INFS1201-Computer Program

Lab 05

The lab consists of three parts:

* Part 1 consists of fundamental exercises: you need to complete, understand, and submit ALL these exercises during lab time.
* Part 2 consists of four intermediate exercises. Choose only TWO to complete and submit before leaving this lab session.
* Part 3 consists of further exercises to think about if you are done with the first two parts. These exercises are optional but are highly recommended for students who find the first two parts easy; no submission required!

Always start your code by the following comments:

##

#firstName lastName - StudentID

#Week 05 lab 04

#Exercise XXX

# Part I

## A group of fruits Description automatically generated with low confidenceExercise 1

Start your program by defining a list of your favorite fruits (for example banana, strawberry, kiwi, orange).

Then display the list of favorite fruits separated by the \ symbol.

Note your solution should use a loop as it should work for any list of fruits, regardless of the number of fruits it contains.

Sample run:

Your favorite fruits are: banana\strawberry\kiwi\orange

## A picture containing text Description automatically generatedExercise 2

Write a program that asks the user for ten integers and store them in a list.

Display the obtained list.

Then display them nicely with a sentence for each element you entered.

You can re-use exercise 6 from week 3 to create a small function for the ordinal part.

See the below sample run:

Enter an integer: 10

Enter an integer: 23

Enter an integer: 52

Enter an integer: 63

Enter an integer: 84

Enter an integer: 65

Enter an integer: 12

Enter an integer: 10

Enter an integer: 96

Enter an integer: 5

We just built the following list: [10, 23, 52, 63, 84, 65, 12, 10, 96, 5]

The 1st element in your list is: 10

The 2nd element in your list is: 23

The 3rd element in your list is: 52

The 4th element in your list is: 63

The 5th element in your list is: 84

The 6th element in your list is: 65

The 7th element in your list is: 12

The 8th element in your list is: 10

The 9th element in your list is: 96

The 10th element in your list is: 5

## Exercise 3

Write a function wordParity with a unique parameter text (of type string), that returns True if the number of characters of text is even and False otherwise. Add a small docstring to describe your function.

In the main program, ask the user how many words he would like to enter, then ask him to enter these words and finally display the number of words with even parity.

See sample run below:

Enter the number of words to be checked: 4

Enter word 1 :This

Enter word 2 :lab

Enter word 3 :is

Enter word 4 :easy

You entered 3 words of even length

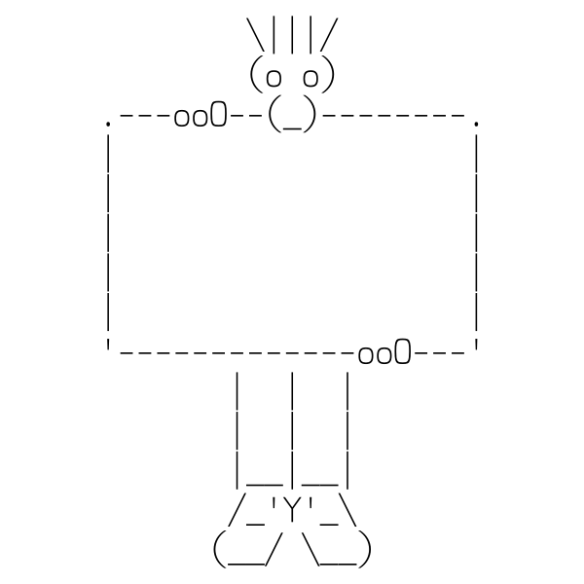
>>> help(wordParity)

Help on function wordParity in module \_\_main\_\_:

wordParity(text)

returns True if the length of text is even, False otherwise

## Exercise 4

Similar to the previous solid, filled-in ASCII art shapes of last week and the week before, this time you have to draw only the border of a rectangle. The + plus character is used for the corners, the - dash character for horizontal lines, and the | pipe character for vertical lines.

Write a drawBorder(width, height) function that draws the border of a rectangle with the given integer sizes, assuming both dimensions are 2 or more. Don’t forget to add an appropriate docstring. See below sample runs.

>>> drawBorder(1,5)

Incorrect parameters, cannot draw such a rectangle!

>>> drawBorder(4,-4)

Incorrect parameters, cannot draw such a rectangle!

