ChandonnetModule06Lab01

December 4, 2022

1 Assignment 6

1.0.1 Ray Chandonnet

1.0.2 12/4/2022

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]:
```
# Problem 1
import random as r
r.seed(10)
int_list = [r.randint(0,15) for counter in range(50)]
print("The entire list is :",int_list)
print("The 10th element is : ", int_list[9])
print("The 29th element is : ",int_list[29])
# Note that since Python uses zero-based indexing, the 10th element is at index␣
↪#9 and the 30th element is at index #29
```

```
The entire list is : [1, 13, 15, 0, 6, 14, 15, 8, 5, 1, 15, 10, 2, 7, 11, 1, 13,
4, 11, 12, 13, 9, 8, 14, 5, 9, 11, 4, 14, 7, 14, 12, 1, 0, 7, 4, 6, 9, 11, 7,
10, 14, 13, 15, 2, 10, 5, 7, 13, 7]
The 10th element is :  1
The 29th element is :  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 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>:    <list operation>
```

[2]:
```
# Problem 2
import string as s
az_upper = s.ascii_uppercase
print(az_upper)
az_list=[]
for counter in az_upper:
    az_list.append(counter)
print(az_list)
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
['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]:
```
set_1=set(range(1,6))
print("Set_1:",set_1)
set_2=set(int_list)
print("Set_2:",set_2)
set_3 = set_1.symmetric_difference(set_2)
print("Set_3:",set_3)
len_1=len(set_1)
len_2=len(set_2)
len_3=len(set_3)
print("Set 1 has",len_1,"elements")
print("Set 2 has",len_2,"elements")
print("Set 3, the symmetric_difference set, has",len_3,"elements")
print("For my own comprehension - that is the number of unique elements that do␣
 ↪NOT appear in both sets (opposite of")
print("intersection)")
set_4 = set_1.intersection(set_2)
len_4=len(set_4)
print("Set 4, the intersection set, has",len_4,"elements")
tot_elements=len_1+len_2
```

```
print("There are a total of",tot_elements,"elements in the two sets")
print(len_4,"of those elements appear both times = ",len_4*2,"instances")
print(len_3,"of those elements only appear once = ",len_3,"instances")
print("Added together, they equate to the total # of elements:",␣
 ↪tot_elements,"=",len_4*2+len_3)
```

```
Set_1: {1, 2, 3, 4, 5}
Set_2: {0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
Set_3: {0, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
Set 1 has 5 elements
Set 2 has 15 elements
Set 3, the symmetric_difference set, has 12 elements
For my own comprehension - that is the number of unique elements that do NOT
appear in both sets (opposite of
intersection)
Set 4, the intersection set, has 4 elements
There are a total of 20 elements in the two sets
4 of those elements appear both times =  8 instances
12 of those elements only appear once =  12 instances
Added together, they equate to the total # of elements: 20 = 20
```

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?

[5]:
```
#Problem 4
from collections import defaultdict
def def_value():
    return "Not Present"
dict_1 = defaultdict(def_value)
dict_1["int_list"]=int_list
dict_1["set_2"]=set_2
dict_1["set_3"]=set_3
print("Dictionary 1:",dict_1)
print(dict_1['az_list'])
dict_2=defaultdict(def_value)
dict_2 = {"set_1":set_1,"az_list":az_list}
print("Dictionary 2:",dict_2)
```

```
set_4=set(dict_1['az_list'])
print("Set 4:", set_4)
print("Since 'az_list' is not a key in dict_1, this just took the default value␣
 ↪'Not Present' and turned it into a set by letter!")
set_diff=set(dict_2['az_list']).symmetric_difference(set_4)
print("The length of the difference between dict_2['az_list'] and dict_2 is",␣
 ↪len(set_diff),"elements")
dict_2.update(dict_1)
print(dict_2['az_list'])
print("It appears that when the dictionary was updated with the contents of␣
 ↪dict_1, the value for key 'az_list',")
print("which used to be our list of capital letters, was replaced with 'Not␣
 ↪Present' which is the default value.")
print("Even after researching, it is unclear to me why!")
```

```
Dictionary 1: defaultdict(<function def_value at 0x7fd758a2c820>, {'int_list':
[1, 13, 15, 0, 6, 14, 15, 8, 5, 1, 15, 10, 2, 7, 11, 1, 13, 4, 11, 12, 13, 9, 8,
14, 5, 9, 11, 4, 14, 7, 14, 12, 1, 0, 7, 4, 6, 9, 11, 7, 10, 14, 13, 15, 2, 10,
5, 7, 13, 7], 'set_2': {0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
'set_3': {0, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}})
Not Present
Dictionary 2: {'set_1': {1, 2, 3, 4, 5}, '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']}
Set 4: {'t', 'r', ' ', 'n', 'o', 'e', 's', 'P', 'N'}
Since 'az_list' is not a key in dict_1, this just took the default value 'Not
Present' and turned it into a set by letter!
The length of the difference between dict_2['az_list'] and dict_2 is 31 elements
Not Present
It appears that when the dictionary was updated with the contents of dict_1, the
value for key 'az_list',
which used to be our list of capital letters, was replaced with 'Not Present'
which is the default value.
Even after researching, it is unclear to me why!
```

```
[ ]:
```