15-110 Refresher Session : Week 11

No Calculators, only Brains !!

1. **Warm up : Function hoard**
   Think and fill the table with at least 5 methods/functions you use on strings and dictionaries. Discuss if you have written the same with your partner.

Strings	Dictionaries

2. **Act like a Computer**

```python
def mystery(d, S):
    S = S + "s"
    d['Animal'] = S
    print(S)

S = "Stag"
d = {'Animal': 'S'}
mystery(d, S)
print(mystery(d, S))
print(d, S)
```

   What would the output of the code snippet above?

3. **Act like a programmer**
   Fatma is reading a story book and comes across a word “melancholy”. She is determined to count how many times the word melancholy appears in the book. As she read, she encountered the word many times, but she was too involved in the book that she forgot to keep track of the word. Help Fatma by implementing a function called `countWords(s)` which takes in a passage string `s` and

   (a) prints a dictionary of all the words that appears in the book and the number of times it appears

   (b) returns a list of all the words and its occurences as tuples in alphabetical order

   We can assume that the words are all in lower case, single spaced with no other symbols. For example :

```
countWords("i will call this a sentence a sentence i will call") returns
{'i': 2, 'will': 2, 'call': 2, 'this': 1, 'a': 2, 'sentence': 2}
[('a', 2), ('call', 2), ('i', 2), ('sentence', 2), ('this', 1), ('will', 2)]
```

4. **Act like a Computer**

```
def mystery():
    numbers = {12 : "twelve" , "one" : "thirteen"}
    for i in range(12, 14) :
        print(numbers.get(i , "one"))
```

What does `mystery()` print ?

5. **Act like a Programmer**
Morty has been assigned to develop a comprehensive movie recommendation system that considers various details about movies, including their title, release year, genre, and the actors involved. The information is stored in a dictionary named `movies`, where each movie is represented by a title and associated details.

```
movies = {
    'Movie1':
            {'year': 2020,
            'genre': 'Action',
            'actors': ['Actor1', 'Actor2', 'Actor3']},
    'Movie2':
            {'year': 2019,
            'genre': 'Comedy',
            'actors': ['Actor2', 'Actor4', 'Actor5']},
    'Movie3':
            {'year': 2021,
            'genre': 'Drama',
            'actors': ['Actor1', 'Actor3', 'Actor6']},
     #... more movies }
```

(a) Your first task is to implement a function that allows the addition of new movies to the recommendation system. This function, named `add_movie (title, year, genre, actors)` takes the title, release year, genre, and a list of actors of the new movie as input. It should then update the movies dictionary to include this new movie.

(b) The next operation involves providing users with the ability to explore movies based on their genres. Implement the function `get_movies_by_genre(genre)` that takes a genre as input and returns a list of movies falling under that specific genre.