>>> drawBorder(2,2)

++

++

>>> drawBorder(10,5)

+--------+

| |

| |

| |

+--------+

>>> help(drawBorder)

Help on function drawBorder in module \_\_main\_\_:

drawBorder(width, height)

Draws the borders of a rectangle using ASCII art.

# Part II

## Exercise 5

Diagram

Description automatically generatedWrite a median function that has a numbers parameter. numbers is a list of numbers. This function returns the statistical median of the numbers list. The median of an odd-length list is the number in the middlemost number when the list is in sorted order. If the list has an even length, the median is the average of the two middlemost numbers when the list is in sorted order. You can use Python’s built-in sort() method to sort the numbers list.

Passing an empty list to median should cause it to return None.

Don’t forget to add an appropriate docstring.

To test your function try the below assert statements (copy paste them to your code in the main function).

assert median([]) == None, "Empty list test failed!"

assert median([1, 2, 3]) == 2, "Test 2 failed."

assert median([3, 7, 10, 4, 1, 9, 6, 5, 2, 8]) == 5.5, "Test 3 failed."

assert median([3, 7, 10, 4, 1, 9, 6, 2, 8]) == 6, "Test 4 failed."

import random

random.seed(42)

testData = [3, 7, 10, 4, 1, 9, 6, 2, 8]

for i in range(1000):

random.shuffle(testData)

assert median(testData) == 6

print("All tests succesfull!")

## Exercise 6

Write a closestAverage function that has a numbers parameter. numbers is a list of numbers. This function returns the element closest to the average of the numbers list. You can use Python’s built-in sum() methodif you wish.

Passing an empty list to closestAverage should cause it to return None.

Don’t forget to add an appropriate docstring.

To test your function try the below assert statements (copy paste them to your code in the main function).

assert closestAverage([]) == None, "Empty list test failed!"

assert closestAverage([1, 2, 3]) == 2, "Test 2 failed."

assert closestAverage([3, 7, 10, 4, 1, 9, 5, 6, 2, 8]) == 5, "Test 3 failed."

assert closestAverage([3, 7, 10, 4, 1, 9, 6, 2, 8]) == 6, "Test 4 failed."

import random

random.seed(42)

testData = [3, 7, 10, 4, 1, 9, 6, 2, 8]

for i in range(1000):

random.shuffle(testData)

assert closestAverage(testData) == 6

print("All tests succesfull!")

## Exercise 7

The goal of this exercise is to write a function shuffleString with a parameter text, that returns a permutation of text, which means the letters of text are returned in a different order. The built-in function random.shuffle() does something similar, but you basically have to write your own version of it, so you are forbidden to use random.shuffle. Other functions from the random module would be useful of course, such as random.randrange() or random.randint().

Optional: It could be easier to start by a function that shuffles a list, then use the function shuffleList to shuffle the indices of the string.

Don’t forget docstrings!

To check that your function works well, run it several times on the same input and check that you get different results each time, as per the below sample runs:

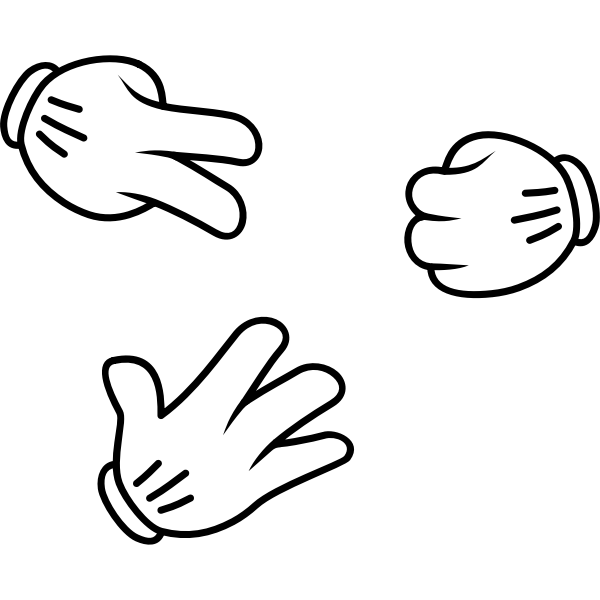
>>> shuffleString("programming")

'miargpmngor'

>>> shuffleString("programming")

'rimmogprgan'

## Exercise 8

The goal of this exercise is to program a game of rock paper scissor. You should ask the user how many games they want to play and, in each game, they should choose rock or paper or scissors, the computer will randomly choose rock, paper or scissors and then decide who wins or if it is a tie. At the end the program should display the total score (in each game, if the computer wins, you score 0, if it is a tie, you score 1, if you win you score 3).

