# Reading and Research - Selection Statements

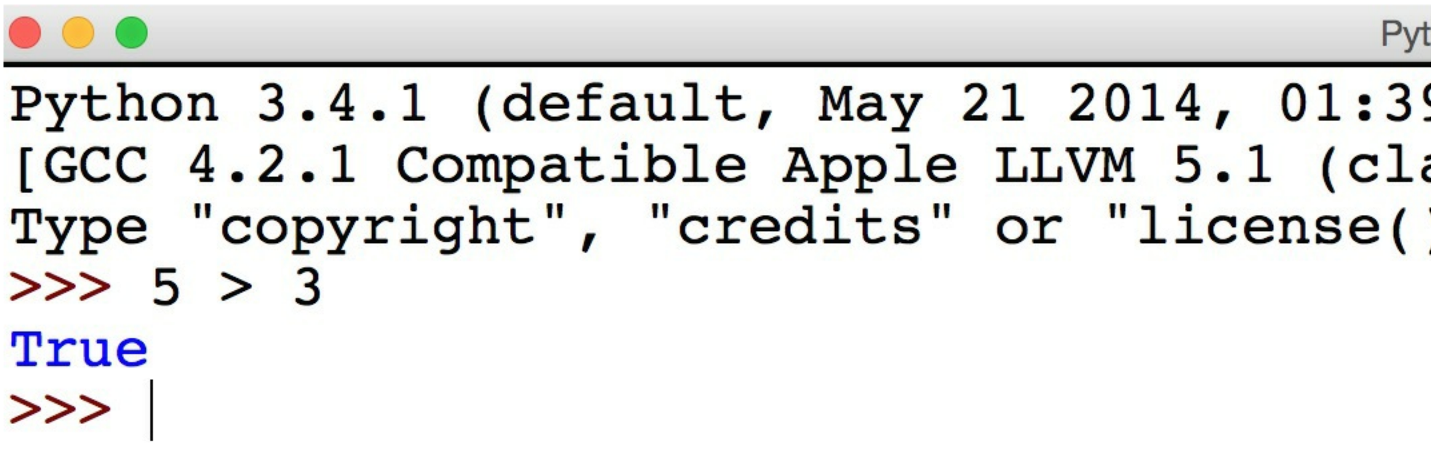
These tasks are designed to introduce you to the programming topic we will be studying in class next lesson. You **must** complete these activities prior to the lesson.

## Boolean Expressions

One of the most common tasks in computer programming is to **evaluate an expression**. An expression allows us to test whether a value (or set of values) meets particular criteria. The Python shell can evaluate expressions, we will use this to investigate expressions further.

### Task 1

Use the Python shell to investigate the expressions given below, describe what each symbol represents and indicate whether the expression evaluates to True or False.



```
Python 3.4.1 (default, May 21 2014, 01:39
[GCC 4.2.1 Compatible Apple LLVM 5.1 (cla
Type "copyright", "credits" or "license(
>>> 5 > 3
True
>>> 
```

| Expression | Symbol description | Result |
|---|---|---|
| 2 == 4 | *equals* | False |
| 5 > 3 | grater than | True |
| 4 >= 4 | Greater or equal to | True |
| 3 < 2 | Less than | False |
| 7 <= 7 | Less than or equal to | True |
| 8 != 9 | Not equal | True |

The symbols in **Task 1** are called **relational operators** and when an expression containing a relational operator is evaluated it returns a **boolean value** (True or False) as an answer.

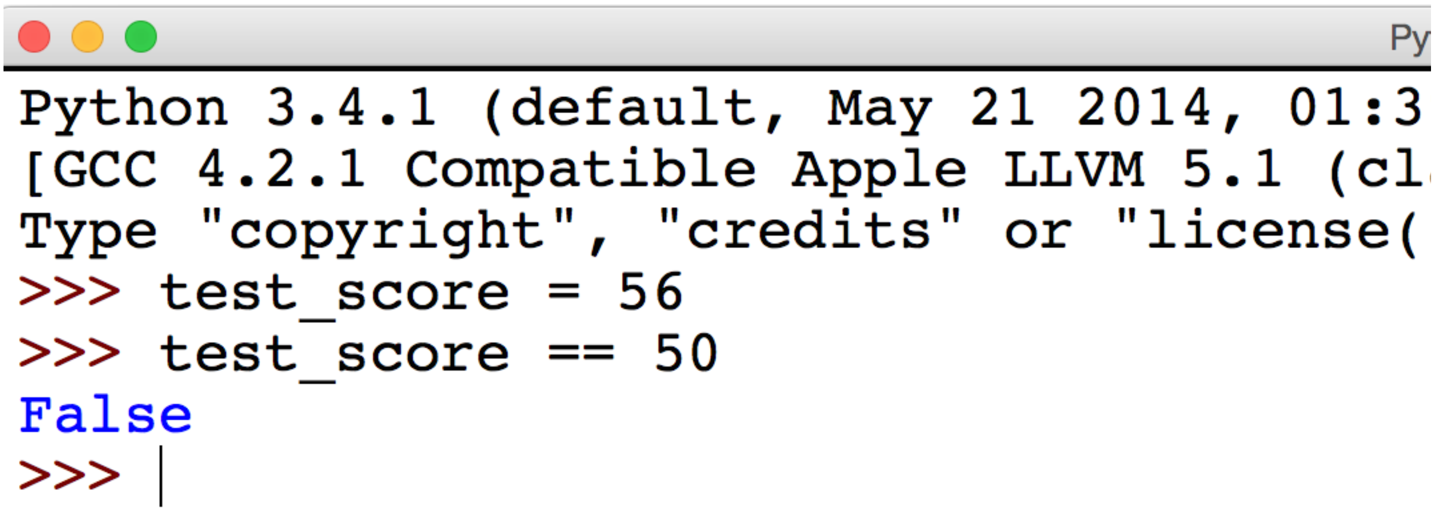
In addition to evaluating expressions containing numbers we can also use **variables** in expressions. For example, imagine we had the following variable:

```
test_score = 56
```

