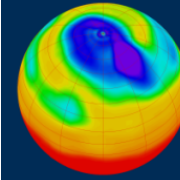




**National Centre for  
Atmospheric Science**

NATURAL ENVIRONMENT RESEARCH COUNCIL



**Centre for Environmental  
Data Analysis**

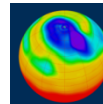
SCIENCE AND TECHNOLOGY FACILITIES COUNCIL  
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Common operators: and, not and or



**National Centre for  
Atmospheric Science**

NATURAL ENVIRONMENT RESEARCH COUNCIL



**Centre for Environmental  
Data Analysis**

SCIENCE AND TECHNOLOGY FACILITIES COUNCIL  
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Introducing the "and", "not" and "or" operators

When testing with an ``if`` or ``while`` statement you often want to test multiple expressions in a single line. You will want to use the following operators to do this:

- ``and``: e.g. ``if temp > 0 and temp < 100:``
  - only execute block if both expressions are True (i.e. temp is in range 1-99 degC).
- ``not``: e.g. ``if not myvar:``
  - only execute block if ``myvar`` does not equate to False/None.
- ``or``: e.g. ``if name == "Mary" or name == "Paul":``
  - execute block if either (any) of the expressions equate to True.

# Using "and"

You can chain any number of expressions together with and operators:

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

if name == "Jemma" and age >= 23 and height >= 1.63:
    print "It is definitely you Jemma!"
```

It is definitely you Jemma!

# Using "not"

You can use not to test for a negative result from an expression and exclusion from a collection:

```
if not name == "Hannah":  
    print "Not allowed in."
```

Not allowed in.

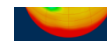
```
x = 25  
if x not in [1, 2, 3]:  
    print "Didn't find x in list."
```

Didn't find x in list.

*In fact, you can even write: is not*

```
if x is not 100: print "NOT 100!"
```

NOT 100!



# Using "or"

You can test if any expression is True with or:

```
greeting = "hello"  
if greeting == "hi" or greeting == "hello":  
    print "Good day to you too."
```

Good day to you too.

You can chain any number of expressions with `or` operators:

```
arg = 5.1  
if type(arg) == str or arg > 5 or arg < -5:  
    print "Arg is good."
```

Arg is good.

# Chaining all of these operators

And you can chain all these operators together if you want or need to:

```
start = None
end = 55
status = "STARTED"

if status == "STARTED" and (start is not None or end > 0):
    print "Running."
```

Running.

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