You need to validate the input as well.

If you know how to do it as in the sample run below you can solve it directly, otherwise you can look at a step-by-step approach on the other lab document.

Sample run:

How many games of Rock, Paper Scissors do you want to play? Five

This not a valid number of games!

Number of games should be a positive integer

How many games of Rock, Paper Scissors do you want to play? 5.5

This not a valid number of games!

Number of games should be a positive integer

How many games of Rock, Paper Scissors do you want to play? 5

\*\*\* Game 1 \*\*\*

What do you choose (rock or paper or scissors)?r

Choose rock or paper or scissors!

What do you choose (rock or paper or scissors)?1

Choose rock or paper or scissors!

What do you choose (rock or paper or scissors)?rock

Computer chose scissors

You win this round!

\*\*\* Game 2 \*\*\*

What do you choose (rock or paper or scissors)?rock

Computer chose scissors

You win this round!

\*\*\* Game 3 \*\*\*

What do you choose (rock or paper or scissors)?rock

Computer chose paper

Computer wins this round!

\*\*\* Game 4 \*\*\*

What do you choose (rock or paper or scissors)?rock

Computer chose rock

This round is a tie...

\*\*\* Game 5 \*\*\*

What do you choose (rock or paper or scissors)?rock

Computer chose paper

Computer wins this round!

End of the game

Your score is 7

# Part III

## Exercise 9

Write a function generatePassword to generate a password with a parameter length.

If the length is invalid or below 8, it should be set to 12 as default length. Besides, the password should contain at least on upper case letter, one lower case letter, one digit and one special character (similar to exercise 6 from week 5). You can use any function from the random module or exercise 7 from this week.

As usual don’t forget docStrings.

You can test your code with the below assert statements (if you are using the same constants as in exercise 6 from week 5):

assert len(generatePassword(7)) == 12

pw = generatePassword(14)

assert len(pw) == 14

hasLowercase = False

hasUppercase = False

hasNumber = False

hasSpecial = False

for character in pw:

if character in LOWER:

hasLowercase = True

if character in UPPER:

hasUppercase = True

if character in DIGITS:

hasNumber = True

if character in SPECIAL:

hasSpecial = True

assert hasLowercase and hasUppercase and hasNumber and hasSpecial

Sample runs to generate password with length from 7 to 16:

invalid length parameter!

Length has been defaulted to 12

password generated with parameter 7 : lt@p~SM9lyb!

password generated with parameter 8 : Zq1YfRj&

password generated with parameter 9 : TSbNhi15\_

password generated with parameter 10 : KYi\*dHb9WD

password generated with parameter 11 : L0kNU+E%I0)

password generated with parameter 12 : m72I7~X^4HR2

password generated with parameter 13 : hV244QA9Z4Pv!

password generated with parameter 14 : (4stA#k9QO\_o1k

password generated with parameter 15 : umz!mYXSy4^dS3a

password generated with parameter 16 : kK5SLd(xKNd%m)+r

## Exercise 10

The game from exercise 8 has a nice extension called “Rock, Paper, Scissors, Lizard, Spock”!

If you want to know the rules of this game, just watch the video:

[](https://www.youtube.com/embed/x5Q6-wMx-K8?start=133&feature=oembed)

Then program this game as in exercise 8…Have fun!

Sample run:

How many games of Rock, Paper, Scissors, Lizard, Spock do you want to play? 5

\*\*\* Game 1 \*\*\*

What do you choose (rock/paper/scissors/lizard/Spock)?spock

Computer chose scissors

You win this round!

\*\*\* Game 2 \*\*\*

What do you choose (rock/paper/scissors/lizard/Spock)?spock

Computer chose rock

You win this round!

\*\*\* Game 3 \*\*\*

What do you choose (rock/paper/scissors/lizard/Spock)?spock

Computer chose spock

This round is a tie...

\*\*\* Game 4 \*\*\*

What do you choose (rock/paper/scissors/lizard/Spock)?spock

Computer chose scissors

You win this round!

\*\*\* Game 5 \*\*\*

What do you choose (rock/paper/scissors/lizard/Spock)?spock

Computer chose paper

Computer wins this round!

End of the game

Your score is 10