Lab 5 – Mutable Data Types

Deadline: Sunday, Oct. 12th 2025 11:59 pm
Summission

- `lab5_<your_name>_<SID>_Q1.py`
- `lab5_<your_name>_<SID>_Q2.py`
- `lab5_<your_name>_<SID>_Q3.py`
- `lab5_<your_name>_<SID>_Q4.py`

1. (6 Pts) Lab CheckOff

Q1. (2 Pts) Learning Practice to use list

Fill the tables in the Word file of "Lab5_Mutable_Types"

Q2. (2 Points)

Write a Python function that takes two tuples of numbers, and returns dot product of the two sequences of numbers. Specifications are listed below

```
def dot_product(tA, tB):
    """
    tA and tB: a tuple of numbers. Assumes tA and tB are the same length.
    Returns a tuple where the:
    * first element is the length of one of the tuples
    * second element is the sum of the pairwise products of tA and tB
    """
    # function body

# Examples:
tA = (1, 2, 3)
tB = (4, 5, 6)
print(dot_product(tA, tB)) # prints (3,32)
```

Q3. (2 Points)

Write a Python function that takes a list of integers as input, and return the list of all the pairs that add up to a given target value. Each item can only used once in the result.
For example:

```
def target_sum_pair(list, target):
    # function body

print(target_sum_pair([1, 2, 3, 4, 5], 6))      # should print [[1,5], [2, 4]]
print(target_sum_pair([1, 1, 2, 3, 4, 5], 6))        # should print [[1,5], [2, 4]]
print(target_sum_pair([1, 1, 2, 3, 4, 5, 5], 6))     # should print [[1,5], [1,5], [2, 4]]
print(target_sum_pair([1, 2, 6, 5, 3], 10))     # should print []
```

2. Program Assignment – Submit on Canvas

Q1. (2 Points) Write a function that merges two sorted lists into one sorted list.

Note: you can't use the built-in *sort()* function

For example:

```
def merge_sorted_lists(list1, list2):
     # function body

print(merge_sorted_lists([1,3,5], [2,4,6]))      # should print [1,2,3,4,5,6]
print(merge_sorted_lists([1,3,5], []))           # should print [1,3,5]
print(merge_sorted_lists([10], [2,4,6]))         # should print [2,4,6,10]
```

Q2. (2 Points)

The built-in function zip takes two list arguments, and returns a list that contains zipped pairs of corresponding elements in the input arguments.

```
>>> list(zip([1, 2, 3], ['A', 'B', 'C']))
[(1, 'A'), (2, 'B'), (3, 'C')]
```

Write a function, unzip, that takes a list of pairs and performs the reverse function to zip. If the input pairs is not list, return None. **Note: you can't use the built-in *zip()* function**

For example:

```
def unzip(pairs):
     # function body

print(unzip([(1, 'A'), (2, 'B')])          # should print ([1, 2], ['A', 'B'])
print(unzip([(1, 'A'), (2, )]))            # should print ([1, 2], ['A'])
print(unzip(""))                           # should print None
```

Q3. (2 Points) Write a function that finds the length of the longest substring without repeating characters.

For example:

```
def longest_substring(text):
     # function body

print(longest_substring("abcabdef"))              # should print 6
print(longest_substring("aaaaa"))                 # should print 1
print(longest_substring("pwwkew"))                # should print 3
```

Q4. (2 Points) Word Frequency

Write a program to calculate the word frequency of a text file, and print out the top 3 most frequent words with their frequencies.

Note:

- Please use the given text file ***"article.txt"*** (uploaded on Canvas) in your program.
- The text file only includes letters and spaces
- Convert all words to lower case when calculating the frequency
- If multiple words have same frequency, you can display anyone of them.

For example:

```
def top_3_frequency_words(f):
     # function body


# assume the "sample.txt" includes "hi you How are you doing today How do you do"
# should ouput
# you 3
# do 2
# how 2
print(top_3_frequency_words("sample.txt"))
```