Dashboard / My courses / PSPP/PUP / Functions: Built-in functions, User-defined functions, Recursive functions / Week9_Coding

Started on	Wednesday, 19 June 2024, 9:26 PM
State	Finished
Completed on	Wednesday, 19 June 2024, 9:49 PM
Time taken	23 mins 20 secs
Marks	3.00/5.00
Grade	**60.00** out of 100.00

Question **1**

Incorrect

Mark 0.00 out of 1.00

complete function to implement coin change making problem i.e. finding the minimum

number of coins of certain denominations that add up to given amount of money.

The only available coins are of values 1, 2, 3, 4

Input Format:

Integer input from stdin.

Output Format:

return the minimum number of coins required to meet the given target.

Example Input:

16

Output:

4

Explanation:

We need only 4 coins of value 4 each

Example Input:

25

Output:

7

Explanation:

We need 6 coins of 4 value, and 1 coin of 1 value

Answer: (penalty regime: 0 %)

Reset answer

```python
def min_coins_needed(target):
    # Define coin denominations
    coins = [1, 2, 3, 4]

    # Initialize dp array with a large number (infinity)
    dp = [float('inf')] * (target + 1)
    dp[0] = 0  # 0 coins needed to make amount 0

    # Iterate through each coin denomination
    for coin in coins:
        for j in range(coin, target + 1):
            dp[j] = min(dp[j], dp[j - coin] + 1)

    # The result is the minimum coins needed to make the target amount
    return dp[target]

if __name__ == "__main__":
    target = 25
    result = min_coins_needed(target)
    print(result)


```

	Test	Expected	Got	

✗	`print(coinChange(16))`	4	7
			
			`***Run error***`
			`Traceback (most recent call last):`
			`  File "__tester__.python3", line 49, in <module>`
			`    print(coinChange(16))`
			`          ^^^^^^^^^^`
			`NameError: name 'coinChange' is not defined`
				✗

Testing was aborted due to error.

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

A number is considered to be ugly if its only prime factors are 2, 3 or 5.

[1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...] is the sequence of ugly numbers.

Task:

complete the function which takes a number n as input and checks if it's an ugly number.

return ugly if it is ugly, else return not ugly

Hint:

An ugly number U can be expressed as: U = 2^a * 3^b * 5^c, where a, b and c are nonnegative integers.

For example:

Test	**Result**
`print(checkUgly(6))`	`ugly`
`print(checkUgly(21))`	`not ugly`

Answer: (penalty regime: 0 %)

Reset answer

```python
def checkUgly(n):
    if n <= 0:
        return "not ugly"

    # Divide by 2 as long as possible
    while n % 2 == 0:
        n //= 2

    # Divide by 3 as long as possible
    while n % 3 == 0:
        n //= 3

    # Divide by 5 as long as possible
    while n % 5 == 0:
        n //= 5

    # After dividing by 2, 3, and 5, if n becomes 1, it's an ugly number
    if n == 1:
        return "ugly"
    else:
        return "not ugly"



```

	Test	**Expected**	**Got**	
✓	`print(checkUgly(6))`	`ugly`	`ugly`	✓
✓	`print(checkUgly(21))`	`not ugly`	`not ugly`	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Incorrect

Mark 0.00 out of 1.00

Write a code to check whether product of digits at even places is divisible by sum of digits

at odd place of a positive integer.

Input Format:

Take an input integer from stdin.

Output Format:

Print TRUE or FALSE.

Example Input:

1256

Output:

TRUE

Example Input:

1595

Output:

FALSE

For example:

Test	Result
`print(productDigits(1256))`	`True`
`print(productDigits(1595))`	`False`

Answer: (penalty regime: 0 %)

Reset answer

```
def productDigits(n):
    # Convert number to string to iterate over digits
    num_str = str(n)

    # Initialize variables
    product_even = 1
    sum_odd = 0

    # Iterate over each digit
    for i in range(len(num_str)):
        digit = int(num_str[i])

        # Check if index is even or odd
        if i % 2 == 0:  # Even index (0-indexed)
            product_even *= digit
        else:  # Odd index (0-indexed)
            sum_odd += digit

    # Check the divisibility property
    if sum_odd != 0 and product_even % sum_odd == 0:
        return "True"
    else:
        return "False"
```

	Test	Expected	Got	
✗	`print(productDigits(1256))`	`True`	`False`	✗
✓	`print(productDigits(1595))`	`False`	`False`	✓

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

An e-commerce company plans to give their customers a special discount for Christmas.

They are planning to offer a flat discount. The discount value is calculated as the sum of all

the prime digits in the total bill amount.

Write an algorithm to find the discount value for the given total bill amount.

Constraints

1 <= orderValue< 10e100000

Input

The input consists of an integer orderValue, representing the total bill amount.

Output

Print an integer representing the discount value for the given total bill amount.

Example Input

578

Output

12

For example:

Test	Result
`print(christmasDiscount(578))`	12

Answer: (penalty regime: 0 %)

Reset answer

```python
def christmasDiscount(orderValue):
    # Prime digits considered for discount
    prime_digits = {'2', '3', '5', '7'}

    # Convert orderValue to string to iterate over digits
    orderValue_str = str(orderValue)

    # Initialize discount value
    discount_value = 0

    # Iterate over each digit in orderValue_str
    for digit_char in orderValue_str:
        if digit_char in prime_digits:
            discount_value += int(digit_char)

    return discount_value


```

	Test	Expected	Got	
✓	`print(christmasDiscount(578))`	12	12	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

An abundant number is a number for which the sum of its proper divisors is greater than

the number itself. Proper divisors of the number are those that are strictly lesser than the number.

Input Format:

Take input an integer from stdin

Output Format:

Return Yes if given number is Abundant. Otherwise, print No

Example input:

12

Output:

Yes

Explanation

The proper divisors of 12 are: 1, 2, 3, 4, 6, whose sum is 1 + 2 + 3 + 4 + 6 = 16. Since sum of

proper divisors is greater than the given number, 12 is an abundant number.

Example input:

13

Output:

No

Explanation

The proper divisors of 13 is: 1, whose sum is 1. Since sum of proper divisors is not greater

than the given number, 13 is not an abundant number.

For example:

Test	Result
`print(abundant(12))`	`Yes`
`print(abundant(13))`	`No`

Answer: (penalty regime: 0 %)

Reset answer

```
def abundant(n):
    if n <= 0:
        return "No"  # We consider only positive integers

    sum_divisors = 0
    # Find all divisors of n (excluding n itself)
    for i in range(1, n):
        if n % i == 0:
            sum_divisors += i

    # Compare sum of divisors with n
    if sum_divisors > n:
        return "Yes"
    else:
        return "No"



```

	Test	Expected	Got	
✓	`print(abundant(12))`	`Yes`	`Yes`	✓
✓	`print(abundant(13))`	`No`	`No`	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week9_MCQ

Jump to...

Searching ►