Dashboard / My courses / CD19411-PPD-2022 / WEEK 07-Functions / WEEK-07 CODING

Started on	Thursday, 16 May 2024, 12:46 PM
State	Finished
Completed on	Friday, 17 May 2024, 12:28 PM
Time taken	23 hours 42 mins
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	NAVEEN RAJ B 2022-CSD-A

Question **1**

Correct

Mark 1.00 out of 1.00

A list rotation consists of taking the last element and moving it to the front. For instance, if we rotate the list [1,2,3,4,5], we get [5,1,2,3,4]. If we rotate it again, we get [4,5,1,2,3].

Write a Python function rotatelist(l,k) that takes a list l and a positive integer k and returns the list l after k rotations. If k is not positive, your function should return l unchanged. Note that your function should not change l itself, and should return the rotated list.

Here are some examples to show how your function should work.

>>> rotatelist([1,2,3,4,5],1)
[5, 1, 2, 3, 4]

>>> rotatelist([1,2,3,4,5],3)
[3, 4, 5, 1, 2]

>>> rotatelist([1,2,3,4,5],12)
[4, 5, 1, 2, 3]

Answer: (penalty regime: 0 %)

Reset answer

```python
def Reverse(l, start , end):
    while start < end:
        temp = l[start]
        l[start] = l[end]
        l[end] = temp
        start += 1
        end -= 1

def rotatelist(l,k):
    k = k % len(l)
    Reverse(l , 0 , len(l) - 1)
    Reverse(l , 0 , k - 1)
    Reverse(l , k, len(l) - 1)
    return l
```

	Test	Expected	Got	
✔	print(rotatelist([1,2,3,4,5],1))	[5, 1, 2, 3, 4]	[5, 1, 2, 3, 4]	✔
✔	print(rotatelist([1,2,3,4,5],3))	[3, 4, 5, 1, 2]	[3, 4, 5, 1, 2]	✔
✔	print(rotatelist([1,2,3,4,5],12))	[4, 5, 1, 2, 3]	[4, 5, 1, 2, 3]	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Write a Python function sumofsquares(m) that takes an integer m returns True if m is a sum of squares and False otherwise. (If m is not positive, your function should return False.)

Here are some examples to show how your function should work.
>>> sumofsquares(41)
True

>>> sumofsquares(30)
False

>>> sumofsquares(17)
True

Answer: (penalty regime: 0 %)

Reset answer

```python
from math import *

def issquare(n):
    k = int(sqrt(n))
    return(k*k == n)

def sumofsquares(m):
    for num in range(m//2):
        res=issquare(num) and issquare(m-num)
        if res==True:
            break
    return res
```

	Test	Expected	Got	
✔	print(sumofsquares(41))	True	True	✔
✔	print(sumofsquares(30))	False	False	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Euclid was a Greek mathematician who lived approximately 2,300 years ago. His
algorithm for computing the greatest common divisor of two positive integers, a and
b, is both efficient and recursive. It is outlined below:

If b is 0 then
    eturn a
Else
    Set c equal to the remainder when a is divided by b
    Return the greatest common divisor of b and c

Write a program that implements Euclid's algorithm and uses it to determine the
greatest common divisor of two integers entered by the user. Test your program with some very large integers. The result will be
computed quickly, even for huge numbers consisting of hundreds of digits, because Euclid's algorithm is extremely efficient.

Answer: (penalty regime: 0 %)

```
 1  def euclidean_algorithm(a, b):
 2      while b != 0:
 3          a, b = b, a % b
 4      return a
 5
 6  # Example usage:
 7  a = int(input())
 8  b = int(input())
 9  gcd = euclidean_algorithm(a, b)
10  print(gcd)
11
```

	Input	Expected	Got	
✔	8 12	4	4	✔
✔	720 1000	40	40	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Write a function that takes three numbers as parameters, and returns the median value of those parameters as its result.

Answer: (penalty regime: 0 %)

[Reset answer]

```
1  def median(a, b, c):
2      l = []
3      l.append(a)
4      l.append(b)
5      l.append(c)
6      l.sort()
7      return l[1]
```

	Test	Expected	Got	
✔	print(median(10, 20, 30))	20	20	✔
✔	print(median(60, 50, 40))	50	50	✔
✔	print(median(70, 90, 80))	80	80	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

A string with parentheses is well bracketed if all parentheses are matched: every opening bracket has a matching closing bracket and vice versa.

Write a Python function wellbracketed(s) that takes a string s containing parentheses and returns True if s is well bracketed and False otherwise.

Hint: Keep track of the nesting depth of brackets. Initially the depth is 0. The depth increases with each opening bracket and decreases with each closing bracket. What are the constraints on the value of the nesting depth for the string to be wellbracketed?

Here are some examples to show how your function should work.

>>> wellbracketed("22)")
False

>>> wellbracketed("(a+b)(a-b)")
True

>>> wellbracketed("(a(b+c)-d)((e+f)")
False

Answer: (penalty regime: 0 %)

[Reset answer]

```python
1  def wellbracketed(s):
2      depth = 0
3      for char in s:
4          if char == '(':
5              depth += 1
6          elif char == ')':
7              depth -= 1
8          if depth < 0:
9              return False
10     return depth == 0
11
12
13
```

	Test	Expected	Got	
✔	print(wellbracketed("22)"))	False	False	✔
✔	print(wellbracketed("(a+b)(a-b)"))	True	True	✔
✔	print(wellbracketed("(a(b+c)-d)((e+f)"))	False	False	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Week-07_MCQ

Jump to...

WEEK-07-Extra ►