COL1000
Introduction to Programming

Priyanka Golia

Most (if not all) of the content is borrowed from Prof. Subodh Kumar's slides

Conditional Execution

Conditionals let a program make decisions based on tests.

For example: "If it's raining, take an umbrella. Otherwise, don't."

The decision depends on the condition: "Is it raining?"

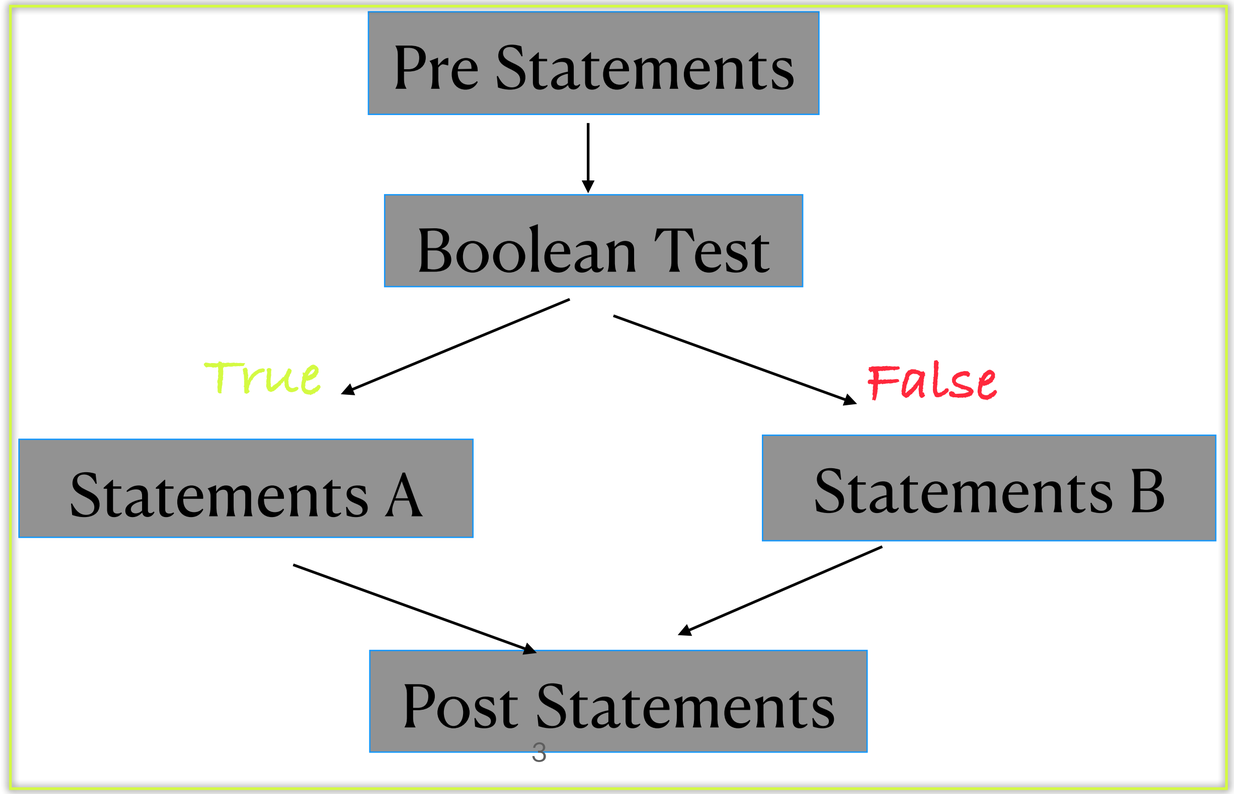
Every conditional is based on a **Boolean** test — something that evaluates to either: True or False

Python uses these True or False values to decide which part of the code to run.

