Coding Challenge Notes

Stefan Webb
University of Oxford

1 Observations/unorganized

- *Do the simplest/naive solution first! Then refine.*
- Being super familiar with all the basic operations on the basic types is super important!
- For each language, know by heart: What are the basic types? What are the basic operations on these? What are common gotchas with the basic types? How do the basic types relate to those of other languages?
- I appear to be much slower at implementing recursive solutions! It really helps to draw a picture of the recursion stack here.
- A general problem: generate all subsets of a string/list.
- Learn about the min heap data structure in the Python standard library.

2 Useful functions

- `reversed(s)` reverses an iterable and returns an iterator.
- `sorted(s, key=lambda x:x[0])` can take a key function to do custom sorting.

3 Converting types

3.1 How do I convert a list of chars/strings to string?

```
>>> string = ['a', 'b', 'c']
>>> "".join(strings)
```

3.2 How do I convert a number to a string/vice versa?

```
>>> a = str(6.24)
>>> b = int("8")
>>> c = float("1.25")
```

4 Dictionary manipulation

4.1 How do I get the keys/values of a dictionary?

```
>>> x = {'a': 1, 'b': 2, 'c': 3}
>>> x.keys(), x.values()
```

Get the values with, e.g., `x.values()` not `x.vals()`.

5 List manipulation

5.1 How do I get the max/min element of a list?

```
>>> x = [5,7,12,1,2]
>>> min(x), max(x)
```

5.2 How do I filter a list?

```
>>> [x for x in arr if x < 5]
```

The `if` must come after the `for` in the comprehension, unlike when you do an `if/else` clause.

5.3 How do I remove duplicates?

```
>>> x = [1, 3, 7, 9, 1]
>>> x = list(set(x))
```

This will change the order.

5.4 How do I multiply together all elements?

```
>>> import operator
>>> import functools
>>> x = [1, 3, 7, 9, 1]
>>> total = functools.reduce(operator.mul, x, 1)
```

This is unlike adding together all elements `x.sum()`, which is built into the standard library.

6 String manipulation

6.1 How do I convert from a character to its ASCII/vice versa?

```
>>> n = ord('a')
>>> a = chr(n)
```

6.2 How to I convert a character/string to lower/upper case?

```
>>> 'Abc'.lower()
>>> 'Abc'.upper()
```

6.3 How do I remove/replace a character?

```
>>> '(abc)'.replace('(','').replace(')', ']')
```

6.4 How do I reverse a string?

```
>>> 'abc'[::-1]
```

7 Set manipulation

7.1 How do I add/remove an element from a set?

```
>>> x = {1, 3, 9, 6}
>>> x.remove(3)
>>> x.add(7)
```

These methods have an intuitive name.

7.2 How do I get the maximum/minimum element?

```
>>> max(x), min(x)
```

The same methods work on lists.

7.3 How do I take the union/intersection/difference of sets?

```
>>> x.intersection(y), x & y
>>> x.union(y), x | y
>>> x.difference(y) x - y
```

8 Discriminating between different types of characters

8.1 How do I determine if a char is punctuation?

```
>>> import string
>>> if c is in string.punctuation:
>>>   ...
```

8.2 How do I determine if a char is whitespace?

```
>>> if c.isspace():
>>>   ...
```

8.3 How do I determine if a char is alphabetical/alphanumeric?

```
>>> if c.isalpha():
>>>   ...
>>> if c.isalnum():
>>>   ...
```

8.4 How do I determine if upper or lower case?

```
>>> if c.isupper():
>>>   ...
>>> if c.islower():
>>>   ...
```

9 Common gotchas

- Do not forget to return the value from a function. This is a common mistake I have made!
- Think whether the type is string or number when dealing with digits. Another common mistake!
- `range(n)` goes from 0 to $n-1$, *not* up to n.
- There is neither `++` nor `--` operators in Python. Use `+= 1` and `-= 1` instead.
- The not operator is **`not`**, not !
- To increment a dictionary value that may not have a corresponding key:

```
>>> x.setdefault('x', 0) += 1
>>> x['x'] += 1
```

- You cannot overwrite the value of a **`for`** loop inside the loop body. Instead you should use **`while`** for this idiom:

```
    >>> i = 0
    >>> while i < 10:
    >>>   ...
    >>>   if cond:
    >>>     i += 5
    >>>   i += 1
```

- A slice is not iterable. Instead should do, e.g. **`set`**`(`**`range`**`(2, 8))`.
- A string is immutable, so you cannot assign to one of its elements with e.g. `s[2] = 'a'`.
- You add an item to a set with `.add()`, *not* *`.insert()`* *which is used for lists*.
- Do not confuse multiplication `*` and power `**`.