Practice makes your coding skill perfect. So, whenever you are given exercises, please try and practice them.

Remember that computer coding skills need a systematic approach.

Therefore, every week's learning is built on the current week and previous week learning.

CSE5APG

Week05: Lab05

Python Data Structures: Lists, Tuples, Sets and Dictionaries

OBJECTIVES

- Implement Input, Process and Output steps.
- Learn how to use Lists.
- Learn how to use Tuples.
- Learn how to use Sets.
- Learn how to use Files and Dictionaries

---

1. Review Questions on Lists

a. Write a statement to create (i) a list of even integers from 2 to 100, inclusive; (ii) a list of one hundred 0.

b. What is the difference between methods **append** and **insert**? Give an example of each.

c. What is the difference between methods **remove** and **pop**? Give an example of each? What precaution should we take in calling these methods?

d. Given a list of numbers, write a loop to print out all the even numbers of the list. **Can we do this without using the element indexes**?

e. Given a list of numbers, write a loop to double the value of each element. **Can we do this without using the element indexes**?

f. Given a list of numbers, write a statement to sort the elements of the list by their absolute values.

2. Reversing List

Without using the available list method **reverse**, write a code that takes a list and then **reverses** all list elements. Use an additional list.

3. Reversing List (Without additional list and reverse method)

Repeat the previous question. This time perform the task without creating an additional list.

4. Moving Even Numbers to the Front

Write a code that creates a list of integers and then moves all the even numbers to the front, otherwise preserving the order of the elements.

For example, if we pass to the function list **a** below

a = [1, 2, 3, 4, 5, 6, 7, 8]

then after the function call, list **a** becomes

a = [2, 4, 6, 8, 1, 2, 3, 4]

Use additional lists.

5. Moving Even Numbers to the Front (Without extra lists)

Repeat the previous question. This time perform the task without creating additional lists.

6. The Sieve of Eratosthenes

The Sieve of Eratosthenes is an algorithm, known to the ancient Greek, that computes all prime numbers up to **N**. It is carried out as follows:

- Make the list of numbers from 2 to N.
- Then remove the multiples of 2 (but not 2), multiples of 3 (but not 3), and so on, up to multiples of $\sqrt{N}$
- The remaining numbers are prime numbers from 2 up to N.

Write a program that creates a list of numbers from 2 to 100. Then remove the multiples of 2 (but not 2), multiples of 3 (but not 3), and so on, up to multiples of 10 (square root of 100). Print the remaining numbers. They are all the prime numbers up to 100.

For example, given a number ***n***, print all primes smaller than or equal to ***n***. It is also given that ***n*** is a small number. If ***n*** = 10, the output should be "2, 3, 5, 7". If ***n*** is 20, the output should be "2, 3, 5, 7, 11, 13, 17, 19".

I. Make a list of all numbers from 2 to n.
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ..., n]

II. Starting from 2, delete all its multiples in the list, except itself.
[2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~, 11, ~~12~~, 13, ~~14~~, 15, ~~16~~, 17, ~~18~~, 19, ~~20~~, …, n]

III. Repeat step **II** till the square root of ***n***.
For 3 – [2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~, …, n]
For 5 – [2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~,…, n]

Till sqrt(n). The remaining list only contains prime numbers.

7. Neighbour-Watch Sort

The following is a simple algorithm that sorts a list of N elements:

```
Repeat this N-1 times:
    For elements from index 0 to N-2:
        Watch the neighbour on the right
        If the element is bigger than its neighbour: Swap
            them
```

The algorithm is easy to understand and implement, and we are much less likely to make mistakes in implementing it. Write a code to sorts the list by the above neighbour-watch algorithm.

---

8. Review Questions on Tuples

a. Write an expression to create a tuple that has (i) two elements, (ii) one element, and (iii) no elements.

b. We have tuple **t1 = (1, 2, 3, 4, 5)**. Write a statement that uses **t1** to create tuple **t2** which is equal to **(1, 2, 3, 4, 5, 6, 7)**.

c. Given a variable **date,** which is a tuple with three integers representing the day, month, and year, in that order. Then, write an expression to get the month of the date.

d. Given the variable **date = (24, 4, 2020)**. Write a single assignment statement to extract the day, the month and the year.

e. Method **append** can be used to add an element to a list. Can we use this method to add an element to a tuple?

9. Pet Shop

(From P4E)

A pet shop wants to give a discount to its clients if they buy one or more pets and at least five other items. The discount is equal to 20% of the cost of other items, but not the pets.

Write a program that takes a list of tuples

(Price, isPet, quantity)

The sale of a pet that costs $550 is represented by a tuple

(550, True, 1)

The sale of an item that costs $10.50 (before discount), and for which the customer buys 5 of them, is represented by the tuple **(10.50, False, 5)**

The program calculates and displays the cost before discount, discount amount and the cost after discount.

