Python

Programming Logic

Intro

Up until now we have been dealing with sequential programs - the program executes line by line until the last line.

Python also supports structures known and selection and iteration.

Selection

A selection structure is like a decision.

It provides programmers with a branching mechanism whereby certain blocks of code may be either executed or skipped at runtime.

The decision of which block of code to select for execution depends on the result of a **condition** also known as a **Boolean expression.**

The main Python keywords used to support decision structures are **if, else and elif.**

Boolean Logic

Boolean Logic was invented by the mathematician George Boole who was the first professor of Mathematics at UCC.

The algebra on which Boolean Logic is based is used to build electronic circuits and write computer programs.

Boolean logic forms the basis of all modern digital devices and software systems.

Boolean Expressions

At any given moment in time, a Boolean expression will evaluate to either **True** or **False.** It can never be anything in between.

True and False are two Python keywords which technically behave as if they were the numbers 0 and 1.

A simple boolean expression uses a single relational operator (<, >, =) to compare two values.

Example:

7>3 evaluates to True

8<4 evaluates to False

Python Relational Operators

Operator	Description	Example	Result
>	Greater than	7 > 5	True
>=	Greater than or equal to	7 >= 5	True
<	Less than	7 < 5	False
<=	Less than or equal to	7 <= 5	False
==	Equal to (the same as)	7 == 5	False
!=	Not equal to (not the same as)	7 != 5	True

Boolean Operators

Boolean expressions can be combined using additional boolean operators

- and
- or
- not

Boolean operator	Explanation
and	Both parts of the expression must be true for the whole expression to result in the answer `True`.
or	Only one part of the expression needs to be true for the whole expression to result in the answer `True`.
not	The `not` operator operates on only one Boolean expression. It returns the opposite value of the expression, e.g., if the value of the expression is `True`, `not` returns the value `False`.

Selection - if, else, elif

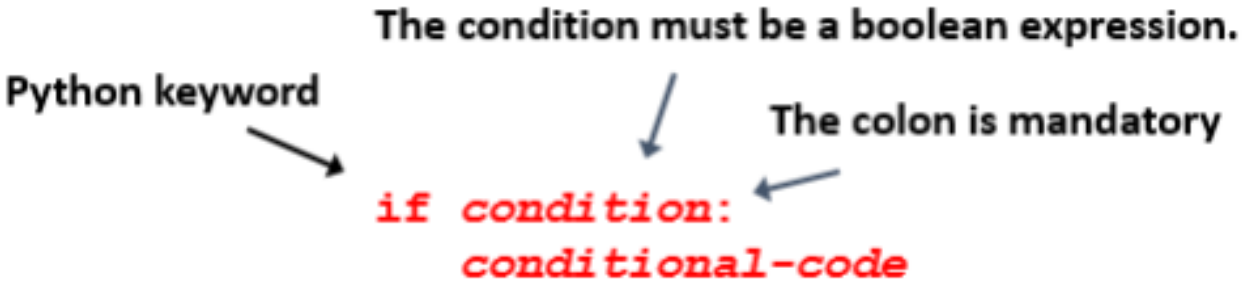
Selection statements are used by programmers to build alternative execution paths through their code. These are known as decision statements.

Python provides built in supports for three different kinds of selection statements:

- Single option (basic if statement)
- Double option (the if-else statement)
- Multiple option (if-elif-else statement)

When a running program executes a selection statement, it evaluates a condition, and based on the result of this evaluation it will decide which statement to execute next.

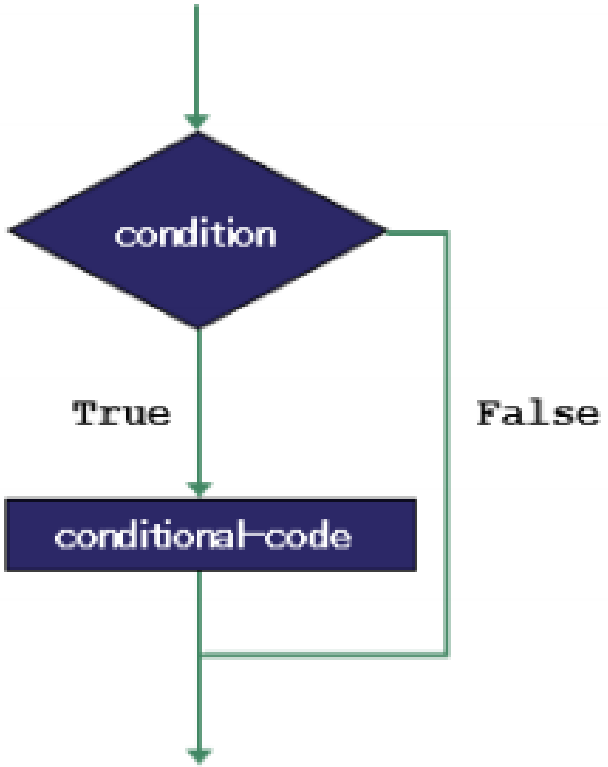
The basic **if** statement



If Python evaluates the conditional statement to be True, then the conditional code will be executed.

