**Chapter 3 Selection**

It is possible in a program to choose whether statements are executed or not, or to choose between statements or sets of statements. Statements that accomplish this are called ***selection*** statements and include the **if** statement and the **if-else** statement. Python has additional selection statements including the **match** statement and **try-except**.

**3.1 If Statements**

The **if** statement chooses whether statement(s) are executed or not. The general form is

if condition:

action

# rest of code

The **if** statement starts with the reserved word **if**, then an expression that evaluates to either **True** or **False**, then a colon. The expression is frequently called a ***condition***. After that, the ***action*** of the **if** is indented. The action consists of one or more statements that are executed only if the expression evaluates to **True**. If the expression evaluates to **False**, the action is skipped.

The indentation is very important in Python. If the action consists of more than one statement, all must be indented to the same level. It is customary to indent by 4 spaces. Note: use the space bar, not the tab key!

In the following example, it is **True** that the value of the variable *num* is less than 50, so the action is executed, which prints ‘It is smaller’.

| *print('OK')*  *num = 33*  *if num < 50:*  *print('It is smaller') print('And that is it')* |
| --- |

OK

It is smaller

And that is it

If the value of *num* is changed to be something greater than or equal to 50, the action would be skipped entirely.

| *print('OK')*  *num = 62*  *if num < 50:*  *print('It is smaller') print('And that is it')* |
| --- |

OK

And that is it

In interactive environments in which cells are not used, the **if** statement will result in the use of primary and secondary prompts. For example, it might look like this:

In [1]: num = 62

In [2]: if num < 50:

...: print('It is smaller')

...:

In [3]: print('And that is it')

In this case, the primary prompt is In [n]:, and the secondary prompt is ...: Just hitting the enter key at the secondary prompt will end the action of the **if** statement.

Of course, building in the value of the variable *num* by assigning a value, and then testing it, does not really make sense! It would make much more sense to generate a random number, or to prompt the user for the value of the variable *num*, and then print (or not) based on the random number or what the user entered.

| from random import randint num = randint(0,100)  if num < 50:  print('It is smaller') print('And that is it') |
| --- |

It is smaller

And that is it

We can use the **in** operator to test to see whether a particular character is in a string, or whether a particular value is in a list.

| if 'x' in 'abcde':  print("Yay, there is an 'x'!") print('Done.') |
| --- |

Done.

Again, it would make more sense if the string was not built in. It could instead be something entered by the user.

| urname = input('Please enter your name: ') if 'z' in urname:  print("Yay, there is a 'z'!")  print('OK.') |
| --- |

Please enter your name: Frazier

Yay, there is a 'z'!

OK.

**3.2 If-else Statements**

To choose between two statements, or sets of statements, the **if-else** statement is used. The general form is

if condition:

ifaction

else:

elseaction

# rest of code

The **if-else** statement chooses between two actions, called ‘ifaction’ and ‘elseaction’ here. The way it works is that the condition is evaluated. If the value of the condition is **True**, then all of the statements in the ‘ifaction’ will be executed, and the **if-else** statement ends. If, on the other hand, the value of the condition is **False**, the statement(s) in the second action, ‘elseaction’, will be executed. The **if-else** statement chooses between executing the ‘ifaction’ or the ‘elseaction’. One of these actions, and only one, will be executed. The terminology is that ***control*** goes to the chosen action, and those statements are executed. Once the statements in the chosen action have been executed, control goes to the rest of the code after the **if-else** statement.

| *num = input('Enter a number: ')*  *num = float(num) # convert the string to a float if num < 0:*  *print(f'{num} is a negative number')*  *else:*  *print(f'{num} is a nonnegative number')* |
| --- |

In this example, the user is asked for a number. The code then prints whether it is a negative number or a nonnegative number. Here are two examples of what the output would look like:

Enter a number: -33

-33.0 is a negative number

Enter a number: 14

14.0 is a nonnegative number

In order to choose from more than one option, it is possible to ***nest* if-else** statements, which means putting one inside of another.

| *num = input('Enter a number: ')*  *num = float(num) # convert the string to a float if num < 0:*  *print(f'{num} is a negative number')*  *else:*  *if num == 0:*  *print('It is a zero')*  *else:*  *print(f'{num} is a positive number')* |
| --- |

When this is executed, the expression *num < 0* is evaluated. If that is **True**, the code prints that it is a negative number and the entire **if-else** statement ends. If, however, it is **False**, the action consisting of the second **if-else** statement is executed. The second, nested, **if-else** statement evaluates the expression *num == 0*. If that expression is **True**, it prints that it is a zero, and if not it prints that it is a positive number.

Notice the indentation. In this example, all actions were indented 4 spaces. This can be simplified using the **elif** keyword, as follows.

| *num = input('Enter a number: ')*  *num = float(num) # convert the string to a float if num < 0:*  *print(f'{num} is a negative number')*  *elif num == 0:*  *print('It is a zero')*  *else:*  *print(f'{num} is a positive number')* |
| --- |

With **elif**, it is not necessary to have **else** followed by another **if-else** statement. The indentation, which is required in Python, is also simpler using **elif**. There can be as many **elif** clauses as necessary to handle all of the possible actions. It is not a requirement to have an **else** at the very end, although it is common to do so to handle ‘none of the above’.

Note that in Python, unlike many languages, relational operators can be ***chained*** together. The expression *a < x < b*, for example, is equivalent to the expression *a < x and x < b*. The following will test whether or not a variable *num* is in the range from 5 to 10 inclusive.

| *if 5 <= num <= 10:*  *print('In range')*  *else:*  *print('Not in range')* |
| --- |

Like **if** statements, **If-else** statements can also be used with strings. For example, the following uses the **in** operator to determine whether or not the character ‘x’ is in a string variable *mystr*.

| *if 'x' in mystr:*  *print('There is an x!') else:*  *print('Alas, no x.')* |
| --- |

**If-else** statement can also be used to print the range of a random number, for example from **randint**.

| *from random import randint*  *ri = randint(0,10)*  *print(ri, "is", end = " ")*  *if ri < 5:*  *print("in range 0 to 5 inclusive") else:*  *print("in range 5 through 10")* |
| --- |

3 is in range 0 to 5 inclusive

Try running this code multiple times to see the (possibly) different results!