Python Notes

Dictionaries

In [2]:

```
capitals = {"India":"Delhi", "USA":"Washington","UK":"London"}
capitals["India"]
```

Out[2]:

```
'Delhi'
```

In [3]:

```
"USA" in capitals
```

Out[3]:

```
True
```

In [5]:

```
"Delhi" in capitals
```

Out[5]:

```
False
```

In [4]:

```
"Germany" in capitals
```

Out[4]:

```
False
```

In [6]:

```
capitals["France"] = "Paris"
print(capitals)
```

```
{'UK': 'London', 'USA': 'Washington', 'France': 'Paris', 'India': 'Delhi'}
```

In [7]:

```
list(capitals.keys())
```

Out[7]:

```
['UK', 'USA', 'France', 'India']
```

In [8]:

```
list(capitals.values())
```

Out[8]:

```
['London', 'Washington', 'Paris', 'Delhi']
```

zip()

In [1]:

```
a = [2, 3, 5, 7]
b = ["a", "b", "c", "d"]
ab = []
for i in range(len(a)):
    ab.append((a[i], b[i]))
print(ab)
for x in ab:
    print(x)
```

```
[(2, 'a'), (3, 'b'), (5, 'c'), (7, 'd')]
(2, 'a')
(3, 'b')
(5, 'c')
(7, 'd')
```

In [2]:

```
a = [2, 3, 5, 7]
b = ["a", "b", "c", "d"]
print(list(zip(a,b)))
for x in zip(a,b):
    print(x)
```

```
[(2, 'a'), (3, 'b'), (5, 'c'), (7, 'd')]
(2, 'a')
(3, 'b')
(5, 'c')
(7, 'd')
```

Comparing the two code snippets, you should get a good idea of what zip does. Think of the everyday zipper in a dress. It takes two rows of single teeth, and produces one row of two interlocked teeth. Similarly zip takes two iterables and produces one iterable combining the elements of the original iterables in order

list comprehension

In [3]:

```
a = [2, 3, 5, 7]
asq = []
for x in a:
    asq.append(x * x)
print(asq)
```

```
[4, 9, 25, 49]
```

In [4]:

```
a = [2, 3, 5, 7]
asq = [x * x for x in a]
print(asq)
```

```
[4, 9, 25, 49]
```

[1, 9, 25, 49]

In [5]:

```
vowels = ['a', 'e', 'i', 'o', 'u']
VOWELS = []
for v in vowels:
    VOWELS.append(v.upper())
print(VOWELS)
```

```
['A', 'E', 'I', 'O', 'U']
```

In [6]:

```
vowels = ['a','e', 'i', 'o', 'u']
VOWELS = [ch.upper() for ch in vowels]
print(VOWELS)
```

```
['A', 'E', 'I', 'O', 'U']
```

Comparing these code snippets, you should get a good idea of list comprehensions. They are a very compact and with a little practice, very easy to read construct for building lists from other lists

Recall the set builder notation from set theory (You can groan here)

$S = \{f(x) \mid x \epsilon A,\ p(x)\}$

Translated to English, this means S is the set of all values obtained by applying f to the elements of A that have property p

The simple list comprehension is the same as the set builder with no p

- In the case of the first example, A is the list [2,3,5,7], f is the square function and S is the list asq
- In the second example, A is the list of lower case vowels, f is the uppercase converter, and S is the uppercase vowels.

In [8]:

```
alphabets = "abcdefghijklmnopqrstuvwxyz"
VOWELS = [ch.upper() for ch in alphabets if ch in ['a','e', 'i', 'o', 'u']]
print(VOWELS)
```

```
['A', 'E', 'I', 'O', 'U']
```

Here we have added property p, as beloging to the list of vowels.

Optional

We can rewrite the distance functions using the zip and list comprehension constructs, as below

In [10]:

```
import math
def diff(x, y):
    return x - y
def square(x):
    return x * x
```

```
def euclidean(a, b):
    return math.sqrt(sum([square(diff(x,y)) for x, y in zip(a, b)]))
def manhattan(a, b):
    return sum([abs(diff(x, y)) for x, y in zip(a, b)])
```