Introduction to

with Application to Bioinformatics

- Day 3

Tuples (Q 1&2)

1. Which of the following variables are of the type tuple?

```
a = (1, 2, 3, 4)
a = ([1, 2], 'a', 'b')
```

2. What is the difference between a tuple and a list?

A tuple is immutable while a list is mutable

In [ ]:
```
myTuple    = (1, 2, 3)
myList     = [1, 2 ,3]
```

In [ ]:
```
myList[2]  = 4
myList
```

In [ ]:
```
myTuple[2] = 4
```

Is it true that we can never modify the co
tuple?

In [ ]:
```
myTuple = [1, 2, [1,2,3]]
print(myTuple)
myTuple[2][2] = 4
print(myTuple)
```

Is it true that we can never modify the co tuple?

In [ ]:
```
myTuple = [1, 2, [1,2,3]]
print(myTuple)
myTuple[2][2] = 4
print(myTuple)
```

- The immutability of tuples in Python means that **the struc cannot be changed**, you cannot add, remove, or replace e
- However, if a tuple contains mutable objects like lists, dicti the contents of those mutable objects can still be changed

Functions and methods (Q 4&

4. What are the following examples of?

```
len([1, 2, 3, 4])
print("my text")
```

Functions

5. What are the following examples of?

```
"my\ttext".split("\t")
[1, 2, 3].pop()
```

Methods

In [ ]:
```
myList = [1, 2, 3.5, 5 ,6.2]
round(sum(myList)/len(myList),1)
```

In [ ]:

```
my_list   = ['I','know','Python']
my_string =' '.join(my_list).upper()
print(my_string)
```

In [ ]:

```
# Code Snippet for Finding the Movie with the High
# Note that this is just one of the solutions
with open('../downloads/250.imdb', 'r') as fh:  #
    movieList = [] # create an empty list to start
    highestRating = -1.0 # Create a variable to st
                         # and initialize it with
    for line in fh:     # iterate over the file
        if not line.startswith('#'):
            # split the line separated by '|' into
            cols = line.strip().split('|')
            # extract rating and movie title from
            rating = float(cols[1].strip())
            title = cols[6].strip()
            # votes = int(cols[0].strip())
            # year = int(cols[2])
            movieList.append((rating, title))
            if rating > highestRating:
                highestRating = rating
    print("Movie(s) with highest rating " + str(hi
    for i in range(len(movieList)):
        if movieList[i][0] == highestRating:
            print(movieList[i][1])
```

In [ ]:

```
mySet = {"1", "2", "3", "4", "5"}
for e in mySet:
    print(e)
```

In [ ]:

```
mySet = {"1", "1", "2", "2", "3"}
print(mySet)
```

In [ ]:

```
mySet = {1, "tga", (3, 4), 5.6, False}
print(mySet)
```

Set can only have hashable elements

In [ ]:
```
mySet = {1, "tga", (3, 4), 5.6, False}
print(mySet)
```

In [ ]:
```
mySet = {1, "tga", [3, 4], 5.6, False}
```

Set can only have hashable elements

In [ ]:
```
mySet = {1, "tga", (3, 4), 5.6, False}
print(mySet)
```

In [ ]:
```
mySet = {1, "tga", [3, 4], 5.6, False}
```

In [ ]:
```
mySet = {1, "tga", (3, 4, [1, 2]), 5.6, False}
```

Set can only have hashable elements

In [ ]:
```
mySet = {1, "tga", (3, 4), 5.6, False}
print(mySet)
```

In [ ]:
```
mySet = {1, "tga", [3, 4], 5.6, False}
```

In [ ]:
```
mySet = {1, "tga", (3, 4, [1, 2]), 5.6, False}
```

Although tuples are immutable, but when it contains mutable hashable. Be careful!

In [ ]:

```
# Add elements to a set
myset = set()
myset.add(1)
myset.add(100)
myset.add(100)
print(myset)
```

Basic operations on `set`

In [ ]:
```
# Add elements to a set
myset = set()
myset.add(1)
myset.add(100)
myset.add(100)
print(myset)
```

In [ ]:
```
# get the number of elements of a set
len(myset)
```

Basic operations on `set`

In [ ]:
```
# Add elements to a set
myset = set()
myset.add(1)
myset.add(100)
myset.add(100)
print(myset)
```

In [ ]:
```
# get the number of elements of a set
len(myset)
```

In [ ]:
```
# membership checking
1 not in myset
```

Basic operations on `set`

In [ ]:
```
# Add elements to a set
myset = set()
myset.add(1)
myset.add(100)
myset.add(100)
print(myset)
```

In [ ]:
```
# get the number of elements of a set
len(myset)
```

In [ ]:
```
# membership checking
1 not in myset
```

Learn more on https://www.w3schools.com/python/python_set

In [ ]:

```
myDict = {'drama': 4,
          'thriller': 2,
          'romance': 5}
myDict
```

In [ ]:
```
myDict = {'drama': 4,
          'thriller': 2,
          'romance': 5}
myDict
```

In [ ]:
```
len(myDict)
```

In [ ]:

```
myDict = {'drama': 182,
          'war': 30,
          'adventure': 55,
          'comedy': 46,
          'family': 24,
          'animation': 17,
          'biography': 25}
```

- How many genres are in this dictionary?
- How many movies are in the `comedy` genre?
- You're not interested in biographies, delete this entry
- You're interested in fantasy; add that we have `29` movies to this dictionary.
- Which genres are listed in this dictionary after the change?
- You remembered another comedy movie; increase the num `comedy` genre by one.