Conditional Execution

Every conditional is based on a **Boolean** test — something that evaluates to either: True or False

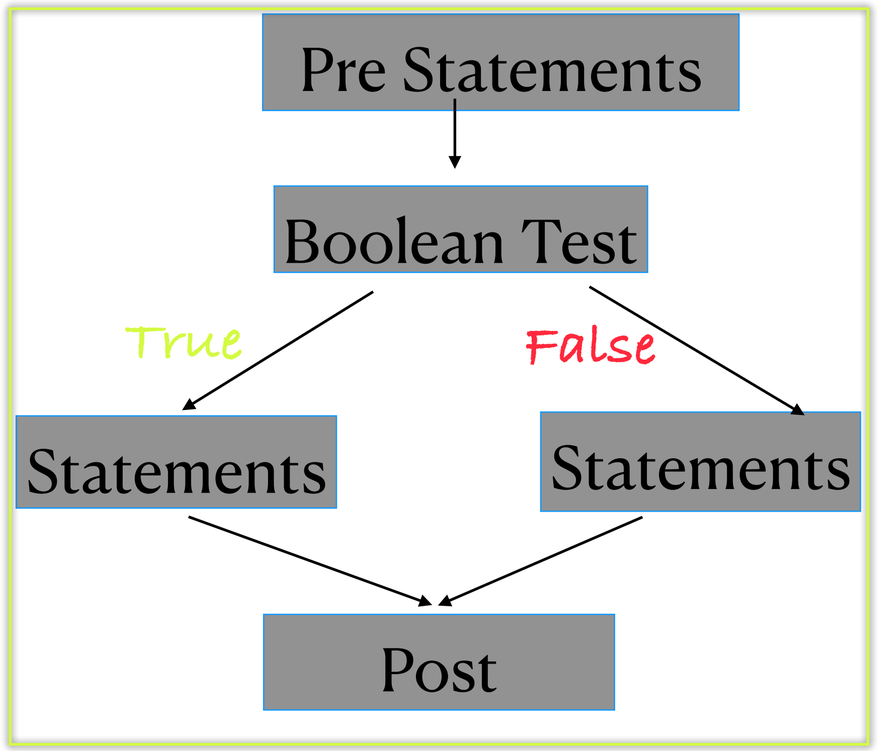
Python uses these True or False values to decide which part of the code to run.



Conditional Execution

Every conditional is based on a **Boolean** test — something that evaluates to either: True or False

Python uses these True or False values to decide which part of the code to run.




```
1 num = input("enter a number here ")
2 num = int(num)
3 if num > 10:
4     print("number entered is greater than 10")
5 else:
6     print("number is less than or equal to 10")
7 print("we have executed IF")
```

":" is important

Colon (:) tells that a block of code is coming next, and it belongs to this statement.

Indentation (tabs or spaces) is used to define blocks of code

Conditional Execution

```
1 num = input("enter a number here ")
2 num = int(num)
3 if num > 10:
4     print("number entered is greater than 10")
5 else:
6     print("number is less than or equal to 10")
7 print("we have executed IF")
```

```
enter a number here 11
number entered is greater than 10
we have executed IF
```

Number entered is 11, it executed the pre statement(s), checked for condition, and executed IF part, and then post statement(s).

```
enter a number here 10
number is less than or equal to 10
we have executed IF
```

Number entered is 10, it executed the pre statement(s), checked for condition, and executed else part, and then post statement(s).

Conditional Execution

If Without else — that's totally fine!

if block without an else — do something only if the condition is True. Otherwise, do nothing.

```
marks = int(input("enter your marks:  "))
if marks >= 90:
        print("you did a good job)
```

No else is needed!

It is okay not to have ELSE

```python
num = input(" enter a number: ")
if int(num) % 2 == 0:
    print("the given number is even")
else:
    print("the given number is odd")
```

```
 enter a number: 8
the given number is even
```

```
 enter a number: 11
the given number is odd
```

```python
num = input(" enter a number: ")
if int(num) % 2 == 0: # this is if#1
    print("the given number is even")
    if int(num) % 4 == 0: # this is if#2
        print("the number is divisible by 4")
else:
    print("the given number is odd")
```

Nested If

```
 enter a number: 12
the given number is even
the number is divisible by 4
```

```
 enter a number: 6
the given number is even
```

```
 enter a number: 11
the given number is odd
```

```
1  num = input(" enter a number: ")
2  if int(num) % 2 == 0: # this is if#1
3      print("the given number is even")
4      if int(num) % 4 == 0: # this is if#2
5          print("the number is divisible by 4")
6      else:
7          print('the number is not divisible by 4')
8      print("this is where the if#1  ends")
9  else:
10     print("the given number is odd")
```

if within if: aka Nesting

```
 enter a number: 6
the given number is even
the number is not divisible by 4
this is where the if#1  ends
```

```
 enter a number: 12
the given number is even
the number is divisible by 4
this is where the if#1  ends
```

```
 enter a number: 11
the given number is odd
```

