# Reading and Research - Selection Statements

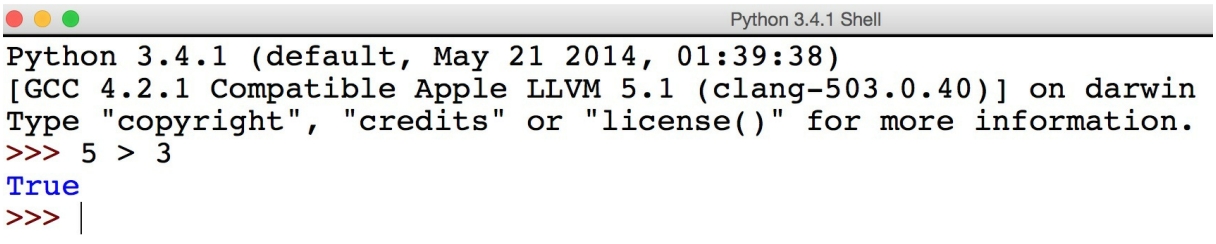
These tasks are designed to introduce you to the programming topic we will be studying in class next lesson. You **must** complete these activities prior to the lesson.

## 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`.




| Expression | Symbol description | Result |
|---|---|---|
| `2 == 4` | *equals* | `False` |
| `5 > 3` | *Greater than* | True |
| `4 >= 4` | *Greater than or equal to* | True |
| `3 < 2` | *less than* | False |

| Expression | Symbol description | Result |
|---|---|---|
| `7 <= 7` | *less than or equal to* | True |
| `8 != 9` | *are the two values equal or not* | 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.

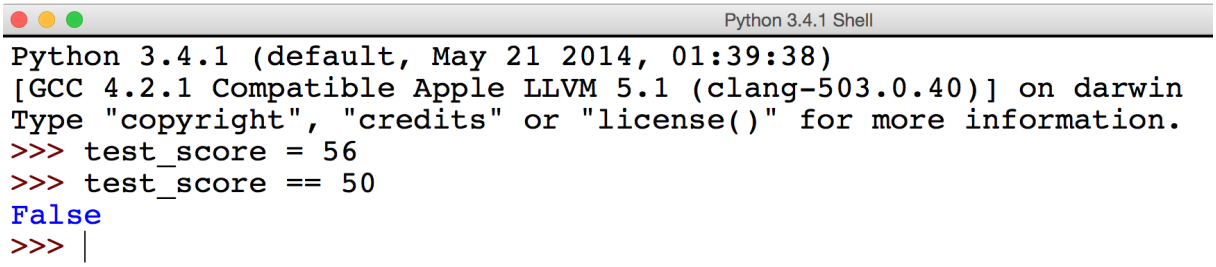
In addition to evaluating expressions containing numbers we can also use **variables** in expressions. For example, imagine we had the following variable:

```
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.




| 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` | *are the values equal or 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 |

| Expression | Result |
|---|---|
| `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` | When both “True” |
| `or` | When one is “True” |

# 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 |
|---|
| *Takes the variables and if the number is larger than or equal to 50 is prints “pass.” However if the variable is below 50 it prints “Fail.”* |

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.

“ `python

# task 5.1

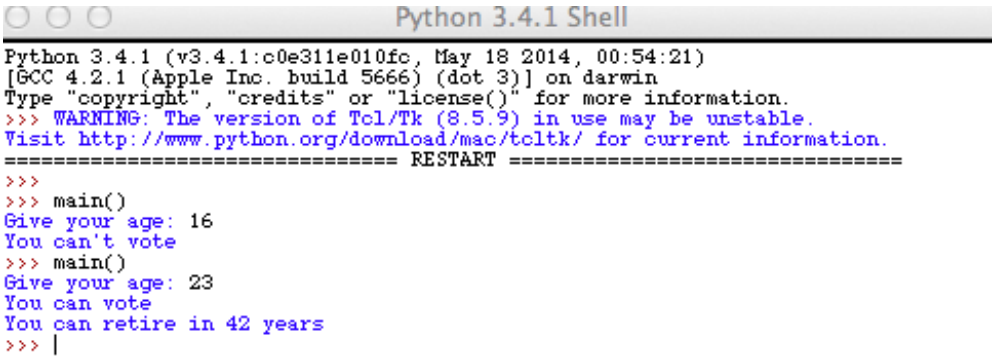
# Harry Robinson

# 21-09-2014

# Programme asking for age and displaying if you can vote

```
def main ():
age = int(input(“Give your age: “))
if age <= 18:
```

print(“You can’t vote”)
else :
print (“You can vote”)
retirement = 65 - age
if age >= 18:
print(“You can retire in {0} years”.format(retirement))“ `




You can’t vote

# task 5.2

“ `python

# Harry Robinson

# 21-09-2014

# Programme asking for a number within a range

def main():
number1 = int(input("Enter a number; "))
if number1 <= 20 and number1>=1:
print("Within range")
else:

```
    print("Out of range")
```

```
>>>
>>> main()
Enter a number; 20
Within range
>>> 1
1
>>> main()
Enter a number; 1
Within range
>>> main()
Enter a number; 25
Out of range
>>> main()
Enter a number; -1
Out of range
>>>
```

# 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.