# Python Notes/Cheat Sheet

**Comments**
# from the hash symbol to the end of a line

**Code blocks**
Delineated by colons and indented code; and not the curly brackets of C, C++ and Java.

```
def is_fish_as_string(argument):
    if argument:
        return 'fish'
    else:
        return 'not fish'
```

**Note**: Four spaces per indentation level is the Python standard. Never use tabs: mixing tabs and spaces produces hard-to-find errors. Set your editor to convert tabs to spaces.

**Line breaks**
Typically, a statement must be on one line. Bracketed code - (), [] or {} - can be split across lines; or (if you must) use a backslash \ at the end of a line to continue a statement on to the next line (but this can result in hard to debug code).

**Naming conventions**

| Style | Use |
|---|---|
| **StudlyCase** | Class names |
| **joined_lower** | Identifiers, functions; and class methods, attributes |
| **_joined_lower** | Internal class attributes |
| **__joined_lower** | Private class attributes<br># this use not recommended |
| **joined_lower**<br>**ALL_CAPS** | Constants |

**Basic object types (not a complete list)**

| Type | Examples | |
|---|---|---|
| **None** | None | # singleton null object |
| **Boolean** | True, False | |
| **integer** | -1, 0, 1, sys.maxint | |
| **long** | 1L, 9787L | # arbitrary length ints |
| **float** | 3.14159265<br>inf, float('inf')<br>-inf<br>nan, float('nan') | <br># infinity<br># neg infinity<br># not a number |
| **complex** | 2+3j | # note use of j |
| **string** | 'I am a string', "me too"<br>'"multi-line string"', """+1"""<br>r'raw string', b'ASCII string'<br>u'unicode string' | |
| **tuple** | empty = ()<br>(1, True, 'dog') | # empty tuple<br># immutable list |
| **list** | empty = []<br>[1, True, 'dog'] | # empty list<br># mutable list |
| **set** | empty = set()<br>set(1, True, 'a') | # the empty set<br># mutable |
| **dictionary** | empty = {}<br>{'a': 'dog', 7: 'seven', True: 1} | # mutable object |
| **file** | f = open('filename', 'rb') | |

**Note**: Python has four numeric types (integer, float, long and complex) and several sequence types including strings, lists, tuples, bytearrays, buffers, and xrange objects.

**Operators**

| Operator | Functionality |
|---|---|
| + | Addition (also string, tuple, list concatenation) |
| - | Subtraction (also set difference) |
| * | Multiplication (also string, tuple, list replication) |
| / | Division |
| % | Modulus (also a string format function, but use deprecated) |
| // | Integer division rounded towards minus infinity |
| ** | Exponentiation |
| =, -=, +=, /=, *=, %=, //=, **= | Assignment operators |
| ==, !=, <, <=, >=, > | Boolean comparisons |
| and, or, not | Boolean operators |
| in, not in | Membership test operators |
| is, is not | Object identity operators |
| \|, ^, &, ~ | Bitwise: or, xor, and, compliment |
| <<, >> | Left and right bit shift |
| ; | Inline statement separator<br># inline statements discouraged |

**Hint**: float('inf') always tests as larger than any number, including integers.

**Modules**
Modules open up a world of Python extensions that can be imported and used. Access to the functions, variables and classes of a module depend on how the module was imported.

| Import method | Access/Use syntax |
|---|---|
| **import math** | math.cos(math.pi/3) |
| **import math as m**<br>**# import using an alias** | m.cos(m.pi/3) |
| **from math import cos,pi**<br>**# only import specifics** | cos(pi/3) |
| **from math import ***<br>**# BADish global import** | log(e) |

Global imports make for unreadable code!!!

**Oft used modules**

| Module | Purpose |
|---|---|
| **datetime**<br>**time** | Date and time functions |
| **math** | Core math functions and the constants pi and e |
| **pickle** | Serialise objects to a file |
| **os**<br>**os.path** | Operating system interfaces |
| **re** | A library of Perl-like regular expression operations |
| **string** | Useful constants and classes |
| **sys** | System parameters and functions |
| **numpy** | Numerical python library |
| **pandas** | R DataFrames for Python |
| **matplotlib** | Plotting/charting for Python |