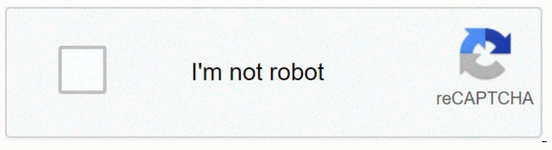


Continue

**Codehs fibonacci answer**

Covid-19 is increasingly threatening the lives of all mankind. Below is a manual of medicines and first aid supplies that you should take with you during an epidemic. Read More During the Covid-19 pandemic, the first and most important thing you need to keep in mind is Don'#39;t: Accumulation and mass purchases of toilet paper and medical masks. Do: Be sure to provide enough Read More The quality of the floor product is &#39 not depend only on the fabric; however, the floor seller is important at the same time that he chooses and buys the floor. The carpet gallery is legendary and reliab Read More People are talking about Artificial Intelligence and Cryptocureence that greatly affect the online activities of Internet users, such as online shopping these days. Both aspects are being narrated as Read More Crypto-jacking, DDoS attack, ransomware attacks are all some of the most popular cyber attacks. That's why everyone focuses on these attacks. Yes, it may seem like a sensible decision to protect ag Read More Page 2 Covid-19 is increasingly threatening the lives of all mankind. Below is a manual of medicines and first aid supplies that you should take with you during an epidemic. Read More During the Covid-19 pandemic, the first and most important thing you need to keep in mind is Don'#39;t: Accumulation and mass purchases of toilet paper and medical masks. Do: Be sure to provide enough Read More The quality of the floor product is &#39 not depend only on the fabric; however, the floor seller is important at the same time that he chooses and buys the floor. The carpet gallery is legendary and reliab Read More People are talking about Artificial Intelligence and Cryptocureence that greatly affect the online activities of Internet users, such as online shopping these days. Both aspects are being narrated as Read More Crypto-jacking, DDoS attack, ransomware attacks are all some of the most popular cyber attacks. That's why everyone focuses on these attacks. Yes, it may seem like a sensible decision to protect ag Read More Page 3 The Covid-19 is increasingly threatening the lives of all mankind. Below is a manual of medicines and first aid supplies that you should take with you during an epidemic. Read More During the Covid-19 pandemic, the first and most important thing you need to keep in mind is Don'#39;t: Accumulation and mass purchases of toilet paper and medical masks. Do: Be sure to provide enough Read More The quality of the floor product is &#39 not depend only on the fabric; however, the floor seller is important at the same time that he chooses and buys the floor. The carpet gallery is legendary and reliab Read More People are talking about Artificial Intelligence and Cryptohealing that greatly affect the online activities of the internet, such as online shopping these days. Both aspects are being narrated as Read More Crypto-jacking, DDoS attack, ransomware attacks are all some of the most popular popular That's why everyone focuses on these attacks. Yes, it may seem like a sensible decision to protect ag Read More Page 4 Covid-19 is increasingly threatening the lives of all mankind. Below is a manual of medicines and first aid supplies that you should take with you during an epidemic. Read More During the Covid-19 pandemic, the first and most important thing you need to keep in mind is Don'#39;t: Accumulation and mass purchases of toilet paper and medical masks. Do: Be sure to provide enough Read More The quality of the floor product is &#39 not depend only on the fabric; however, the floor seller is important at the same time that he chooses and buys the floor. The carpet gallery is legendary and reliab Read More People are talking about Artificial Intelligence and Cryptocureence that greatly affect the online activities of Internet users, such as online shopping these days. Both aspects are being narrated as Read More Crypto-jacking, DDoS attack, ransomware attacks are all some of the most popular cyber attacks. That's why everyone focuses on these attacks. Yes, it may seem like a sensible decision to protect the print ag Read More(Hello world) (2 + 2) print(10) (My name is %s and I am %d years old! % ('Zara', 21)) # Make a variable to store the text name = Zach # Create variables that are numbers num_one = 3 num_two = 4 sum = num_one + num_two # We can also assign multiple variables at the same time num_one, num_two = 3, 4 # The value of a variable can be changed after created num_one = num_one + 1 # The output will be str, or string print (type(name)) # The output will be int, or integer printing (type(sum)) # The output will be bool, or boolean printing (type(3==3)) # We can change the type of element using the shortened name of type # We do this by concatenating strings , such as: age = 16 printing (My age is + str(age)) # Ask the user for input and save it for a variable to be used in code # This will only work if the entry is being used as a string name = input (What is your name?) If the entry needs to be used as a number, include the term 'int' num_one = int(Type a number: )) num_two = int(entry(Type a second number:)) # This entry can then be used to control different parts of the code print (hello, + name) sum = num_one + num_two A multiple-line comment describes your code to someone who is reading. Example: This program will ask the user for two numbers. It will then add the numbers and print the final value. number_one = int(entry(enter a number: )) number_two = int(int(Type a second number: )) print (Sum: + str(number_one + number_two)) # Use single-line comments to clarify parts of the code. Example: # This program adds 1 and 2 added = 1 + 2 printing (added) # This code will end when use comes into a negative = int(input(Enter a number: )) if the number < 0: break something else: print(str(number)) # This code will only print the numbers 0 to 3 and 6 for i in the range(5): if I < < print(i) another: continue printing (6) # Try/Except with input attempt: my_number = int(Type an integer: )) print(Your number: + str(my_number)) except ValueError: print(This was not an integer!) # Try/Except for Type Errors try: my_number = '2' + 2 except TypeError: print(A type error occurred!) # Try/Except for Key dictionary = {'1':'K', '3':'A', '4':'R', '5':'E', '6':'L'} try: dictionary['2'] except KeyError: print(Key error) # Try/Except for Attribute Errors try: dictionary.no_method() except AttributeError: print( Attribute error!) # You can also try: my_number = int(input((input). Enter an integer: )) print (Its number: + str(my_number)) except: print (There was an error.) # Random integer between (and including) random low and high import random_num = random.randint (low, high) random_element = random.choice(string) # Example: # Returns random number in and including 0 and 10. random_num = random.randint(0,10) # Random element in a string random_element = random.choice ('abcdefghij') if BOOLEAN_EXPRESSION: print(This performs if BOOLEAN_EXPRESSION evaluates to True) # Example: # Text will only be printed if the user types a negative number = int(Type a number:)) if the number < 0: print(str(number) + is negative!) if condition_1: print(This runs if condition_1 evaluates for True) elif condition_2: print(This runs if condition_2 evaluates for True) other: print(This performs if no preconditions evaluate the Truth) # Example: # This program will print that the color is secondary == purple if color == red or color == blue or color == yellow: print(Primary color.) elif color == green or color == orange or color == purple: print(Secondary color.) then : print(Not a primary or secondary color.) + Addition - Subtraction * Multiplication / Division % Module (Remainder) () Parentheses (For order of operations) # Examples z = x + y w = x * y # Division a = 5.0 / 2 # Returns 2.5 b = 5.0 // 2 # Returns 2.0 c = 5/2 # Returns 2 2 # Increment (add one) x += 1 # Decrement (subtract one) x -= 1 # Absolute value absolute_value = abs(x) abs_val = abs(-5) # Returns 5 # Square root import math square_root = math.sqrt(x) # Elevate to a power = math.pow(x, y) # Calculates x^y # Rounding rounded_num = round(2,675, 2) # Returns 2.68 x == y # is x equals y x != y # is x # is not equal to y > x gt; y # is x greater than y x >= y # is x greater than or equal to y x < y # is x less than y x 5 : print(x is greater than 5.) # And Operator and_expression = x and y # Or Operator or_expression = x or y # You can combine many Booleans! boolean_expression = x and (y or z) # This for loop will print hello 5 times for i in range (5): print(hello) # This for loop will print even numbers from 1 to 10 to number on the track(2, 11, 2): print (hello) # This code runs on each item in my_list # This loop will print 1, then 5, then 10, then 15 = [1, 5, 10, 15] para item em my_list: my_list: # This program will run as long as the variable 'number' is greater than 0 # Countdown from 10 to 0 number = 10 while the number >= 0: print number (number) -= 1 # You can also use user input to control a loop of