py101_lesson_3_easy_2

December 4, 2024

1 Practice Problems: Easy 2

1.1 Question 1

Write two distinct ways of reversing the list without mutating the original list.

[12]:
```python
numbers = [1, 2, 3, 4, 5]

reversed_numbers = numbers.copy()
reversed_numbers.reverse()
print(reversed_numbers)
print(numbers)

new_list = numbers[::-1]

print(new_list)
print(numbers)
```

```
[5, 4, 3, 2, 1]
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
[1, 2, 3, 4, 5]
```

1.2 Question 2

Given a number and a list, determine whether the number is included in the list.

[13]:
```python
numbers = [1, 2, 3, 4, 5, 15, 16, 17, 95, 96, 99]

number1 = 8  # False (not in numbers)
number2 = 95 # True (in numbers)

print((number1 in numbers) == True)
print((number2 in numbers) == True)
```

```
False
True
```

1.3 Question 3

Programmatically determine whether 42 lies between 10 and 100, inclusive. Do the same for the values 100 and 101.

```
[15]: a = 42
      b = 100
      c = 101

      lst = [a, b, c]

      for item in lst:
          print(f'Is {item} between 10 and 100, inclusive? {item in range(10, 101)}.')
```

```
Is 42 between 10 and 100, inclusive? True.
Is 100 between 10 and 100, inclusive? True.
Is 101 between 10 and 100, inclusive? False.
```

1.4 Question 4

Given a list of numbers [1, 2, 3, 4, 5], mutate the list by removing the number at index 2, so that the list becomes [1, 2, 4, 5].

```
[16]: lst = [1, 2, 3, 4, 5]
      del lst[2]

      print(lst)
```

```
[1, 2, 4, 5]
```

1.5 Question 5

How would you verify whether the data structures assigned to the variables numbers and table are of type list?

```
[17]: numbers = [1, 2, 3, 4]
      table = {'field1': 1, 'field2': 2, 'field3': 3, 'field4': 4}

      print(type(numbers) == list)
      print(type(table) == list)
```

```
True
False
```

```
[19]: # This is the preferred solution, as type(x) may have issues:

      print(isinstance(numbers, list) == True)
      print(isinstance(table, list) == True)
```

```
True
False
```

1.6 Question 6

Back in the stone age (before CSS), we used spaces to align things on the screen. If we have a 40-character wide table of Flintstone family members, how can we center the following title above the table with spaces?

```
[24]: title = "Flintstone Family Members"
      centered_title = '|' + ((20 - (len(title) // 2 )) * " ") + title + ((20 -␣
       ↪(len(title) // 2)) * " ") + '|'
      print(centered_title)
```

```
|        Flintstone Family Members        |
```

```
[25]: # Hahaha - there's a str.center()
      centered_title = title.center(40)
      print(f'| {centered_title} |')
```

```
|        Flintstone Family Members         |
```

1.7 Question 7

Write a one-liner to count the number of lower-case t characters in each of the following strings:

```
[34]: statement1 = "The Flintstones Rock!"
      statement2 = "Easy come, easy go."

      print((statement1 + statement2).count('t'))
      print(statement1.count('t'))
      print(statement2.count('t'))
```

```
2
2
0
```

1.8 Question 8

Determine whether the following dictionary of people and their age contains an entry for 'Spot':

```
[43]: ages = {'Herman': 32, 'Lily': 30, 'Grandpa': 5843, 'Eddie': 10}
      if "Spot" in ages:
          print(ages["Spot"])
```

1.9 Question 9

We have most of the Munster family in our ages dictionary; Add entries for Marilyn and Spot to the dictionary:

```
[44]: ages = {'Herman': 32, 'Lily': 30, 'Grandpa': 5843, 'Eddie': 10}

      additional_ages = {'Marilyn': 22, 'Spot': 237}
      for k in additional_ages:
```

```python
    ages[k] = additional_ages[k]

print(ages)
# Suggested solution is better than mine, haha:
ages.update(additional_ages)
```

```
{'Herman': 32, 'Lily': 30, 'Grandpa': 5843, 'Eddie': 10, 'Marilyn': 22, 'Spot':
237}
```