If Python evaluates the conditional statement to be False, then the conditional code is skipped and the programs continues to the next line of code.

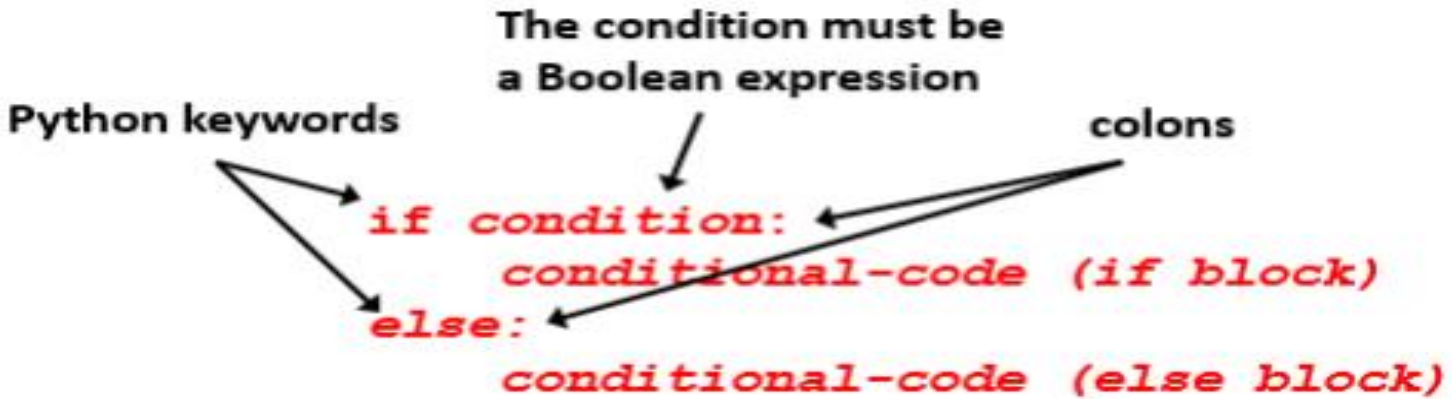
NOTE: The colon and indentation of the conditional code is mandatory



Flow chart illustration of `if`-statement

```
1.  # A program to demonstrate the single if statement
2.  import random
3.
4.  number = random.randint(1, 10)
5.  # print(number)
6.
7.  guess = int(input("Enter a number between 1 and 10: "))
8.
9.  # Evaluate the condition
10. if guess == number:
11.     print("Your guess was correct")
12.     print("Well done!")
13.
14. print("Goodbye")
```

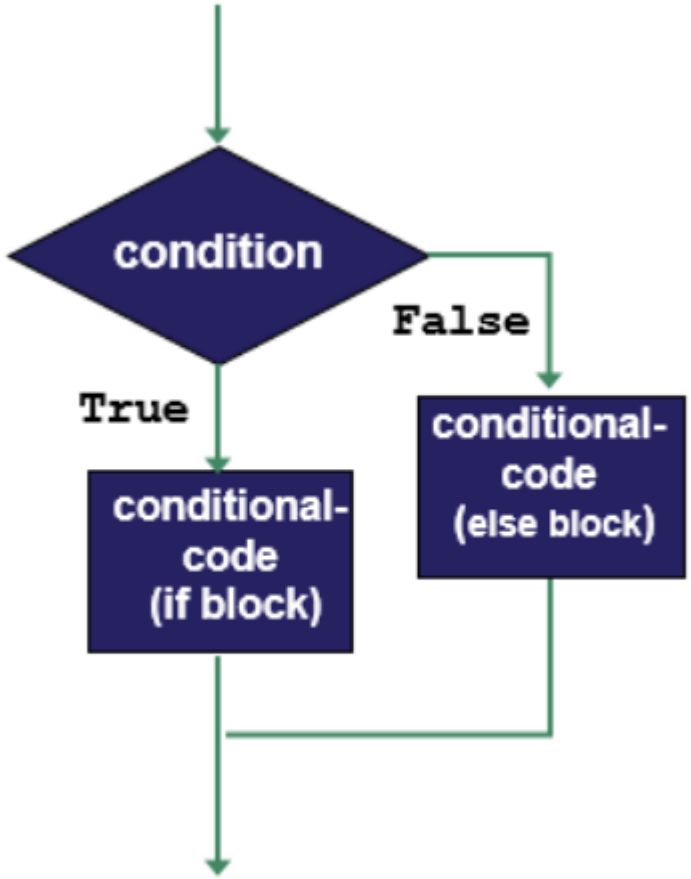
If-else statement



If Python evaluates the condition to be True, then the block of code associated with the if block is executed.

