COMP 1012 – Cheatsheet

Ranges

```
range(x)       # range from 0 up to and not including x
range(x, y)    # range from x up to  and not including y
range(x, y, z) # range from x up to and not including y, counting up by z
```

Lists

```
list(collection) # convert sequence to list
my_list.append(d)# Append data d to the list
sorted(my_list)  # sort my_list (ascending)
my_list.sort()   # sort my_list (ascending)
max(my_list)     # the maximum value in the list
min(my_list)     # the minimum value in the list
my_list.pop(el)  # Remove and return an element from location el
my_list.remove(i)# find and remove the first instance of i
```

Dicts

```
d = dict()
d.keys()     # get a list-like object of all the keys
d.values()   # get a list-like object of all the values
d.items()    # get a list-like object of all the key-value pairs
d.clear()    # delete all key-value pairs
d['key'] = v # set a new key, or overwrite an existing key in the dict
```

Sets

```
s = set()
s = set(iterable)
s.add(item)         # add an item to the set
s.intersection(s2)  # return a set intersection with se
s.difference(s2)    # return a set difference
s.union(s2)         # return a set union
```

Strings

All string functions are *non-destructive*. String functions do not modify the string, but return a transformed copy.

```
my_str = "My mom loves CAKE."
my_str.upper()       # upper-case version
my_str.lower()       # lower-case version
my_str.strip()       # remove whitespace from beginning and end of string
my_str.split()       # split the string on the space character
my_str.index('z')    # find first location of 'z', or error if not in string
my_str.split("-")    # split the string on the - character
my_str.count("m")    # counts how many of a character is in the string
my_str.replace(x, y) # replace all copies of x with y
my_str.format(args)  # format the string (with {})
my_str.capitalize()  # upper-case first character, lower-case everything else
my_str.isnumeric()   # True if all characters are numeric (0 to 9)
my_str.isdecimal()   # True if numeric, but allows one . character
my_str.find("s")     # return the first index of passed string, or -1 if it is not in my_str
```

Format codes

```
7.3f
^ ^^--- floating point (type)
| |
| ----- 3 decimal places (precision)
|
------- display up to 7 characters (width)
```

- `type` may be any of `d` (integer), `f` (floating point), or `s` (string)
- `width` may be 0 or omitted entirely
- `precision` may be 0 or omitted entirely

File handling

```
f = open("aFile.txt")   # open a file
f.read()                # read an entire file
f.readline()            # read a single line from the file
f.readlines()           # read an entire file into a list of lines
f.close()               # close the file
for line in f:          # read a file line by line
f.write("something")    # write data to a file
```

Python Random

```
import random
random.seed(n)          # Seed the random number generator
random.random()         # A single floating random number [0,1)
random.randint(x, y)    # A single integer random number [x,y]
random.choice(list_of_options)
                        # Choose randomly from the given list
```

Numpy

```
import numpy as np
np.array([1,2,3])        # Create np array with values from list
np.linspace(x, y, z)     # Return z evenly spaced numbers over a specified interval x-y
np.arange(x, y, z)       # Count from x to y by z
np.random.seed(n)        # Seed numpy's random number generator
np.random.random(n)
np.random.random(size=n) # Create array size n of random data with data [0, 1)
np.random.random((n, m))
np.random.random(size=(n,m))  # Create 2D array size n by m of random data with data [0, 1)
np.random.randint(x,y,n)
np.random.randint(x,y,size=n) # Create a random array from x to y (not inclusive) of size n
np.random.randint(x,y,(n,m))
np.random.randint(x,y,size=(n,m)) # Create a random 2D array from x to y (not inclusive) of size n by m
np.random.choice(list_of_options, n)
np.random.choice(list_of_options, size=n)
                         # Create an array size n filled with random data
                         # Chosen from the passed list
np.zeros(n)              # Create an array of n zeros
np.zeros( (n, m) )       # Create a 2D array of zeros, size n by m
np.ones(n)               # Create an array of n ones
np.ones( (n, m) )        # Create an array of n ones, size n by m
a = np.random.randint(0, 10, 10)
a.sum()   # get the sum of the data in the array
a.prod()  # get the product of the data in the array
a.mean()  # get the mean of the data in the array
a.std()   # get the standard deviation of the data in the array
a.abs()   # return an array with the absolute value of the data in a
```

Matplotlib

```
import matplotlib.pyplot as plt
plt.scatter(x, y, c) # plot a scatter plot of x vs y, coloured with c
plt.line(x, y, c)    # plot a line plot of x vs y, coloured with c
plt.plot(x, y, c)    # plot a line (or scatter) of x vs y, coloured with c

# plot a circle
circle = plt.Circle( (0,0), 3)  # Circle at (0,0), with radius 3
plt.gca().add_patch(circle)     # Plot circle on current plot

rect = plt.Rectangle( (0,0), 1, 2) # rectangle at (0, 0), width 1, height 2
plt.gca().add_patch(rect)          # plot rectangle on current plot
```

Pandas

```
import pandas as pd
pd.DataFrame({"column1":data, "column2":data2})
pd.read_csv("filename.csv")
pd.read_excel("filename.xls", sheet="sheetname")
```