Programming with Python Feb 17-18 2025

Note: Brief Summary of contents discussed.

Practice

Example 1: Write a function that takes a string as a parameter and returns a string with every successive repetitive character replaced with a star (*). For example 'balloon' is returned as 'bal*o*n'.

```
def update_str(st):
    my_st = st[0]
    for i in range(1, len(st)):
        if st[i] == st[i-1]:
            my_st += '*'
        else:
            my_st += st[i]
    return my_st

print(replace_with_star('balloon'))
```

Example 2: Write a function that takes a sentence as input parameter and returns the number of words in the sentence.

```
def count_words(sentence):
    words = sentence.split()#Words is a list
    return len(words)

print(count_words('This is a sentence'))
```

Note: So far, we have handled data in simple forms such as integers, booleans, and strings. Moving forward, we will discuss more complex data structures: lists, tuples, sets, and dictionaries.

List: A list in Python is a comma separated, **ordered**, **mutable** collection of elements. Lists allow storing multiple data types in a single variable.

Creating a list:

```
fruits = ["apple", "banana", "cherry"]
numbers = [1, 2, 3, 4, 5]
mixed = ["hello", 10, 3.5, True] #heterogeneous data
type
```

Aliasing: different names for same list

```
colors = [‘red’, ‘green’, blue’]
print(id(colors)
my_colors = colors
print(id(my_colors)
```

#id: returns reference to the list object (similar to addressing)

Length of list: len()

```
colors = [‘red’, ‘green’, blue’]
len(colors)
```

2-d list

```
arr2d = [[‘Web designing’, 101], [‘Python’, 102], [‘Algebra’, 103]]
```

List indexing/slicing: Similar to strings

```
colors = [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’]
colors[0]
colors[-1]
colors[:3]
colors[4:]
colors[2:5]
colors[::-1]
colors[:15]+colors[15:]
```

list function

```
>>> l1 = 'python'
>>> l2 = list(l1)
>>> l2
['p', 'y', 't', 'h', 'o', 'n']
>>> ''.join(l2)
'python'
```

list/string membership

```
>>> 'a' in 'aeiou'
True
>>> 'a' in ['a', 'e', 'i']
True
>>> 'b' in ['a', 'e', 'i']
False
```

```
>>> 'b' in 'aeio'
False
```

List functions:

append(x)	Adds x to the end of the list	lst = [1, 2]; lst.append(3)	[1, 2, 3]

Exercises:

Give output:

1.
```
subject = 'computer'
subject = list(subject)
ch = subject[0]
for i in range(0, len(subject) - 1):
    subject[i] = subject[i+1]
subject[len(subject) - 1] = ch
print(''.join(subject))
```

2.
```
x = [1, 2, 4, 6, 9, 10, 14, 15, 17]
for i in range(0, len(x)):
    if (x[i]%2 == 0):
        x[i] = 4*i
    elif (x[i] % 3 == 0):
        x[i] = 9*i;
    else:
        x[i] *= 3
print(x)
```

3.
```
def func():
    l1 = list()
    l2 = list()
    for i in range(0, 5):
        l1.append(i)
        l2.append(i+3)
        l1, l2 = l2, l1
```

```
print(l1)
print(l2)
```

4. List `lst1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]`, create a new list containing only even numbers.

5. Write a function that takes a list of values as input parameters and returns another list without any duplicates.

6. Input: `l1 = [234, 57, 1, 3467, 32]`
   Output: `l2 = [9, 12, 1, 20, 5]` #Output list is the sum of digit in the input list.

7. Write a function that takes a list of numbers as input from the user and produces the corresponding cumulative list where each element in the list at index i is the sum of elements at index `j<=i`.

8. Given two lists `lst1` and `lst2`. Write a function (use loops) to merge the two lists without duplicates.

9. List `lst1 = [2, 4, 7, 2, 1, 1 , 3, 5, 9, 5, 9]`. Count the frequency of each element.

Note:

1. Explore the following website
   https://pythontutor.com/python-compiler.html#mode=edit

2. eval: a security risk (Someone in the class pointed this out)
   Yes, since eval can execute a valid python expression, it can be used by a hacker to run harmful code as input.

```
user_input = "os.system('rm -rf /')"  # Dangerous
command!
eval(user_input)  # This can delete files
```