time # This code will continue to run while the user answers 'Yes' answer = input (Continue code?:) while the answer == Yes: answer = input (Continue code?: ) def name_of_your_function(): # Code that will run when you make a call to # this function. # Example: # Teach the computer to add two numbers num_one = 1 num_two = 2 def add_numbers(): sum = num_one + num_two # We add a return declaration in order to use the value of the variable sum num_one = 1 num_two = 2 def add_numbers(): sum = num_one + num_two return sum # Call the add_numbers (function once # The computer will return a value of 3 add_numbers() # Call the add_numbers() function 3 times and print the output # The output will be the number 3 printed on 3 separate lines print(add_numbers()) print(add_numbers()) print(add_numbers()) # In this program, the parameters are used to give two numbers def add_numbers(num_one, num_two): sum = num_one + num_two return sum # We call the function with values within the parentheses # This program will print '7' (add_numbers(3) , 4)) # If we have a list with the same number of parameters, we can use the items to assign arguments using an asterisk my_list = [3, 4] print (add_numbers(*my_list)) # Prints a character on a specific index my_string = hello! print (my_string[0]) # prints h print(my_string[5]) # prints ! # Prints all character prints after the specific index my_string = Hello world! print (my_string[1:]) # prints ello world! print (my_string[6:]) # print world! # Prints all characters before the specific index my_string = Hello world! print (my_string[:6]) # prints hello print (my_string[:1]) # prints h # Prints all characters between specific indexes my_string = Hello world! printed (my_string[1:1:1:16]) # prints ello print (my_string[4:7]) # prints w # Itera through each character on string # Will print a letter of string on each line to my_string = Turtle to c in my_string : print(c) # Complete commands if the string is found within the given sequence my_string = hello world! if world in my_string: print (world) # Concatetenage my_string = Tracy the print (my_string My_string + turtle) # prints Tracy, the turtle # Divides the string into a list of letters my_string = Tracy my_list = list(my_string) # my_list = ['T', 'r', 'a', 'c', 'y'] # Using enumerate will print the index number followed by a colon and the word # in that index for each word in the list my_string = Tracy is a turtle for index, word enumerated(my_string.split()): print(str(index) + : + word) # top: To make a whole string uppercase = hello my_string = my_string.upper() # hello returns # bottom: To make a string all lowercase my_string = Hello my_string = my_string.lower() # returns hello # # Returns True if a string is all capital letters and false letters otherwise my_string = HELLO printing (my_string.isupper()) # returns True # islower: Returns True if a string is all lowercase letters and False my_string = Hello print (my_string.islower()) # Returns False # swapcase: Returns a sequence where each letter is the opposite case of the original my_string = PyThOn my_string = my_string.swapcase() # returns pYt #strip: Returns a copy of the string without any white space at the beginning or end my_string = hi there my_string = my_string.strip() # returns hi there # find: Returns the smallest string index where substr resarem -1 if no substring is found my_string = eggplant index = my_string.find (plant) # returns 3 index = my_string.find(Tracy) # returns -1 # split : Divides the string into a list of words in whitespace my_string = Tracy is a turtle my_list = my_string.split() # Returns ['Tracy', 'is', 'a', 'turtle'] # Make a new tuple called my_tuple my_tuple = (1, 2, 3, 4, 5) # Tuple with elements of different types my_tuple = (0, 1, Tracy, (1, 2)) # Tuple with single element my_tuple = (3,) # Tuple of tuples my_tuple((0, 1), (2, 3)) # Get the length of the tuple print (len(my_tuple)) # Access elements within nested tuples print (my_tuple[0][0]) printing (my_tuple [1][0]) # Concateando tuples x = (1, 2) y = (5, 6) my_tuple = x + + (3,) + y # Create an empty list my_list = [] # Create a list with any number of items my_list = [item1 , item2, item3] # Example: number_list = [1, 2, 4] # A list can have any type my_list = [integer, string, boolean] # Example: a_list = [hello, 4, True] # Access an element in a list a_list = [hello, 4, True] first_element = a_list[0] # Returns hello # Set an element in a list a_list = [Hello, 4, True] a_list[0] = 9 # Changes a_list be [9, 4, True] # Looping over a list # Prints