Otherwise, the block of code associated with the else block is executed.

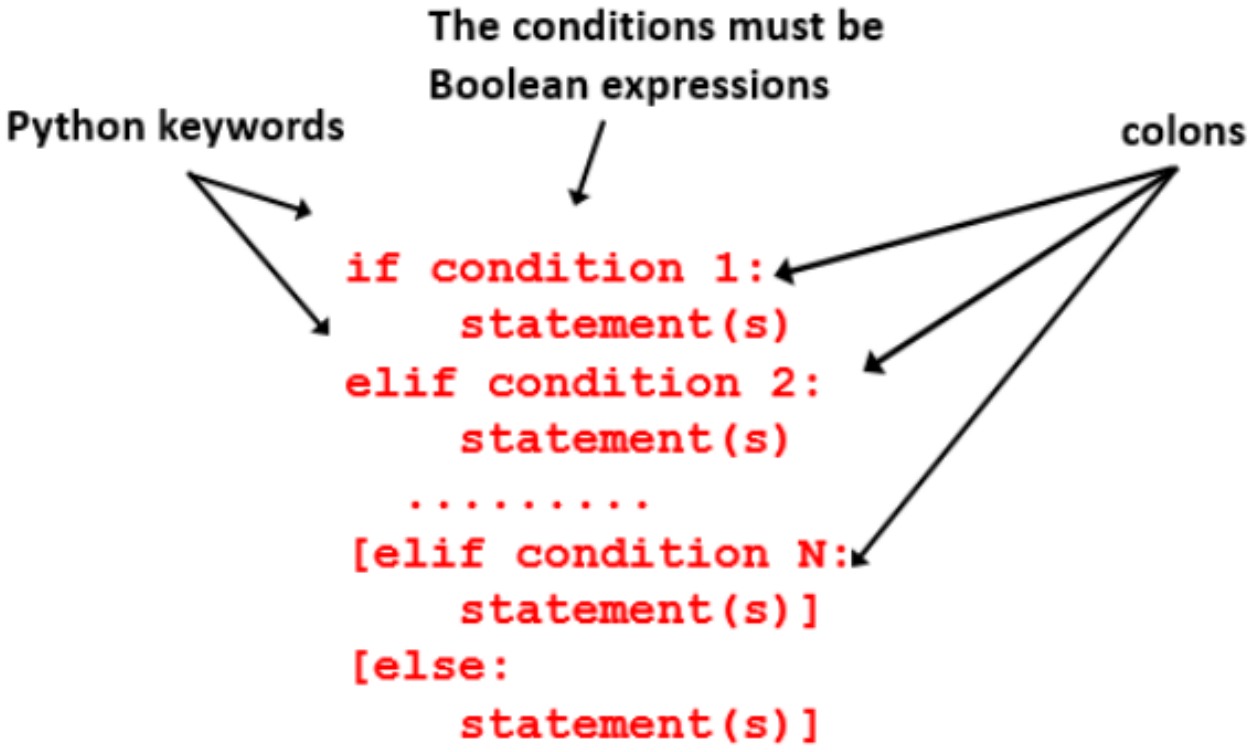
Only one block will run, never both.



