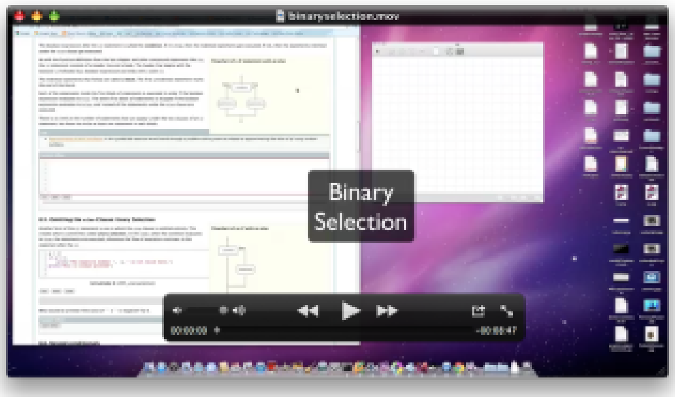
Conditional Execution: Binary Selection



In order to write useful programs, we almost always need the ability to check conditions and change the behavior of the program accordingly. **Selection statements**, sometimes also referred to as **conditional statements**, give us this ability. The simplest form of selection is the **if statement**. This is sometimes referred to as **binary selection** since there are two possible paths of execution.

```
x = 15

if x % 2 == 0:
    print(x, "is even")
else:
    print(x, "is odd")

```

ActiveCode: 1 (ch05_4)

Run

```
15 is odd
```

The syntax for an `if` statement looks like this:

```
if BOOLEAN EXPRESSION:
    STATEMENTS_1        # executed if condition evaluates to True
else:
    STATEMENTS_2        # executed if condition evaluates to False
```

The boolean expression after the `if` statement is called the **condition**. If it is true, then the indented statements get executed. If not, then the statements indented under the else clause get executed.

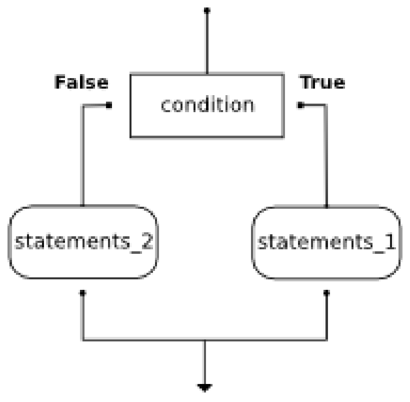
As with the function definition from the last chapter and other compound statements like `for`, the `if` statement consists of a header line and a body. The header line begins with the keyword `if` followed by a *boolean expression* and ends with a colon (:).

The indented statements that follow are called a **block**. The first unindented statement marks the end of the block.

Each of the statements inside the first block of statements is executed in order if the boolean expression evaluates to `True`. The entire first block of statements is skipped if the boolean expression evaluates to `False`, and instead all the statements under the `else` clause are executed.

There is no limit on the number of statements that can appear under the two clauses of an `if` statement, but there has to be at least one statement in each block.

Flowchart of a if statement with an else



Lab

- Approximating Pi with Simulation (http://dcs.asu.edu/faculty/abansal/CST100/Labs/Decisions&Selection-ApproximatingValueOfPi.html) In this guided lab exercise we will work through a problem solving exercise related to approximating the value of pi using random numbers.

Check your understanding

sel-4: How many statements can appear in each block (the if and the else) in a conditional statement?

- a) Just one.
- b) Zero or more.
- [✓ c) One or more.]
- d) One or more, and each must contain the same number.