each item on a separate line (9, then 4, then True) a_list = [9, 4, True] for item in a_list: print (item) # Length of a list a_list = [9, 4, True] a_list_length = len(a_list) # Returns 3 # Creates a list based on the first operation # This will create a list with numbers 0 to 4 a_list = [x to x in the range(5)] # This will create a list with multiples from 0 to 8 list_of_multiples = [2*x for x in the track(5)] # appendix : Add to a list a_list = [hello, 4, True] a_list.append(Puppy) # Now a_list = [hello, 4, True, Puppy] # pop: Remove and return the last element of the list a_list = [Hello, 4, True] last_item = a_list.pop() # Remove True, now a_list = [hello, 4] # Remove and return an item from a list in the index i a_list = [hello, 4, True] a_list.pop(0) # Remove hello, now a_list = [4, True] # index: Returns the index value of the first item in the list that corresponds to the element # There is an error if there is no such item a_list = [hello, 4, True] a_list.index(4) # Returns 1 because 4 is found in the index[1] a_list.index(hi) # Error because no item hi # : Returns a sorted sorted list = [9, 7, 1, 2, 3] my_list.sort() # Returns [1, 2, 3, 7, 9] # reverse: Returns an inverted list my_list = [1, 2, 3, 4] my_list.reverse() # Returns [4, 3, 2, 1] # count: Returns the number of instances of a specific item that were found my_list = [1, 4, 2, -4, 10, 0, 4, 2, 1, 4] print(my_list.count(4)) # Returns 3 print (my_list.count(123)) # Returns 0 because 123 does not exist in the # extend list: Allows you to add a list to a list my_list = [1, 2, 3] my_list.extend([4, 5, 6]) # Returns [1, 2, 3, 4, 5, 6] # remove: Allows you to remove one specific item from a list # Only removes the first instance of item my_list = [apple, banana, orange, grapefruit] my_list.remove(orange) # Returns [apple , banana, grapefruit] # join: Creates string out of list with specified string placed between each item my_list = [Tracy, is, a, turtle] ( .join(my_list) # Returns the list as a string with spaces between words # Create an empty list my_list = [] # Add to the list my_list.append([1, 2]) my_list.append([4, 5, 6]) # Access elements within the nested print lists(my_list[0]) Returns # [1, 2, 3] print(my_list[0][1]) # Take a slice of my_list the inner list print(my_list[0][0:2]) # Returns [1, 2, 3], [4, 5, 6]] # Take a slice of the inner list print(my_list[0][0:2]) # Returns [1, 2] a_dictionary = {key 1:value1, key2 :value2} # Example: my_farm = {pigs:2, cows:4} # This dictionary keeps the animal count of a farm # Creates an empty dictionary a_dictionary = {} # Inserts a key value pair a_dictionary[key] = value my_farm[horses] = 1 # The farm now has a horse # Gets a value to a key my_dict[key] # Will return the key my_farm[pigs] # Will come back 2, the value of pigs # Using the keyword 'in' my_dict = {aa: 1, b: printing 2} (a in my_dict) # Returns True print (z in my_dict) # Returns false print(2 in my_dict) # False Returns, because 2 is not a key # Iterating through a dictionary for key in my_dict: print(key: + str(key)) print (value: + str(my_dict[key])) # Declare a MyClass class : # The __init__ method is called whenever we instantiate our class def __init__(self): print (Started class) self.my_num = 0 # Instantiate its class my_class = MyClass() # Access instance variables in its class impression (my_class.my_num) my_class.my_num = 10 # argument Adding to your point class : def __init__(self, x 0, y = 0): self.x = x self.y = y # # Instantiate the class p = Point(3, 4) # Make a new set called new_set new_set = set([[]]) girl_scout_badges = set([]) # Add to a set new_set New_set.add(item) girl_scout_badges.add (Squirrel Whisperer) # A set contains a value item in my_set # Returns a Charming from Squirrels in girl_scout_badges # Returns True # Number of elements in the set len (my_set) len (my_set) len (girl_scout_badges my_set <1> <4>) # Returns 1 since there is only one item in the set # Extracting Data from a : # Sample file: # test.txt # ---------- # Hello World # This is File Input # ---------- # Opening file, file, a file object and store it in a variable: file = open('test.txt') # Getting all the text: file.read() # Returns: # Hello World # Returns: # Hello World! That's file entry! # Note '', meaning the end of a line of text to line in file: print (line + '!') # Returns: # Hello World # ! This is the entry of files ! # To remove this new extra line, we can use: for line in file: print(line.strip() + '!') # Returns: # Hello World! # This is the file entry! # Closing a file.close() file.close()