Flow chart illustration of `if-else`-statement

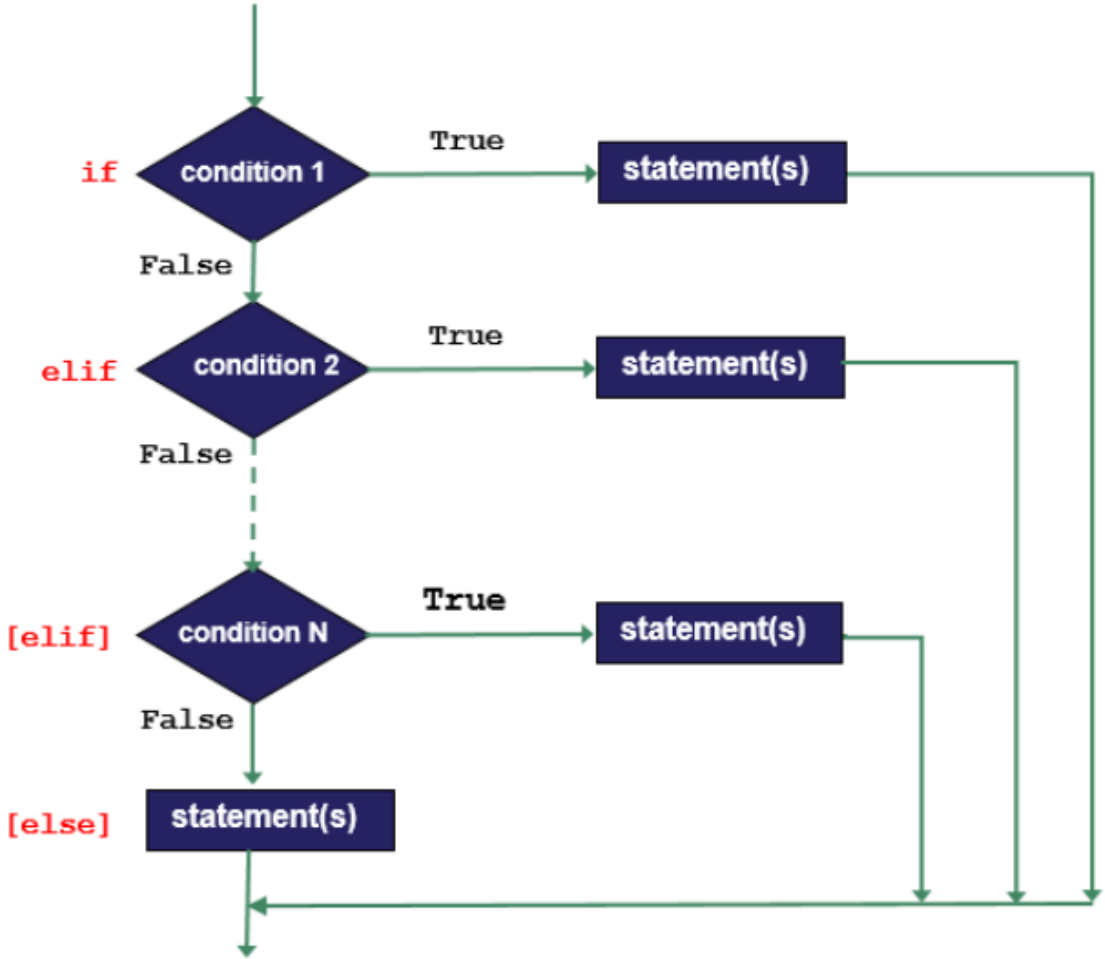
```
1.  # A program to demonstrate the double if statement
2.  import random
3.
4.  number = random.randint(1, 10)
5.  print(number) # comment this line out later!
6.
7.  guess = int(input("Enter a number between 1 and 10: "))
8.
9.  # Evaluate the condition
10. if guess == number:
11.     print("Your guess was correct")
12.     print("Well done!")
13.     print(" ..... play again soon!")
14. else:
15.     print("Hard luck!")
16.     print("Incorrect guess")
17.     print(" ..... play again soon!")
18.
19. print("Goodbye")
```

If-elif-else statement



if-elif-else

- The first condition is always inside an if statement
- There can be as many elif statements as required.
- Each elif statement must include a condition
- The use of a final else statement is optional
- The if, elif and else keywords must all be at the same level of indentation
- A colon must be used at the end of any lines containing if, elif and else
- Each condition is evaluated in sequence. Should Python evaluate a condition to be True, then the associated statements are executed and the flow of control continues from the next line following the end of the entire if-elif statement
- If none of the conditions are found to be True then Python executes any statements associated with the else


if
condition 1
True
statement(s)
False
elif
condition 2
True
statement(s)
False
[elif]
condition N
True
statement(s)
False
[else]
statement(s)

```
1.  # A program to demonstrate the multiple if statement
2.  import random
3.
4.  number = random.randint(1, 10)
5.  print(number) # comment this line out later!
6.
7.  guess = int(input("Enter a number between 1 and 10: "))
8.
9.  # Evaluate the condition
10. if guess == number:
11.       print("Correct")
12.       print("Well done!")
13. elif guess < number:
14.     print("Hard luck!")
15.     print("Too low")
16. else:
17.     print("Hard luck!")
18.     print("Too high")
19.
20. print("Goodbye")
```