



Fourth Industrial Summer School

Day 2

Python Programming II

Session Objectives

- ✓ Conditional statements
 - if
- ✓ Loops
 - while
 - for

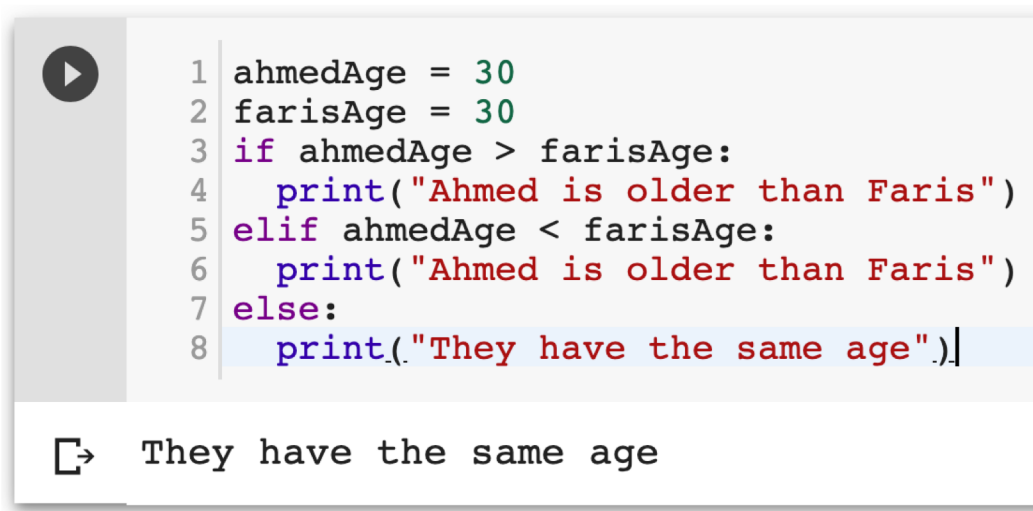


Part A

Conditional statements

Conditions

- Similar to other programming languages (if statements)
 - No switch support
- Python relies on indentation
 - Using whitespace to define code scope.
 - Other programming languages often use {}

A code editor window with a play button icon on the left. It contains Python code that checks the ages of Ahmed and Faris. The code is as follows:

```
1 ahmedAge = 30
2 farisAge = 30
3 if ahmedAge > farisAge:
4     print("Ahmed is older than Faris")
5 elif ahmedAge < farisAge:
6     print("Ahmed is older than Faris")
7 else:
8     print("They have the same age").|
```

The last line is highlighted in blue. Below the code editor, there is a terminal window showing the output: `They have the same age`.

Conditions

■ Logical Operators

| Operator | Meaning |
|----------|-----------------------|
| == | equals |
| != | Not equals |
| < | Less than |
| <= | Less than or equal |
| > | Greater than |
| >= | Greater than or equal |

Conditions

- **Compound** conditions uses the following logical operators to combine conditions
 - or
 - and

| AND | T | F |
|-----|---|---|
| T | T | F |
| F | F | F |

| OR | T | F |
|----|---|---|
| T | T | T |
| F | T | F |

```

1 num1 = 100
2 num2 = 200
3
4 if num1 > 0 and num2 > 0:
5     print("Both numbers are positive")
6 else:
7     print("Either numbers is negative")

```

Both numbers are positive

Hands on session

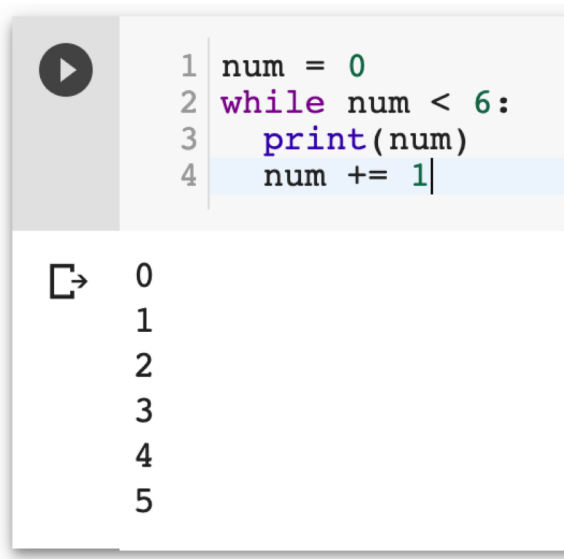
Problem Solving

Part B

Loops

Loops

- Two primitive loops in Python
 - `while`
 - Used to iterate over numbers.
 - `for`
 - Used to iterate over a list, a tuple, a dictionary, a set, or a string.



The image shows a Python code editor with a play button icon on the left. The code is as follows:

```
1 num = 0
2 while num < 6:
3     print(num)
4     num += 1
```

Below the code editor, the output of the program is displayed, showing the numbers 0 through 5, each on a new line, preceded by a small icon of a box with an arrow pointing right.

```
0
1
2
3
4
5
```

While

- Two statements used in loops
 - `break`
 - `continue`



```
1 num = 0
2 while num < 6:
3     if num == 4:
4         break
5     print(num)
6     num += 1
```



```
0
1
2
3
```



```
1 num = 0
2 while num < 6:
3     num += 1
4     if num == 4:
5         continue
6     print(num)
```



```
1
2
3
5
6
```

For

- Iterating over a list

```
1 cars = ["BMW", "Lexus", "Honda"]  
2 for x in cars:  
3     print(x)
```

↳
BMW
Lexus
Honda

- No indexing is required beforehand
- Iterating over a String

```
1 for x in "Honda":  
2     print(x)
```

↳
H
o
n
d
a

For

- We can use the break and continue statement

```
1 cars = ["BMW", "Lexus", "Honda"]  
2 for x in cars:  
3     if x == "BMW":  
4         continue  
5     print(x)
```

↳ Lexus
Honda

```
1 cars = ["BMW", "Lexus", "Honda"]  
2 for x in cars:  
3     if x == "Lexus":  
4         break  
5     print(x)
```

↳ BMW

For

- else after a for loop
 - Can be used to execute a set of code after loop



```
1 cars = ["BMW", "Lexus", "Honda"]
2 for x in cars:
3     print(x)
4 else:
5     print("Done printing")
```



```
BMW
Lexus
Honda
Done printing
```

For

- Nested loops
 - Loop inside a loop



```
1 cars = ["BMW", "Lexus", "Honda"]
2 colors = ["Red", "Blue"]
3 for x in cars:
4     for y in colors:
5         print(x,y)
```



```
BMW Red
BMW Blue
Lexus Red
Lexus Blue
Honda Red
Honda Blue
```

Hands on session

Problem Solving