Assignment 6

August 11, 2023

1 Assignment 6

1.0.1 Adeline Casali

1.0.2 August 9, 2023

1.1 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: = `[<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.

[1]:
```
# (1) Import random library
import random

# (2) Initialize pseudorandom number generator
random.seed(10)

# (3) Create a list of 50 random integers
int_list = [random.randint(0, 15) for _ in range(50)]

# (4) Print the 10th and 30th elements
print("10th Element:", int_list[9])
print("30th Element:", int_list[29])
```

```
10th Element: 1
30th Element: 7
```

1.2 Question 2

1) import the string library.
2) Create the string `az_upper` using `string.ascii_uppercase`. This is a single string of upper-case 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>:    <list operation>`

[2]:
```python
# (1) Import string library
import string

# (2) Create the string containing uppercase letters
az_upper = string.ascii_uppercase

# (3) Initialize an empty list and iterate over the string and append each␣
↪letter to the list
az_list = []

for i in az_upper:
    az_list.append(i)

# (4) Print the list
print(az_list)
```

```
['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']
```

1.3 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?

[3]:
```python
# (1) Create set_1 from 1 to 5
set_1 = set(range(1, 6))

# (2) Create set_2 from int_list
set_2 = set(int_list)

# (3) Create set_3 by finding the symmetric difference of set_1 and set_2
set_3 = set_1.symmetric_difference(set_2)

# (4) Print the lengths of the sets
print("Length of set_1:", len(set_1))
print("Length of set_2:", len(set_2))
print("Length of set_3:", len(set_3))
```

```
Length of set_1: 5
Length of set_2: 15
Length of set_3: 12
```

1.4 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?

[10]:
```python
# (1) Import default dict and create defaultdict with default value 'Not Present'
from collections import defaultdict

dict_1 = defaultdict(lambda: 'Not Present')

# (2) Add int_list, set_2, and set_3 to dict_1
dict_1['int_list'] = int_list
dict_1['set_2'] = set_2
dict_1['set_3'] = set_3

# (3) Create dict_2 using curly bracket notation
dict_2 = {'set_1': set_1, 'az_list': set(az_list)}

# (4) Access non-existent key 'az_list' in dict_1, create set_4 and calculate␣
 ↪length difference
dict_1['az_list']

set_4 = set(dict_1['az_list'])

length_difference = len(dict_2['az_list'].difference(set_4))
print("Length difference:", length_difference)

# (5) Update dict_2 with dict_1 and print the value of 'az_list' key from dict_2
dict_2.update(dict_1)

print("Value of 'az_list' in dict_2:", dict_2['az_list'])

# In this example, the value of az_list in dict_2 was changed based on the value␣
 ↪of it in dict_1, because dict_2
```

```
# was updated based on dict_1. Dict_1 did not contain az_list, but we called it␣
↪and due to the default dict value
# being set to Not Present, that was then the key/value pair. When dict_2 was␣
↪updated based on that, the value
# associated with az_list then changed to Not Present as well.
```

```
Length difference: 24
Value of 'az_list' in dict_2: Not Present
```