Python 3 Cheat Sheet

code.bo() | www.code.bo
info@code.bo
skype: code.bo

Variables

Integer
```
a = 1
```
Float
```
b = 1.0
```
Complex number
```
c = 1 + 2a
```
String
```
d = 'a'
```
Boolean (True/False)
```
e = False
```
Defining vars, Python 3.5+ supports 'type annotations':
```
{varname}: {type}
{varname}: {type} = {value}
fname: str = "lorenzo"
```

Data Structures

Lists

Creating lists
```
a = [1, 2, 3, 4, 5]
```
Processing in one line
```
vals = [value for value in collection if
condition]
```
This is equivalent to:
```
vals = []
for value in collection:
    if condition:
        vals.append(expression)
```

Tuples

Tuples are immutables and usually contain an heterogeneous sequence

To create tuples place values within parentheses
```
a = (1, 2, 3, 4, 5)
```

Sets

Sets are mutable unordered sequence of unique elements.
To create sets place values within braces
```
a = {1, 2, 3, 4, 5}
```

Dictionaries

Unordered set of key: value pairs.
To create dictionaries place key: value pairs within braces.
```
a = {'first_name': 'Lorenzo', 'age': 30}
```

Strings

A collection of characters
Single quotes:
```
'Hello'
```
Double quotes
```
"World"
```
Multi-line
```
"""
Hello
world
"""
```
Concatenate
```
"Hello" + "world"
```
Multiply
```
>>> "hello" * 8
'hellohellohellohello'
```
Length
```
>>> len("Hello world")
11
```

Format: Access arguments by ordinal position
```
>>> '{0}, {2}, {1}'.format(1, 2, 3)
'1, 3, 2'
```
Format: Implicit positional arguments
```
>>> '{}, {}, {}'.format(1, 2, 3)
'1, 2, 3'
```
Format: Access keyword arguments by name
```
>>> '{val1}, {val2}, {val3}'.format(val1=1,
val2=2, val3=3)
'1, 2, 3'
```

Loops

For

```
for i in range(0, 10):
    print(i)
```
Iterate over an array
```
my_array = [1, 1, 2, 3, 5, 8, 13]
for d in my_array:
    print(d)
```
Iterate over an array and get the index
```
my_array = [1, 1, 2, 3, 5, 8, 13]
for index, d in enumerate(my_array)
    print(index, d)
```

While

```
counter = 0
while counter < 10:
    print(counter):
    counter += 1
```

Functions

Declaring functions

Basic:

```
def name(param, key_parm=default_value):
    ...
    return result
```

*args and **kwargs allow you to pass a variable number of arguments to a function

- *args unamed parameters
- *kwargs named parameters

```
def name(param, key_parm=default_value,
*args, **kwargs):
    ...
    return result
```

Python 3.5+ supports 'type annotations'

```
def add(a: int, b: int) -> int:
    return a + b
```

Exceptions

```
try:
    # normal processing block
    ...
except Exception as e:
    # error processing block
    ...
finally
    # block for final processing in all cases
    ...
```

Classes

Declaring a class

```
class Dog:
     race = 'Dog' # Class Attribute
     # Initializer
     def __init__(self, name):
         self.name = name

     # instance method
     def description(self):
         return "{} is a {}".format(
             self.name, self.race)

courage = Dog("courage")
```

Heritage

```
class Bulldog(Dog):
     race = 'bulldog'

monster = Bulldog("Monster")
```

Working with Files

Open files

```
f = open(file, mode='r', encoding=None)
```

file: path to the file

mode:

```
  'r' Read
  'w' Write
  'x' Exclusive creation, fails if the file
already exists
  'a' Append
  'b' binary mode
  '+' open a disk file for updating (reading
and writing)
```

encoding: is the name of the encoding used to decode or encode the file

Read

read entire file

```
f.read()
```

read line by line

```
f.readline()
```

read all lines and returns a list of lines

```
f.readlines()
```

Write

Need to be open in writing mode:

```
f = open('test.txt', 'w')
```

write content

```
f.write("hello world")
```

write lines

```
f.writelines(["hello", "world"])
```

Always close the file

```
f.close()
```

With statement

to read:

```
with open('test.txt', 'r') as f:
    content = f.read()
```

to write:

```
with open('test.txt', 'w') as f:
    f.write("Hello world")
```