University of Tripoli
Faculty of Information Technology

Department of Software Engineering

ITSE305 مواضيع مختارة
Python Programming
S2025

Lecture (4): Python Basics

Python If … Else

- Python supports the usual logical conditions from mathematics:
  - Equals: a == b
  - Not Equals: a != b
  - Less than: a < b
  - Less than or equal to: a <= b
  - Greater than: a > b
  - Greater than or equal to: a >= b
- An "if statement " is written by using the if keyword.
- The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".
- The else keyword catches anything which isn't caught by the preceding conditions.

Python If … Else

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

Python If … Else

- To combine conditional statements, use the logical operators: and, or and not.

```
a = 3300
b = 200
c = 500

if a > b and a > c:
  print("Both conditions are True")

if a > b or a > c:
  print("At least one of the conditions is True")

if not b > c:
  print("a is NOT greater than b")
```

```
Both conditions are True
At least one of the conditions is True
a is NOT greater than b
```

Python If … Else

- if statements inside if statements, this is called *nested* if statements.

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

```
Above ten,
and also above 20!
```

- if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error

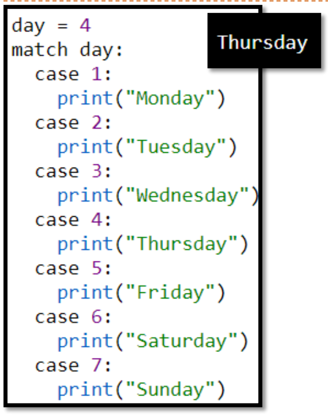
```
a = 33
b = 200

if b > a:
  pass
```

Python Match

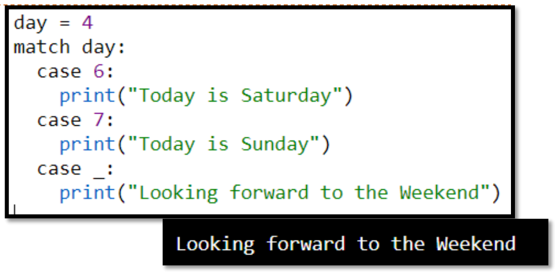
- Instead of writing **many** if..else statements, you can use the match statement.
- The match statement selects one of many code blocks to be executed
- how it works:
  - The match expression is evaluated once.
  - The value of the expression is compared with the values of each case.
  - If there is a match, the associated block of code is executed.
  - Use the underscore character _ as the last case value if you want a code block to execute when there are not other matches
  - Use the pipe character | as an or operator in the case evaluation to check for more than one value match in one case:

Python Match



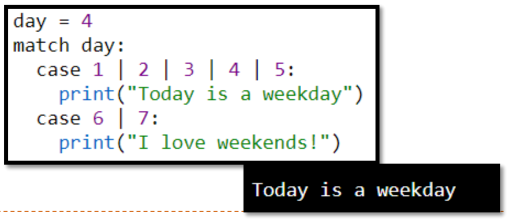
```
day = 4
match day:
  case 1:
    print("Monday")
  case 2:
    print("Tuesday")
  case 3:
    print("Wednesday")
  case 4:
    print("Thursday")
  case 5:
    print("Friday")
  case 6:
    print("Saturday")
  case 7:
    print("Sunday")
```

```
Thursday
```



```
day = 4
match day:
  case 6:
    print("Today is Saturday")
  case 7:
    print("Today is Sunday")
  case _:
    print("Looking forward to the Weekend")
```

```
Looking forward to the Weekend
```



```
day = 4
match day:
  case 1 | 2 | 3 | 4 | 5:
    print("Today is a weekday")
  case 6 | 7:
    print("I love weekends!")
```

```
Today is a weekday
```

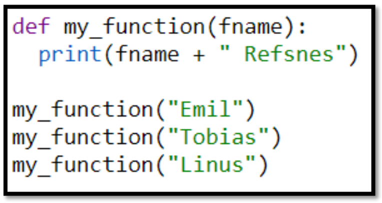
Python Functions

- Python has two primitive loop commands:
  - **while loops**: execute a set of statements as long as a condition is true.
  - **for loops:** used for iterating over a sequence
- With the break statement we can stop the loop even if the while condition is true
- With the continue statement we can stop the current iteration, and continue with the next
- use the range() function, to loop through a set of code a specified number of times
- nested loop is a loop inside a loop.
- for loops cannot be empty, but if you for some reason have a for loop with no content

Python Functions

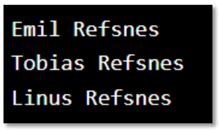
- In Python a function is defined using the def keyword
- To call a function, use the function name followed by parenthesis.
- Information can be passed into functions as arguments (args), add as many args as you want, just separate them with a comma. But, a function must be called with the correct number of args.
- To let a function return a value, use the return statement
- function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.
- Python also accepts function recursion, which means a defined function can call itself.

Python Functions

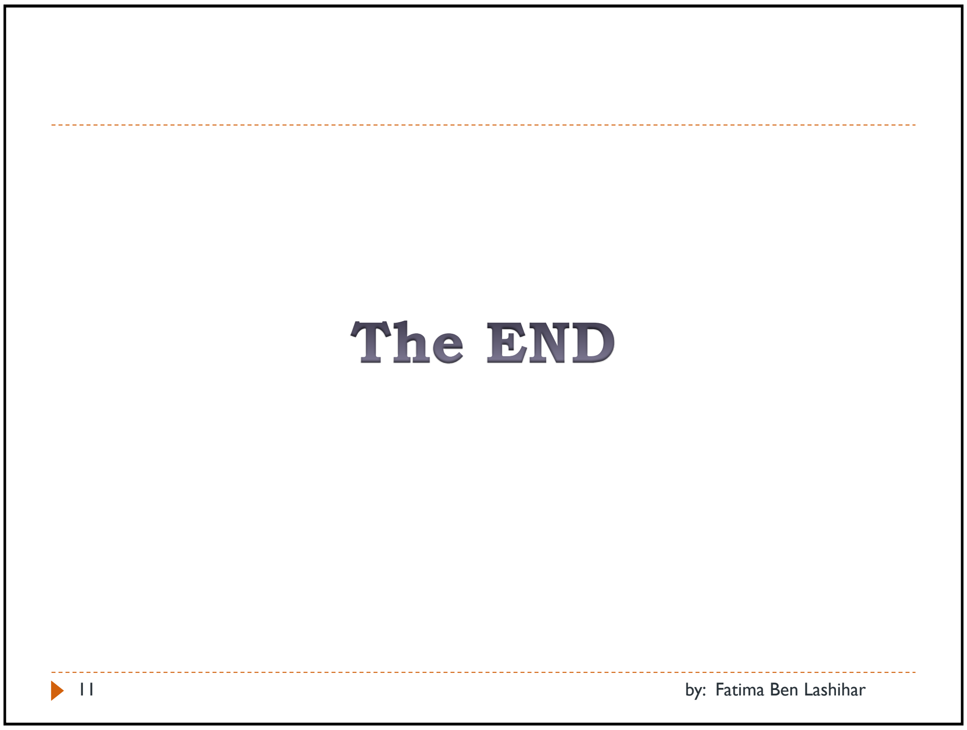


```python
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```



```
Emil Refsnes
Tobias Refsnes
Linus Refsnes
```



The END