Note: let take a look at the following code, it calculate the average
the genre 'drama', How if we also want to calculate the average d
'horror', we would need to either copy the code or using a loop.

and add the complicated code under the loop.

Is there any better solutions? This is how functions are introduced

In [ ]:

```
genre = "drama"
average = sum(genreDict[genre])/len(genreDict[genr
hours   = int(average/3600)
minutes = (average - (3600*hours))/60
reformattedTime = str(hours)+'h'+str(round(minutes
print('The average length for movies in genre '+ g
      ' is '+ reformattedTime)
```

In [ ]:
```
for genre in ['drama', 'horror', 'comedy']:
    average = sum(genreDict[genre])/len(genreDict
    hours   = int(average/3600)
    minutes = (average - (3600*hours))/60
    reformattedTime = str(hours)+'h'+str(round(mi
    print('The average length for movies in genre
          ' is '+ reformattedTime)
```

In [ ]:
```
for genre in ['drama', 'horror', 'comedy']:
    print('The average length for movies in genre
          ' is '+ formatSec(genre, genreDict))
```

Scope

- Local variables - Variables within functions
- Global variables - Variables outside of functions

In [ ]:
```
WEIGHT = 5
def addWeight(value):
    return value * WEIGHT
print(addWeight(4))
```

Scope

- Local variables - Variables within functions
- Global variables - Variables outside of functions

In [ ]:
```
WEIGHT = 5
def addWeight(value):
    return value * WEIGHT
print(addWeight(4))
```

In [ ]:
```
WEIGHT = 5
def changeWeight():
    WEIGHT = 10
    return None
print(WEIGHT)
```

Scope

- Local variables - Variables within functions
- Global variables - Variables outside of functions

In [ ]:
```
WEIGHT = 5
def addWeight(value):
    return value * WEIGHT
print(addWeight(4))
```

In [ ]:
```
WEIGHT = 5
def changeWeight():
    WEIGHT = 10
    return None
print(WEIGHT)
```

We will talk more about the scope of variables tomorrow

Example:

1. Create a file called myFunctions.py, located in the same fo
2. Put a function called `formatSec()` in the file
3. Start writing your code in a separate file and `import` the

In [ ]:

```
from myFunctions import formatSec

seconds = 32154

formatSec(seconds)
```

Session 4

```
- Lecture: Pass arguments from command line using `sys.argv` and string formattin
- Ex4: IMDb exercise - functions and `sys.argv`
- PyQuiz 3.2
```

```
fh  = open('../files/250.imdb', 'r', encoding = 'utf-8')
out = open('../files/imdb_copy.txt', 'w', encoding = 'utf-8

for line in fh:
    out.write(line)

fh.close()
out.close()
```

```python
import sys

if len(sys.argv) == 3:
    fh  = open(sys.argv[1], 'r', encoding = 'utf-8')
    out = open(sys.argv[2], 'w', encoding = 'utf-8')

    for line in fh:
        out.write(line)

    fh.close()
    out.close()

else:
    print('Arguments should be input file name and output fi
```

Run with:

```
$ python3 copy_file.py 250.imdb imdb_co
```

Formatting

Format text for printing or for writing to file.

What we have been doing so far:

In [ ]:
```
title  = 'Toy Story'
rating = 10
print('The result is: ' + title + ' with rating: '
```

In [ ]:

```
title  = 'Toy Story'
rating = 10
print(f'The result is: {title} with rating: {ratin
```

In [ ]:

```
title  = 'Toy Story'
rating = 10
print('The result is: {} with rating: {}'.format(t
```

In [ ]:

```
title  = 'Toy Story'
rating = 10
print('The result is: %s with rating: %s' % (title
```

Answer - Example

```python
import sys

"""
Script that reformats an imdb file and writes it to another file
"""


def FormatSec(seconds):    # formats seconds to hours and minutes
    hours     = seconds/3600
    minutes   = (seconds - (3600*int(hours)))/60
    return str(int(hours))+'h'+str(round(minutes))+'min'

def FormatMovie(movie):    # returns a string with the correct format for writing to file
    formMovie = str(movie[0])+'\t'+movie[1]+' ('+str(movie[2])+') ['+movie[3]+']\n'
    return formMovie


if len(sys.argv) == 3:
    fh        = open(sys.argv[1], 'r', encoding = 'utf-8')
    genreDict = {}

    for line in fh:
        if not line.startswith('#'):
            cols   = line.strip().split('|')
            rating = float(cols[1].strip())
            year   = int(cols[2].strip())
            length = int(cols[3].strip())
            movie  = cols[6].strip()
            genre  = cols[5].strip()
            glist  = genre.split(',')
            for entry in glist:
                if not entry.lower() in genreDict:              # if genre in dictionary, ad
                    genreDict[entry.lower()] = []
                genreDict[entry.lower()].append([rating, movie, year, FormatSec(length)])
    fh.close()


    out = open(sys.argv[2], 'w', encoding = 'utf-8')
    for genre in genreDict:
        out.write('> '+genre.capitalize()+'\n')
        for movie in genreDict[genre]:
            out.write(FormatMovie(movie))
    out.close()

else:
    print('Number of arguments does not match')
```