Python

Common operators: `and`, `not` and `or`

Testing expressions

You may want to create a more complex expression when testing using `if` or `while`.

Testing expressions

You may want to create a more complex expression when testing using `if` or `while`.

```
age = 23

name = "Jemma"

height = 1.63
```

← Some variables

Testing expressions

You may want to create a more complex expression when testing using `if` or `while`.

```
age = 23
name = "Jemma"
height = 1.63
```

$\leftarrow$ Some variables

What if we want someone with all of these qualities?
What if we want someone who is some of these qualities?

Introducing the `and`, `not` and `or` operators

You may want to create a more complex expression when testing using `if` or `while`.

```
age = 23
name = "Jemma"
height = 1.63
if age == 23:
    print("Correct age")
if name == "Jemma":
    print("Correct name")
if height == 1.63:
    print("Correct height")
```

Introducing the `and`, `not` and `or` operators

You may want to create a more complex expression when testing using `if` or `while`.

```
age = 23
name = "Jemma"
height = 1.63
if age == 23:
    print("Correct age")
if name == "Jemma":
    print("Correct name")
if height == 1.63:
    print("Correct height")
```

Could perform 3 tests to make sure the 3 variables are correct

This seems inefficient

Using `and`

Block executes only if both expressions return `True`

```
age = 23
name = "Jemma"
height = 1.63
if name == "Jemma" and age == 23:
    print("It is Jemma!")
It is Jemma!
```

Using and

and can be chained more than once too:

```
age = 23
name = "Jemma"
height = 1.63
if name == "Jemma" and age >= 20 and height < 2:
   print("It is like Jemma!")
It is like Jemma!
```

Using `not`

`not` will reverse the Boolean result of an expression, we can use it to make blocks that execute only if the expression returns `False` or `None`

```
age = 22
name = "Rachel"
height = 1.65
if not name == "Jemma":
    print("It isn't Jemma!")
It isn't Jemma!
```

Using `not`

Can be used to see if variable not in a collection:

```
x = 25
if x not in [1, 2, 3]:
    print("Didn't find x in list")
Didn't find x in list
```

Can even write `is not` to compare:

```
if x is not 100:
    print("x is not 100")
x is not 100
```

Using `or`

`or` will return `True` if either or both expressions are `True`:

```
greeting = "Hello"
if greeting == "Hi" or greeting == "Hello":
    print("Good day")
Good day
```

Can be chained more than once:

```
x = 25
if x < 0 or x > 100 or x == 25:
    print("x is correct")
x is correct
```

Chaining all of these operators

All of these operators can be chained together to create more complex expressions:

```
start = False
end = 55
status = "STARTED"
if status == "STARTED" and (start is not False or end > 0):
    print("Running")
Running
```

You might need brackets (as above) to specify the precedence of evaluation of expressions.