Assignment 6

Ryan Waterman

4/30/25

Question 1

1. import the random library.
2. Use `random.seed(10)` to initialize a pseudorandom number generator.
3. Create a list of 50 random integers from 0 to 15. Call this list `int_list`.
4. Print the 10th and 30th elements of the list.

You will need to use list comprehension to do this. The syntax for list comprehension is: <new_list> = `[<expression> for <item> in <iterable>]`. For this question your expression will be a randint generator from the random library and your iterable will be `range()`. Researh the documentation on how to use both functions.

In [5]:
```
import random

random.seed(10)

int_list = [random.randint(0,15) for i in range(50)]

print(int_list[9], int_list[29])
```

```
1 7
```

Question 2

1. import the string library.
2. Create the string `az_upper` using `string.ascii_uppercase`. This is a single string of uppercase letters
3. Create a list of each individual letter from the string. To do this you will need to iterate over the string and append each letter to the an empty list. Call this list `az_list`.
4. Print the list.

You will need to use a for-loop for this. The syntax for this for-loop should be:

`for i in string>:

`

In [11]:
```
import string

# First, doing this the way the question asks
az_upper = string.ascii_uppercase

az_list = []
for i in az_upper:
    az_list.append(i)

print(az_list)

# I prefer to do this operation with list comprehension, though
az_upper = [letter for letter in string.ascii_uppercase]

print(az_upper)
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

Question 3

1. Create a set from 1 to 5. Call this `set_1`.
2. Create a set from int_list. Call this `set_2`.
3. Create a set by finding the `symmetric_difference()` of `set_1` and `set_2`. Call this `set_3`.
4. What is the length of all three sets?

In [7]:
```
set_1 = set(range(1, 6))
set_2 = set(int_list)
set_3 = set.symmetric_difference(set_1, set_2)

print(set_1, set_2, set_3)
print(len(set_1), len(set_2), len(set_3))
```

```
{1, 2, 3, 4, 5} {0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15} {0, 3, 6, 7, 8,
9, 10, 11, 12, 13, 14, 15}
5 15 12
```

Question 4

1. Import default dict and set the default value to 'Not Present'. Call this dict_1.
2. Add `int_list`, `set_2`, and `set_3` to `dict_1` using the object names as the key names.

3. Create a new dictionary, `dict_2`, using curly bracket notation with `set_1` and `az_list` as the keys and values.

4. Invoke the default value of `dict_1` by trying to access the key `az_list`. Create a new set named `set_4` from the value of `dict_1['az_list']`. What is the lenght of the difference between `dict_2['az_list']` and `set_4'?

5. Update `dict_2` with `dict_1`. Print the value of the key `az_list` from `dict_2`. What happened?

In [16]:
```
from collections import defaultdict

dict_1 = defaultdict(lambda: 'Not Present')

dict_1["int_list"] = int_list
dict_1["set_2"] = set_2
dict_1["set_3"] = set_3
```

In [17]:
```
dict_2 = {
    "set_1":set_1,
    "az_list": az_list
    }
```

In [19]:
```
dict_1['az_list']
```

Out[19]: 'Not Present'

In [27]:
```
set_4 = set(dict_1['az_list'])
print(set_4)
```

```
{'P', 'n', 'r', ' ', 't', 'e', 'N', 'o', 's'}
```

In [29]:
```
set_5 = set(dict_2['az_list'])
```

In [32]:
```
print(len(set_5 - set_4))
```

```
24
```

In [33]:
```
dict_2.update(dict_1)
```

In [34]:
```
dict_2['az_list']
```

Out[34]: 'Not Present'

The value of dict_2's az_list key is now overwritten with the value from dict_1

Question 5

A palindrome is a word, phrase, or sequence that is the same spelled forward as it is backwards. Write a function using a for-loop to determine if a string is a palindrome. Your

function should only have one argument.

In [41]:
```
def is_palindrome(word):
    for i in range(len(word)):
        if i+1>len(word)/2:
            return True
        if word[i] != word[-(i+1)]:
            return False

def is_palindrome_no_loop(word):
    return True if word==''.join(reversed(word)) else False
```

In [42]:
```
word_list = [    #Courtesy of ChatGPT
    "racecar",   # Palindrome
    "table",     # Non-palindrome
    "rotor",     # Palindrome
    "lamp",      # Non-palindrome
    "repaper",   # Palindrome
    "level",     # Palindrome
    "mirror",    # Non-palindrome
    "kayak",     # Palindrome
    "river",     # Non-palindrome
    "civic",     # Palindrome
    "phone",     # Non-palindrome
    "rotator",   # Palindrome
    "paper",     # Non-palindrome
    "stats",     # Palindrome
    "chair",     # Non-palindrome
    "madam",     # Palindrome
    "camera",    # Non-palindrome
    "refer",     # Palindrome
    "gadget",    # Non-palindrome
    "noon",      # Palindrome
    "window",    # Non-palindrome
    "mom",       # Palindrome
    "textbook",  # Non-palindrome
    "pop",       # Palindrome
    "random"     # Non-palindrome
]

for word in word_list:
    print(is_palindrome(word), is_palindrome_no_loop(word))
```

```
True True
False False
True True
False False
True True
True True
False False
True True
False False
True True
False False
True True
False False
True True
False False
True True
False False
True True
False False
True True
False False
True True
False False
True True
False False
```

Question 6

Two Sum - Write a function named two_sum() Given a vector of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.Use defaultdict and hash maps/tables to complete this problem.

Example 1: Input: nums = [2,7,11,15], target = 9 Output: [0,1] Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2: Input: nums = [3,2,4], target = 6 Output: [1,2]

Example 3: Input: nums = [3,3], target = 6 Output: [0,1]

Constraints: 2 <= nums.length <= 104 –109 <= nums[i] <= 109 –109 <= target <= 109 Only one valid answer exists.

In [80]:
```
"""
I was actually able to come up with this on my own without referencing the answer..
then thirty minutes to debug that I was checking for 'if compliment in nums' instea
"""

def two_sum(vect, target):
    nums = {}
    for i,number in enumerate(vect):
```

```
        compliment = target-number
        if number in nums:
            return [nums[number], i]
        else:
            nums[compliment]=i
    return None
```

In [81]:
```
list_1 = [2,7,11,15]
target_1 = 9

print(two_sum(list_1, target_1))
```

```
[0, 1]
```

In [82]:
```
list_2 = [3,2,4]
target_2 = 6

print(two_sum(list_2, target_2))
```

```
[1, 2]
```

In [88]:
```
list_3 = [3,3]
target_3 = 6

print(two_sum(list_3, target_3))
```

```
[0, 1]
```

The following is code and test cases courtesy of ChatGPT. I had to tweak a two of the test cases (97,99) to remove multiple solutions, but other than that, it worked well!

In [90]:
```
import json

with open("data/two_sum_test_cases.json") as f:
    cases = json.load(f)

for k, case in enumerate(cases, 1):
    got = two_sum(case["nums"], case["target"])
    assert got == case["expected"], f"❌ case {k} failed: expected {case['expected
print("✅ all test cases pass!")
```

```
✅ all test cases pass!
```