# PYTHON 2.7 QUICK REFERENCE

**Common operations on numerical types**

`i+j` is the sum of `i` and `j`.

`i-j` is `i` minus `j`.

`i*j` is the product of `i` and `j`.

`i//j` is integer division.

`i/j` is `i` divided by `j`. In Python 2.7, when the operands are both of type `int`, the result is also an `int`, otherwise the result is a `float`.

`i%j` is the remainder when the `int i` is divided by the `int j`.

`i**j` is `i` raised to the power `j`.

`x += y` is equivalent to `x = x + y`. `*=` and `-=` work the same way.

**Comparison and Boolean operators**

`x == y` returns `True` if `x` and `y` are equal.

`x != y` returns `True` if `x` and `y` are not equal.

`<, >, <=, >=` have their usual meanings.

`a and b` is `True` if both `a` and `b` are `True`, and `False` otherwise.

`a or b` is `True` if at least one of `a` or `b` is `True`, and `False` otherwise.

`not a` is `True` if `a` is `False`, and `False` if `a` is `True`.

**Common operations on sequence types**

`seq[i]` returns the ith element in the sequence.

`len(seq)` returns the length of the sequence.

`seq1 + seq2` concatenates the two sequences.

`n*seq` returns a sequence that repeats seq n times.

`seq[start:end]` returns a slice of the sequence.

`e in seq` tests whether `e` is contained in the sequence.

`for e in seq` iterates over the elements of the sequence.

**Common string methods**

`s.count(s1)` counts how many times the string `s1` occurs in `s`.

`s.find(s1)` returns the index of the first occurrence of the substring `s1` in `s`; `-1` if `s1` is not in `s`.

`s.rfind(s1)` same as `find`, but starts from the end of `s`.

`s.index(s1)` same as `find`, but raises an exception if `s1` is not in `s`.

`s.rindex(s1)` same as `index`, but starts from the end of `s`.

`s.lower()` converts all uppercase letters to lowercase.

`s.replace(old, new)` replaces all occurrences of string `old` with string `new`.

`s.rstrip()` removes trailing white space.

`s.split(d)` Splits `s` using `d` as a delimiter. Returns a list of substrings of `s`.

**Common list methods**

`L.append(e)` adds the object `e` to the end of `L`.

`L.count(e)` returns the number of times that `e` occurs in `L`.

`L.insert(i, e)` inserts the object `e` into `L` at index `i`.

`L.extend(L1)` appends the items in list `L1` to the end of `L`.

`L.remove(e)` deletes the first occurrence of `e` from `L`.

`L.index(e)` returns the index of the first occurrence of `e` in `L`.

`L.pop(i)` removes and returns the item at index `i`. Defaults to `-1`.

`L.sort()` has the side effect of sorting the elements of `L`.

`L.reverse()` has the side effect of reversing the order of the elements in `L`.

**Common operations on dictionaries**

`len(d)` returns the number of items in `d`.

`d.keys()` returns a list containing the keys in `d`.

`d.values()` returns a list containing the values in `d`.

`k in d` returns `True` if key `k` is in `d`.

`d[k]` returns the item in `d` with key `k`. Raises `KeyError` if `k` is not in `d`.

`d.get(k, v)` returns `d[k]` if `k` in `d`, and `v` otherwise.

`d[k] = v` associates the value `v` with the key `k`. If there is already a value associated with `k`, that value is replaced.

`del d[k]` removes element with key `k` from `d`. Raises `KeyError` if `k` is not in `d`.

`for k in d` iterates over the keys in `d`.

**Comparison of common non-scalar types**

| Type | Type of Index | Type of element | Examples of literals | Mutable |
|---|---|---|---|---|
| `str` | `int` | characters | `''`, `'a'`, `'abc'` | No |
| `tuple` | `int` | any type | `()`, `(3,)`, `('abc', 4)` | No |
| `list` | `int` | any type | `[]`, `[3]`, `['abc', 4]` | Yes |
| `dict` | Hashable objects | any type | `{}`, `{'a':1}`, `{'a':1, 'b':2.0}` | Yes |

**Common input/output functions**

`raw_input(msg)` prints `msg` and then returns value entered as a string.
`print(s1, …, sn)` prints strings `s1, …, sn` with a space between each.
`open('fileName', 'w')` creates a file for writing.
`open('fileName', 'r')` opens an existing file for reading.
`open('fileName', 'a')` opens an existing file for appending.
`fileHandle.read()` returns a string containing contents of the file.
`fileHandle.readline()` returns the next line in the file.
`fileHandle.readlines()` returns a list containing lines of the file.
`fileHandle.write(s)` write the string `s` to the end of the file.
`fileHandle.writelines(L)` Writes each element of `L` to the file.
`fileHandle.close()` closes the file.