2023



1 Control Flow in Python

Ahmed Moustafa

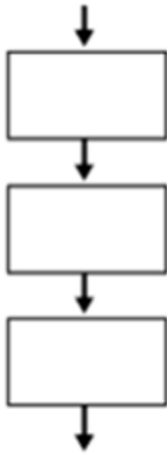
2 Please scan...



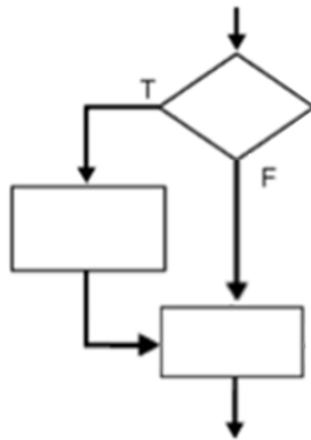
3 Definition of Control Flow

- Control flow is the order in which statements and instructions are executed in a program
- Control flow can be affected by decision-making statements, loops, and function calls.

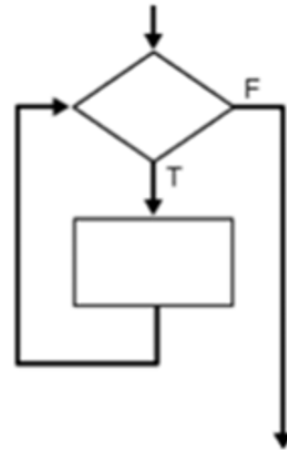
Sequence



Selection



Iteration



4 Code blocks and indentation

- Code blocks are a group of statements that are executed together.
- In Python, code blocks are defined by their indentation level.

```
[4]: %%script echo
if condition:
    # This code is indented, so it's part of the if block
    print('The condition is true.')

# This code is not indented, so it's not part of the if block
print('This code runs regardless of the condition.')
```

5 If-else statements:

- If-else statements are used to execute code based on a condition.
- If the condition is true, the code within the if block is executed.
- If the condition is false, the code within the else block is executed.

Syntax:

```
[5]: %%script echo
if condition:
    # Code to execute if the condition is true
else:
```

```
# Code to execute if the condition is false
```

6 If-else statements - Example

```
[6]: age = 25
     if age >= 18:
         print("You are an adult")
     else:
         print("You are a minor")
```

You are an adult

You can also use a shorthand notation for assigning a value to a variable based on a condition:

```
[7]: x = 10
     y = 20
     max_value = x if x > y else y
     print(max_value)
```

20

7 Boolean Operators

- Boolean operators are used to combine multiple conditions in an `if`-statement
- Python provides the operators `and`, `or`, and `not`

```
[8]: x = 5
     y = 10
     if x > 0 and y < 20:
         print("Both conditions are true")
     if x > 0 or y < 20:
         print("At least one condition is true")
     if not x == y:
         print("x is not equal to y")
```

Both conditions are true

At least one condition is true

x is not equal to y

8 Truth Table

<i>NOT</i>		<i>AND</i>			<i>OR</i>			<i>XOR</i>		
<i>x</i>	<i>x'</i>	<i>x</i>	<i>y</i>	<i>xy</i>	<i>x</i>	<i>y</i>	<i>x+y</i>	<i>x</i>	<i>y</i>	<i>x⊕y</i>
0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1	0	1	1
		1	0	0	1	0	1	1	0	1
		1	1	1	1	1	1	1	1	0

Source [Princeton's Introduction to CS: Boolean Logic](#)

9 Boolean Operators with Non-boolean Values

Boolean operators can also be used with non-boolean values. In Python, any non-zero or non-empty value is considered true, while zero or empty values are considered false:

```
[10]: name = ""
      if not name:
          print("The name is not set")
```

The name is not set

10 Nested If Statements

- Nested if statements allow for more complex conditions to be checked
- Multiple conditions can be checked using `if`, `elif`, and `else` statements inside of each other

Syntax:

```
[11]: %%script echo
      if condition1:
          if condition2:
              # Code to execute if both conditions are true
          else:
              # Code to execute if condition1 is true and condition2 is false
      else:
          # Code to execute if condition1 is false
```

11 Nested If Statements - Example

```
[12]: grade = 85
      if grade >= 90:
          print("You got an A")
      elif grade >= 80:
          print("You got a B")
      elif grade >= 70:
          print("You got a C")
      else:
          print("You failed the course")
```

You got a B

12 Loops: for

for loops are used to iterate over a sequence of items. Syntax:

```
[13]: %%script echo
      for item in sequence:
          # Code to execute for each item in the sequence
```

```
[14]: fruits = ['apple', 'banana', 'cherry']
      for fruit in fruits:
          print(fruit)
```

apple
banana
cherry

13 Loops: while

- while loops are used to repeat a block of code while a certain condition is true.

