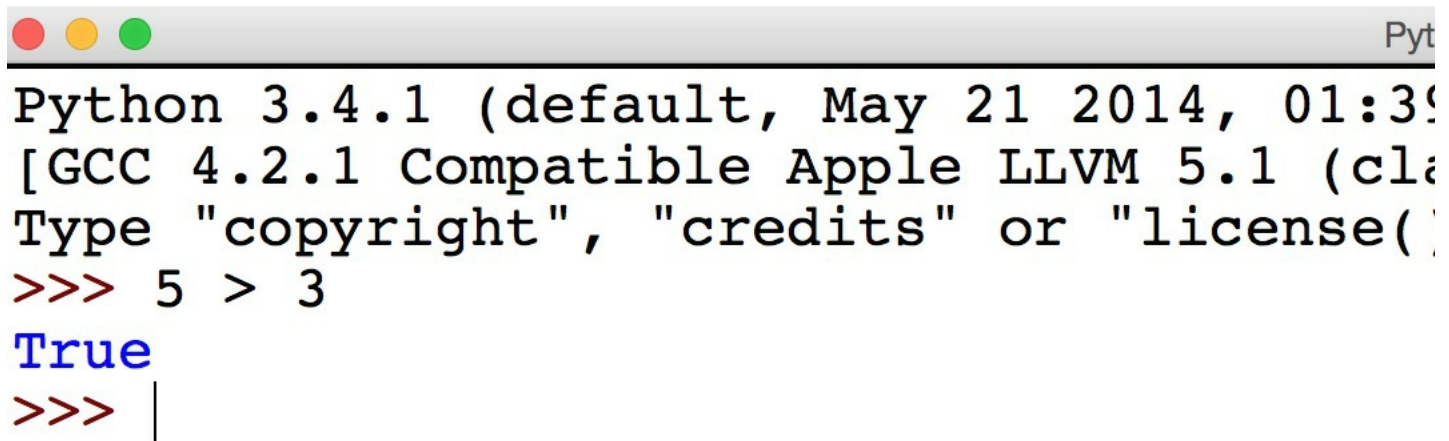# Reading and Research - Selection Statements

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	*greater than*	True
4 >= 4	*greater than or equalled to*	True
3 < 2	*less than*	True
7 <= 7	*less than or equalled 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*	False
test_score >= 60	*greater than or equalled to*	False
test_score < 40	*less than*	False
50 <= test_score	*less than or equalled 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	True
False and False	False
True or True	True
True or False	True
False or True	False
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	The second number is true
or	The first number 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:

<div align="center">**answer**</div>

 If you got greater than or equalled to 50 you have passed but less than 50 is a 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.
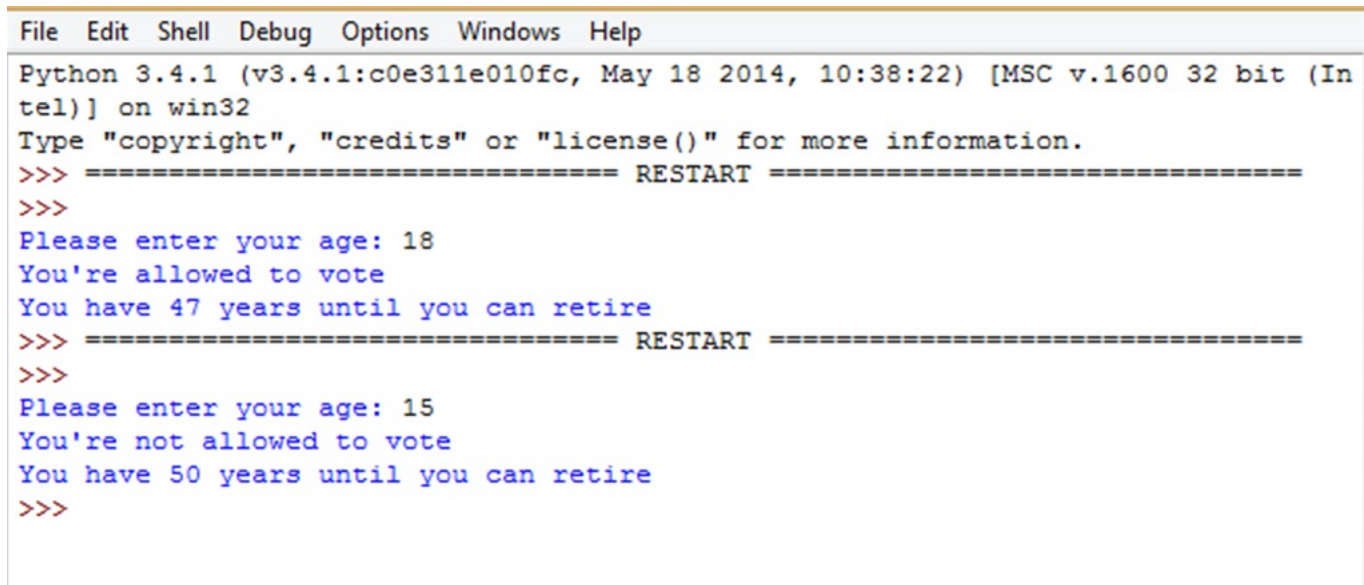
```
#task 5.1

#Kavilan Naidoo
#25-09-2014
#voting exercise

age=int(input("Please enter your age: "))
if age >= 18:
    print("You're allowed to vote")
if age < 18:
    print ("You're not allowed to vote")


retirement= 65 - age

print("You have {0} years until you can retire".format(retirement))
```

```
File  Edit  Shell  Debug  Options  Windows  Help

Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ============================== RESTART ==================================
>>>
Please enter your age: 18
You're allowed to vote
You have 47 years until you can retire
>>> ============================== RESTART ==================================
>>>
Please enter your age: 15
You're not allowed to vote
You have 50 years until you can retire
>>>
```

```
#task 5.2

#Kavilan Naidoo
#25-09-2014
#Range exercise

number=float(input("Please input a number between 1 and 20: "))
if number <= 20 and number >= 1 :
    print(" Your number is within range")
if number <1:
    print(" Your number is too low")
if number >20:
    print ("You're number is too high")
```

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (I
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ============================== RESTART ==============================
>>>
Please input a number between 1 and 20: 0
 Your number is too low
>>> ============================== RESTART ==============================
>>>
Please input a number between 1 and 20: 15
 Your number is within range
>>> ============================== RESTART ==============================
>>>
Please input a number between 1 and 20: 21
You're number is too high
>>> |
```

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.