**Controlling Execution**

So far all of our scripts have just been a sequence of commands executed one after the other. In order to produce more powerful scripts we need to be able to control the flow of the program.

Selection using **if**

We are going to start with looking at the if/else construct, where we make a decision based on a test. The syntax is shown in boldface, and the parts that you must supply are in normal type:

**if** condition**:**

execute these commands

**else:**

execute these commands

The **else** clause is optional, and allows you to specify what happens if the test fails. You must remember to indent blocks!

You can also have nested **if** clauses by using the **elif** command. **elif** is short for "else if." It means exactly what it sounds like: "otherwise, if the following expression is true, do this”

**if** condition**:**

execute these commands

**elif** condition**:**

execute these commands

**else**:

execute these commands

So what kind of conditions can we test for? If you’re dealing with numbers or strings, here are the conditional expressions you can use. In other words, any of these expressions can form the *condition* part of the **if** or **elif** statement:

variable1 **==** variable2 True if variable1 equals variable2

variable1 !**=** variable2 True if variable1 is not equal to variable2

variable1 **<** variable2 True if variable1 is less than variable2

variable1 **>** variable2 True if variable1 is greater than variable2

variable1 **<=** variable2 True if variable1 is less than or equal to variable2

variable1 **>=** variable2 True if variable1 is greater than or equal to variable2

You can also test certain file conditions, such as whether or not the file exist, the type of file, and so on. Here are the conditional expressions for files:

To check if a file/directory exists:

**import os.path**

**os.path.exists(file\_path)**

This returns True for both files and directories.

Use **os.path.isfile(file\_path)** to test if it's a file specifically.

Use **os.path.isdir(file\_path)** to test if it's a directory specifically.

To test for a non-zero file size:

**import os**

**os.path.getsize(file\_path) > 0**

True is file\_path contains data (non-zero size)

Here are the logical operators for performing tests that involve and, or, not conditions:

cond1 **and** cond2 True if both cond1 and cond2 are true

cond1 **or** cond2 True if either cond1 or cond2 are true

**not(**cond1) True if cond1 is false

Selection using **case** Statement

Unlike most other programming languages, Python does not have a switch or case statement. case statement lets you compare a string with several possible values and execute a block of code when it finds a match. To get around this fact, we use dictionary mapping:

**def** numbers\_to\_strings(argument):

switcher = {

0: "zero",

1: "one",

2: "two",

}

**return** switcher.get(argument, "nothing")

The dictionary **get** method returns the value for the key “argument” or the default value (“nothing”) if the key is not present in the dictionary.

In Python we can also include functions or lambdas in our dictionary mapping:

**def** zero():

**return** "zero"

**def** one():

**return** "one"

**def** numbers\_to\_functions\_to\_strings(argument):

switcher = {

0: zero,

1: one,

2: **lambda**: "two",

}

*# Get the function from switcher dictionary*

func = switcher.get(argument, **lambda**: "nothing")

*# Execute the function*

**return** func()

lambda is an anonymous function which doesn’t have a name and is used inline

addTwo = lambda x: x+2

>>> addTwo(2)

4

This assigns addTwo to the anonymous function, which accepts 1 argument x, and in the function body it adds 2 to x, it returns the last value of the last expression in the function body so there's no return keyword.