We could use boolean expressions to evaluate whether testScore meets certain criteria (for example whether it is greater than the pass mark of 50). Let’s test this out:

### Task 2

Enter testScore = 56 into the Python shell and then investigate the expressions below.



```
Python 3.4.1 (default, May 21 2014, 01:3
[GCC 4.2.1 Compatible Apple LLVM 5.1 (cl
Type "copyright", "credits" or "license(
>>> test_score = 56
>>> test_score == 50
False
>>> 
```

| Expression | Symbol description | Result |
|---|---|---|
| test_score == 50 | *equals* | False |
| test_score > 40 | greater than | True |
| test_score >= 60 | greater than or equal to | False |
| test_score < 40 | less than | False |
| 50 <= test_score | less than or equal to | True |
| 56 != test_score | not equal | False |

## More complex boolean expressions

Sometimes it is not enough to evaluate an expression on a single criteria. We can create more complicated boolean expressions using boolean operators. There are three boolean operators that we must consider in programming:

| Operator |
|---|
| and |
| or |
| not |

The and and or operators can be used to join expressions together into more complex expressions. The not operator is used to invert an expressions evaluation. For example if an expression evaluated to True using the not operator would make the result equal False.

### Task 3

Let's look at some straightforward examples. Use the Python shell to evaluate the following expressions:

| Expression | Result |
|---|---|
| True and True | True |
| True and False | False |
| False and True | False |
| False and False | False |
| True or True | True |
| True or False | True |
| False or True | True |
| False or False | False |
| not(True) | False |
| not(False) | True |

Having completed the above table, use the space below to describe when and and or evaluate to True:

| Operator | When it evaluates to True |
|---|---|
| and | will only evaluate to True when both side are true |
| or | or will evaluate to true even if one side is True and the other is False or vise versa. |

## Selection statements

Before we find out more about selection statements let look at an example:

```
test_score = 56
if test_score >= 50:
    print("Pass")
if test_score < 50:
    print("Fail")
```

### Task 4

Without entering the code into Python, attempt to explain what the code does, using the space below for your answer:

**answer**

The first line of the code tells the computer that the test score is 56 and the computer remembers it. The second line commands the computer to print "Pass" if the number you type in is greater than or equal to 50. However the third line commands the computer to print "Fail" if the number you type in is less than 50.

Now that we have looked at an example it is time to investigate selection statements in more detail. We will use the Python School website to do this.

## Task 5

Read the following two pages on Python Summer School and attempt the exercises mentioned.

1. The IF Statement in Python
    - The exercise at the bottom of the page
2. More on IF Statements in Python
    - The **first** exercise at the bottom of the page

## Task 6

In the space below **paste** the code from each of the exercises in Task 5 and include a screenshot of you running each program successfully.

```
#task 5.1
age = int(input("What is your age: "))
if age >= 18:
    print("You are old enough to vote")
else:
    print("You are not old enough to vote")
```

```
#task 5.2
money = int(input("How much money do you have"))
if money > 16:
    print(" You have enough money to buy headphones")
elif money >= 15 :
    print(" You have just enough money to buy headphones")
else:
    print(" you do not have enough money to buy headphones")
```

# Summary

In this R&R you have investigated selection statements. You have seen how expressions are constructed from relational operators and boolean operators. You have have seen the structure and syntax of a basic selection statement and had the opportunity to create programs that use this statement.

Please make sure you have completed this R&R fully before your next programming lesson as it will form the basis of the initial classroom discussion and starter tasks.