Check Me Compare Me

Correct!! Yes, a block must contain at least one statement and can have many statements.

sel-5: What does the following code print (choose from output a, b, c or nothing).

```
if (4 + 5 == 10):
    print("TRUE")
else:
    print("FALSE")
```

- ○ a) TRUE
- [✓ ● b) FALSE]
- ○ c) TRUE on one line and FALSE on the next
- ○ d) Nothing will be printed

Check Me Compare Me

Correct!! Since 4+5==10 evaluates to False, Python will skip over the if block and execute the statement in the else block.

sel-6: What does the following code print?

```
if (4 + 5 == 10):
    print("TRUE")
else:
    print("FALSE")
print("TRUE")
```

```
a. TRUE

b.
   TRUE
   FALSE

c.
   FALSE
   TRUE
d.
   TRUE
   FALSE
   TRUE
```

a) Output a

b) Output b

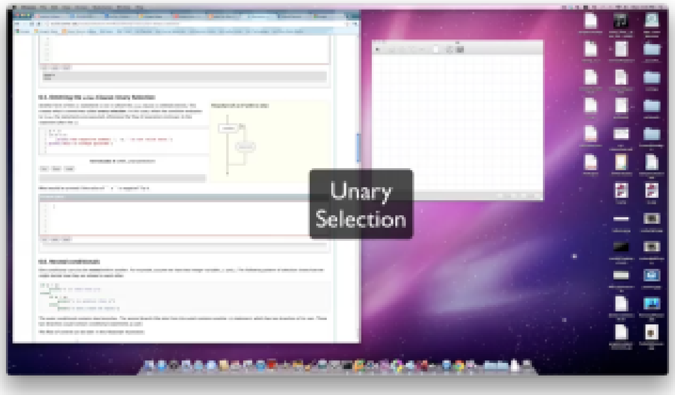
[✓ c) Output c]

d) Output d

Check Me Compare Me

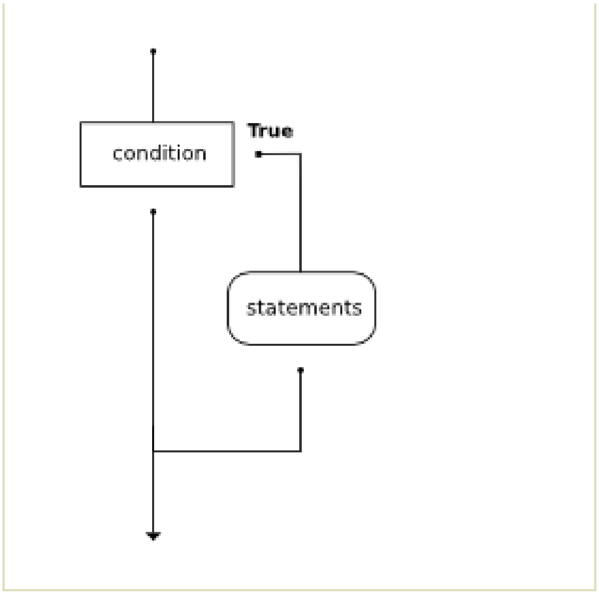
Correct!! Python will print FALSE from within the else-block (because 5+4 does not equal 10), and then print TRUE after the if-else statement completes.

Omitting the else Clause: Unary Selection



Flowchart of an if with no else

Another form of the `if` statement is one in which the `else` clause is omitted entirely. This creates what is sometimes called **unary selection**. In this case, when the condition evaluates to `True`, the statements are executed. Otherwise the flow of execution continues to the statement after the body of the `if`.



```
x = -10
if x < 0:
    print("The negative number ",  x, " is not
print("This is always printed")

```

ActiveCode: 2 (ch05_unaryselection)

Run

```
The negative number  -10  is not valid here.
This is always printed
```

What would be printed if the value of `x` is negative? Try it.

Check your understanding

sel-7: What does the following code print?

```
x = -10
if x < 0:
    print("The negative number ",  x, " is not valid here.")
print("This is always printed")
```

```
a.
This is always printed

b.
The negative number -10 is not valid here
This is always printed

c.
The negative number -10 is not valid here
```

a) Output a [✓ b) Output b] c) Output c d) It will cause an error because every if must have an else clause.

Check Me Compare Me

Correct!! Python executes the body of the if-block as well as the statement that follows the if-block.

sel-8: Will the following code cause an error?

```
x = -10
if x < 0:
    print("The negative number ",  x, " is not valid here.")
else:
    print(x, " is a positive number")
else:
    print("This is always printed")
```

a) No [✓ b) Yes]

Check Me Compare Me

Correct!! This will cause an error because the second else-block is not attached to a corresponding if-block.