Dashboard / My courses / CD19411-PPD-2022 / WEEK_07-Functions / WEEK-07_CODING

Started on	Friday, 26 April 2024, 1:52 PM
State	Finished
Completed on	Friday, 26 April 2024, 9:13 PM
Time taken	7 hours 21 mins
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	ADHITHYA PG 2022-CSD-A

Question **1**

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

```
from math import *
def issquare(n):
    k = int(sqrt(n))
    print(k)
    return(k*k == n)


def sumofsquares(m):
    if m<0:
        return False
    for i in range(1,m):
        for j in range(1,m):
            n = i**2 + j**2
            if m == n:
                return True
            elif n > m:
                break
    return False
```

	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 **2**

Correct

Mark 1.00 out of 1.00

Write a program that reads values from the user until a blank line is entered. Display the total of all of the values entered by the user (or 0 if the first value entered is a blank line). Complete this task using recursion. Your program may not use any loops.
Hint: The body of your recursive function will need to read one value from the
user, and then determine whether or not to make a recursive call. Your function
does not need to take any arguments, but it will need to return a numeric result.

Sample Input

5
10
15
20
25

Sample Output

75

Answer: (penalty regime: 0 %)

Reset answer

```python
def readAndTotal():
  value = input("")
  if not value:
    return 0
  else:
    try:
      # Convert input to float for handling decimals
      num = float(value)
      return num + readAndTotal()  # Recursive call with sum
    except ValueError:
      print("Invalid input. Please enter a number.")
      return get_total()  # Retry on invalid input

# Get the total from the user
total = readAndTotal()
print("%.0f"%total)



```

	Input	Expected	Got	
✓	5 10 15 20 25	75	75	✓

Passed all tests! ✓

Correct
Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

A prime number is an integer greater than one that is only divisible by one and itself. Write a function that determines whether or not its parameter is prime, returning True if it is, and False otherwise.

Answer: (penalty regime: 0 %)

Reset answer

```
def isPrime(n):
  if n <= 1:
    return False
  for i in range(2, int(n**0.5) + 1):
    if n % i == 0:
      return False
  return True
```

	Test	Expected	Got	
✓	`print(isPrime(1))`	False	False	✓
✓	`print(isPrime(2))`	True	True	✓
✓	`print(isPrime(3))`	True	True	✓

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

```
def median(a, b, c):
    return (a+b+c)//3
```

	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

```
def wellbracketed(s):
    count = 0
    open_string = "("
    close_string = ")"

    for char in s:
        if(char in open_string):
            count += 1
        elif(char in close_string):
            count -= 1
        else:
            continue
    if count == 0:
        return True
    else:
        return False
```

	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 ►