One or more condition — logical Op.

```
if ( int(num) % 2 ) and (int(num) % 4):
    print("the given number is not divisible by 2 and 4")
else:
    if (int(num) % 2 == 0) and not (int(num) % 4):
        print("the given number is divisible by both 2 and 4")
    else:
        print("the given number is divisible by 2 and not by 4")
```

Nested if — inside Else block

Op. ***and*** : Returns True only if all conditions are True.
If even one is False → result is False.

Op. ***or*** : Returns True if at least one condition is True.
Only returns False if all conditions are False.

Op. ***not*** : Flips True to False, and False to True.

```
if ( int(num) % 2 ) and (int(num) % 4):
    print("the given number is not divisible by 2 and 4")
else:
    if (int(num) % 2 == 0) and not (int(num) % 4):
        print("the given number is divisible by both 2 and 4")
    else:
        print("the given number is divisible by 2 and not by 4")
```

Short Circuiting — Python stops checking further conditions as soon as it knows the final answer

With ***and*** — If the first condition is False, it doesn't check second— because the result will be False no matter what the second one is.

```
print(False and (10/0))  # No error,  (10/0) not checked
```

With **or** — If the first condition is True, it doesn't check second— because the result will be True no matter what the second one is.

```
print(True or (10/0))  # No error,  (10/0) not checked
```

```
x = int(input("enter a number "))
if x > 5 and (x // 5 == 1):
    print("x is greater than 5, and integer division x // 5 is zero")
    x += 10
if x > 10 :
    print("x is greater than 10")
print("executed the program")
```

Irrespective of whether the first “if” condition is met, the second “if” condition will still be checked. Note that the variable x is updated inside the first if block.

```
enter a number 7
x is greater than 5, and integer division x // 5 is zero
x is greater than 10
executed the program
```

```
x = int(input("enter a number "))
if x > 5 and (x // 5 == 1):
    print("x is greater than 5, and integer division x // 5 is zero")
    x += 10
elif x > 10 :
    print("x is greater than 10")
print("executed the program")
```

If the “ if” condition is met, the “else” part will not get executed.

```
enter a number 7
x is greater than 5, and integer division x // 5 is zero
executed the program
```

Nested Conditions

```
if number % 2:
    if(number % 3):
        print(number, 'is divisible by neither 2 nor 3')
    else
        print(number, 'is divisible by one of 2 and 3')
else:
    if(number % 3):
        print(number, 'is divisible by one of 2 and 3')
    else
        print(number, 'is divisible by both 2 and 3')
```

Homework —Use Logical Op. and Elif to rewrite the same program

Design of Conditions

- Check that the state is as expected (if not: take remedial action)
  - ➡ e.g., Is the input string convertible to an int?
  - ➡ Prevents later errors
- Follow different procedures (commands) for different cases
  - ➡ e.g., if absent: mark = 0.5 * mark
- Find all the different cases that the program needs to handle (at that stage), and divide them in separate branches
  - ➡ Organize into broad cases, then sub-cases within each case, and so on

(Top-down design: more later)

Design of Conditions

- A "fork in the execution path"

Not just what to do next, but whether to do so

1. One of two blocks of statements are executed, based on a "decision" value
2. Or, the execution of a block of statements could be skipped

