Computer Science 1 (../index.html) CPSC 1301K (../../index.html)

Lists Practice Assignments

Overview

Lists are a foundational data structure in computer programming. These assignments will help you practice using lists.

Submission Instructions

For each of the following practice assignments, save your solution in a `.py` file, with the name being `lists` and the number of the assignment. For example, for Lists01: Names, save your solution in a file named `lists01.py`. Then, submit that file the respective assignment on codePost.io (https://codePost.io).
To register for a free account, go to https://codepost.io/signup/join?code=I5039NQNWJ (https://codepost.io/signup/join?code=I5039NQNWJ). Register with your CSU email address. Sometimes it takes more than one try, so please work on this well before the deadline. Additionally, some students have had better luck with using the “Forgot password” link.

Practice Assignments

Lists01: Names

Write a Python program that prompts the user for how many names to store in a `list`. Then, prompt the user that many times, storing each of the user's entries into a `list` named `names`. Display the `names list`. User input in bold

Example 1:

```
How many names do you want to enter? 3
Please enter a name: Pat
Please enter a name: Chris
Please enter a name: Ryan
['Pat', 'Chris', 'Ryan']
```

Example 2:

```
How many names do you want to enter? 7
Please enter a name: Casey
Please enter a name: Riley
Please enter a name: Jessie
Please enter a name: Jackie
```

```
Please enter a name: Jaime
Please enter a name: Kerry
Please enter a name: Jody
['Casey', 'Riley', 'Jessie', 'Jackie', 'Jaime', 'Kerry', 'Jody']
```

Provided code:

```
def main():
  ''' Your solution goes here '''

  print( names )

main()
```

Lists02: Powers of 5

Complete the function `powersOf5()`. Have the function take a parameter *n* and return a list of the first *n* powers of 5. One way to calculate powers of 5 is by starting with 1 and then multiplying the previous number by 5.

Examples:
`powersOf5( 1 )` returns `[1]`
`powersOf5( 2 )` returns `[1, 5]`
`powersOf5( 3 )` returns `[1, 5, 25]`
`powersOf5( 6 )` returns `[1, 5, 25, 125, 625, 3125]`
`powersOf5( 10 )` returns `[1, 5, 25, 125, 625, 3125, 15625, 78125, 390625, 1953125]`

Provided code:

```
def powersOf5():
```

Lists03: Odd and Even

Complete the `oddNumbers()` functions to take an `int` (as a parameter). Return a list of all of the odd numbers between 1 and one less than the parameter.

Also, complete the `evenNumbers()` functions to take an `int` (as a parameter). Return a list of all of the even numbers between 2 and one less than the parameter.

Examples:
`oddNumbers( 9 )` returns `[1, 3, 5, 7]`
`evenNumbers( 7 )` returns `[2, 4, 6]`

Provided code:

```
def oddNumbers():



def evenNumbers():
```

Lists04: Modify a list

Write the `modifyList()` function to delete the first element of `names` and change the last element to Ava.

Examples:
`modifyList( ['Gertrude', 'Sam', 'Ann', 'Joseph'] )` returns `['Sam', 'Ann', 'Ava']`
`modifyList( ['Ulysses', 'Bob', 'Giuseppe'] )` returns `['Bob', 'Ava']`

Provided code:

```
def modifyList( names ):
```

Lists05: Reverse Sort

Complete `reverseSort()` so that it returns a copy of `lst` that is in reverse alphabetical order.

Examples:
`reverseSort( ['Jan', 'Sam', 'Ann', 'Joe', 'Todd'] )` returns `['Todd', 'Sam', 'Joe', 'Jan', 'Ann']`
`reverseSort( ['I', 'Love, 'Python'] )` returns `['Python', 'Love', 'I']`

Provided code:

```
def reverseSort( lst ):
```

Lists06: Min, Max, Sum and Count

Complete the `minMaxSumCount()` function to take a `list` as a parameter and return a different `list` with the following elements (in order):

- the lowest value value
- the largest value,
- the total of all of the values
- the number of elements in the parameter `list`.

Examples:
`minMaxSumCount( [90, 95, 94, 92, 91, 98, 97, 97, 90] )` returns `[90, 98, 844, 9]`
`minMaxSumCount( [95] )` returns `[95, 95, 95, 1]`

Provided code:

```
def minMaxSumCount( ):
```

Lists07: Party Invitation

Write a Python program to display an invitation to each of the people in the `friends list`, using a for loop.