10. **Fibonacci Sequence**

The Fibonacci Sequence is a series of numbers: 1, 1, 2, 3, 5, 8, 13, 21. The next number is found by adding up the two numbers before it:

- the 2 is found by adding the two numbers before it (1+1),
- the 3 is found by adding the two numbers before it (1+2), •
  the 5 is (2+3),
- and so on!

Recall that the Fibonacci sequence is as shown below:

1, 1, 2, 3, 5, 8, 13, 21, ...

Let f(n) denote the value of the n^{th} term. Then we have

f(1) = 1 f(2) = 1 f(3) = f(1) + f(2) = 2 f(4) = f(2) +

f(3), . . .

In this question, we use this kind of tuple to keep information about a particular term **n**:

(n, f(n-1), f(n))

For example, here are the tuples for *n* = 2 and 3:

(2, 1, 1)

(3, 1, 2)

Write a code that takes an integer ***n*** and then uses the tuples given above, computes and returns the Fibonacci value for ***n*** (display the Fibonacci sequence up to ***n***th term). You can assume $\mathbf{n_1}$=0 and $\mathbf{n_2}$=1.

As suggested, we use tuple like this

(n, fib(n-1), fib(n))

To calculate the n-th term (suppose N > 2)

1. Start with tuple f = (2, 1, 1) .

2. Then repeat several times, and each time update tuple ***f*** .

---

11. Review Questions on Sets

a. Write an expression to create a set of (i) two elements, (ii) one element, and (iii) no elements.

b. Given a list **L** with duplicate elements, write an expression to get a version of L with all the duplicate elements removed.

c. Which method or operator can be used to (1) add an element to a set? (ii) remove an element from a set? (iii) determine if an element is in a set?

d. Given two sets S**1** and **S2**, using methods **union**, **intersection** and **difference**, write an expression to get the set of elements that are in **S1** or **S2** but not both.

```
#Use difference() between set A and set B
A = {10, 20, 30, 40, 80}
B = {100, 30, 80, 40, 60}
print (A.difference(B))
print (B.difference(A))
```

---

12. Review Questions on Dictionaries

a. Write an expression to create a dictionary with (i) two key-value pairs (two elements), (ii) one key-value pair, and (iii) no key-value pairs.

b. How do you add a key-value pair to a dictionary?

c. How do you change the value of a particular key of a dictionary?

d. How do you remove a key-value pair from a dictionary?

e. How do you get the value of a particular key?

f. Given a dictionary **d**, write an expression to get (i) the list of all the keys, (ii) the list of all the values, and (iii) the list of all the tuples representing all the key-value pairs.

g. Given a dictionary **d**, write a loop to display all the keys.

13. Student Marks

Write a program that keeps a dictionary of student **id** and their **marks**. Add codes to:

a. Print all the marks

b. Prompt the user to add a student

c. Prompt the user to remove a student

d. Prompt the user to modify a mark

e. Display the marks ordered by mark in decreasing order and within groups of equal marks, ordered by name.

14. Letters-Words Mapping

Write a program that reads a list of words (you can use words_3000.txt file). Build a dictionary whose keys are lower case letters and whose values are the sets of words in which the letter appears, ignoring cases. For example, if the **key** is letter **c,** the **value** is the set of words such as **cat, Christmas, fetch, occur, etc**.

{**c**: [**cat, Christmas, fetch, occur, ..,etc**]}

The program then asks the user to enter a **word**, for example, **hat**, and it prints out all the words that have all the letters in the word entered by the user: **hat**, **that**, **heat**, **theatre**, and so on.

15. Translating Text Messages

(From P4E)
Instant messaging (IM) and texting on portable devices have resulted in a set of common abbreviations useful for brief messages. However, some individuals may not understand these abbreviations.

Write a program that reads a one-line text message containing common abbreviations and translates the message into English using a set of translations stored in a file. For example, if the user enters the text message
y r u l8

the program should print

why are you late

As a simplification, you can assume that are no punctuation marks.

Proceed as follows:

a. Build a dictionary with abbreviations as keys and associated texts as values. Read the provided text file **abbreviations.txt**, each line of which contains an abbreviation and the associated text.

The following code segment reads the file and store each line of the text file as a string in the list **lines**:

```
fname = "abbreviations.txt"
file = open (fname, "r")
lines = file.readlines()
```

(You should display the list **lines** to see what a string of this list looks like.). The loop below shows how we can extract the keys and values to build our dictionary for line in lines:

```
temp = line.split(":")
key = temp[0].strip()
value = temp[1].strip()
print(key, "->", value)
```

Write a code to build a dictionary with abbreviations as keys and the associated texts as value.

b. Translate a message.

Split the message into "words". If a "word" is in the dictionary, replace it with the associated text. Otherwise, simply copy the "word" to the translation.