- If you execute the program again, a different choice may be made

3. because the decision value may be different this time

- Fork within a fork (*nesting*) is allowed

Must Indent

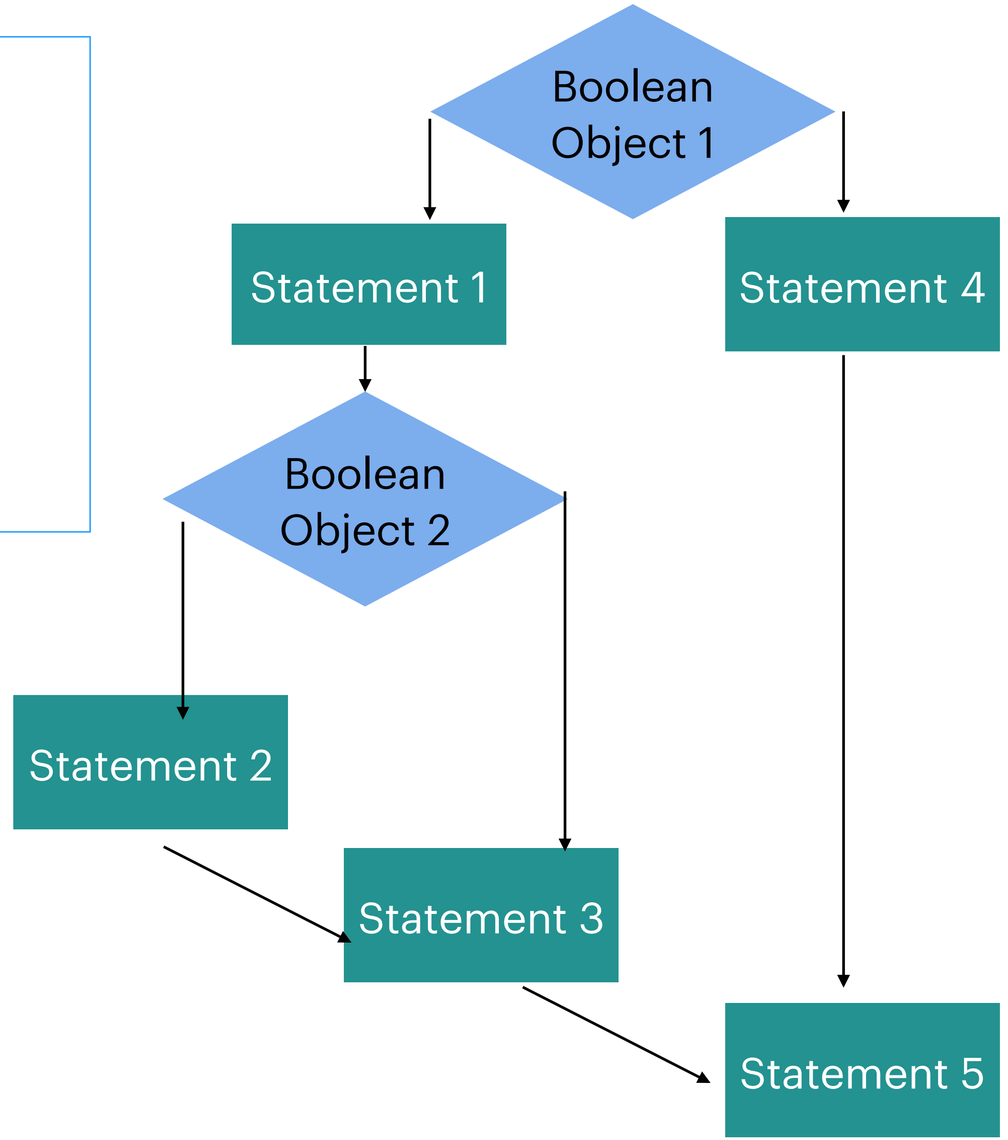
```
if <boolean object>:
    statements
else:
    statements
```

Remember colons

```
if <boolean object 1>:
    statements1
    if <boolean object 2>:
        statements2
    statements3
else:
    statements4
statements5
```

next statement

```
if <boolean object 1>:
    statements1
    if <boolean object 2>:
        statements2
    statements3
else:
    statements4
statements5
```



Example (Design of Conditions)

- *Is given year a leap year*
  - ➡ If year divisible by 4, except years divisible by 100, but not by 400

Divisibility by 4 is an important determinator

```
# year = …
if year%4: # Not divisible ⇒ always non-leap
    leap = False
else  # divisible by 4, not always leap — depends on other factors
    if year%100: # Not divisible by 100
        leap = True
    else:
        leap = not year%400 # Divisible
```

Same as

```
if year % 400:
    leap = False
else:
    leap = True
```

Same as

```
leap = not year%4 and year%100 or not year%400
```

No precedence override is needed here 👆 `%` ⊙ `not` ⊙ `and` ⊙ `or`