Syntax:

```
[15]: %%script echo
      while condition:
          # Code to execute as long as the condition is true
```

14 Loops: while - Example

```
[16]: x = 0
      while x < 5:
          print(x)
          x += 1
```

```
0
1
2
3
4
```

15 continue and break Statements

- continue statements are used to skip to the next iteration of a loop.
- break statements are used to exit a loop early.

```
[17]: %%script echo
      for item in sequence:
          if condition:
              continue # Skip to the next iteration
          if other_condition:
              break # Exit the loop early
          # Code to execute for each item in the sequence
```

16 continue and break Statements - Example

```
[18]: fruits = ['apple', 'banana', 'cherry']
      for fruit in fruits:
          if fruit == 'banana':
              continue # Skip printing "banana"
          if fruit == 'cherry':
              break # Exit the loop
          print(fruit)
```

```
apple
```

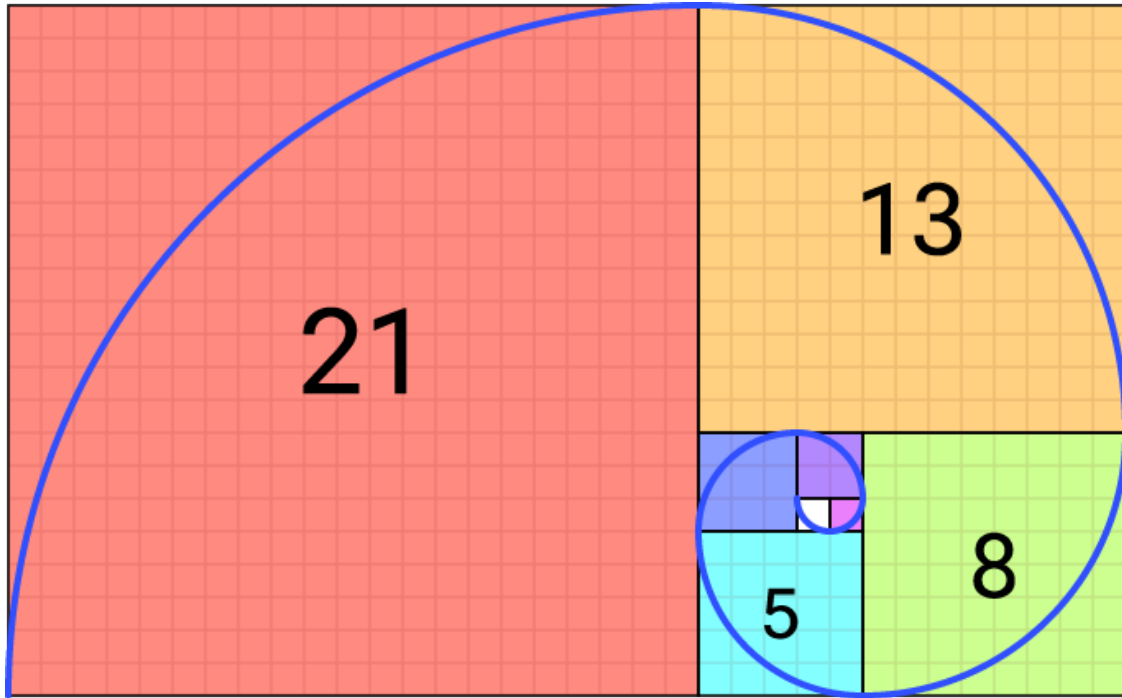
17 Exercise: Fibonacci Sequence

Write a program that generates the first 20 numbers in the Fibonacci sequence.

Hints: - The Fibonacci sequence is a sequence of numbers where each number is the sum of the two preceding numbers. - The first two numbers in the sequence are 0 and 1. - Use a for loop to generate the sequence.

Example output:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181



18 Exercise: Fibonacci Sequence (GitHub)



19 Summary

Python provides several control flow statements to manage the order of execution of code, including if-else statements, nested if statements, for loops, while loops, break statements, and continue statements. These statements allow data scientists to write more efficient and effective code.