Example:
Casey, please come to my party on Saturday
Riley, please come to my party on Saturday
Jessie, please come to my party on Saturday
Jackie, please come to my party on Saturday
Jaime, please come to my party on Saturday
Kerry, please come to my party on Saturday
Jody, please come to my party on Saturday

Provided code:

```
def main( ):
    # Display an invitation to each friend in the following list
    friends = ['Casey', 'Riley', 'Jessie', 'Jackie', 'Jaime', 'Kerry', 'Jody']



main()
```

Lists08: Extra Credit Total

Complete the `extraCreditTotal()` function to take a `list` of grades as a parameter and return the total number points above 100.

Examples:
`extraCreditTotal( [101, 83, 107, 90] )` returns `8` (because `1 + 0 + 7 + 0` is `8`)
`extraCreditTotal( [89, 113, 95] )` returns `13`
`extraCreditTotal( [89, 73, 96, 97, 99, 100] )` returns `0`

Provided code:

```
def extraCreditTotal( ):



```

Lists09: Shift Grades

Write a Python program that asks the user how many points to shift the grades in the `list` named `grades`. Display each of the values in `grades` (each on its own line), shifted by the value from the user.

Example 1:

```
Please enter the amount to shift the grades: 1
92
91
86
88
94
77
93
```

Example 2:

```
Please enter the amount to shift the grades: 5
96
95
90
92
98
81
97
```

Provided code:

```
def main( ):
    grades = [91, 90, 85, 87, 93, 76, 92]


main()
```

Lists10: Longest Word

Complete the `longestWord()` function to take a `list` as a parameter and return the length of the longest word in that `list`.

Examples:
`longestWord(['Python', 'rocks'])` returns `6`
`longestWord(['Casey', 'Riley', 'Jessie', 'Jackie', 'Jaime', 'Kerry', 'Jody'])` returns `6`
`longestWord(['I', 'a', 'am', 'an', 'as', 'at', 'ax', 'the'])` returns `3`

Provided code:

```
def longestWord( ):
```

Lists11: isScrambled

Complete the `isScrambled()` function to return `True` if `stringA` can be reordered to make `stringB`. Otherwise, return `False`. Ignore spaces and capitalization. You must use a `list` for this assignment.

Examples:
`isScrambled( 'Easy', 'Yase' )` returns `True`
`isScrambled( 'Easy', 'EasyEasy' )` returns `False`
`isScrambled( 'eye', 'eyes' )` returns `False`
`isScrambled( 'abcdefghijklmnopqrstuvwxyz', 'zyxwvutsrqponmlkjihgfedcba' )` returns `True`
`isScrambled( 'Game Test', 'tamegest' )` returns `True`

Provided code:

```
def isScrambled( stringA, stringB ):
```

Lists12: Rows and Columns Sums

Write two functions, `sumRows()` and `sumCols()`, which each takes a two-dimensional `list`. `sumRows()` should return a `list` of totals for each row. `sumCols()` should return a `list` of totals for each column.

Examples:
`exampleA2DList = [[44, 45], [46, 47], [48, 49]]`
`sumRows(exampleA2DList)` returns `[89, 93, 97]` (because 89 = 44 + 45, 93 = 46 + 47, and 98 = 48 + 49)
`sumCols(exampleA2DList)` returns `[138, 141]` (because 138 = 44 + 46 + 48, and 141 = 45 + 47 + 49)

exampleB2DList = [[76, 64, 23, 58], [73, 15, 24, 94], [86, 39, 89, 35], [39, 15, 76, 2]]
`sumRows(exampleB2DList)` returns `[221, 206, 249, 132]`
`sumCols(exampleB2DList)` returns `[274, 133, 212, 189]`

Hints:
You can assume that each row has the same number of elements. How do you iterate through a 2D list? Think about rows and columns, and how they correspond to a 2D list.

Provided code:

```
def sumRows():

def sumCols():
```

Lists13: Multiplication Table

Write a function `makeMultiplicationTable()` that takes two parameters, `m` and `n`, where `m` equals the number of rows and `n` equals the number of columns. This function should return a two-dimensional `list`, where each element is the product of the row index plus 1 and the column index plus 1.

Examples:

```
rows = 3
cols = 4
makeMultiplicationTable( rows, cols ) returns [[1, 2, 3, 4], [2, 4, 6, 8], [3, 6, 9, 12]]

rows = 10
cols = 10
makeMultiplicationTable( rows, cols ) returns
```

Hints:

- Every cell starting at row `r` and column `c`, will be the value of `(r + 1) * (c + 1)`
- Note: recall that range is exclusive and that indexing starts at 0

Provided code:

```
def makeMultiplicationTable():
```


Last Modified: 05/06/2021 09:24:32