COMPUTATIONAL THINKING EXERCISE:

FUNCTIONS REFRESHER

Notes: A function is defined with the word followed by the name of the function (which describes what the function does), brackets (with the things that the function will use inside the brackets called **parameters**) and a semi colon.

```
1.  def name_of_function(parameters):
2.    #your code hear
3.    return(result)
```

The code/instructions that belong to a function must be indented. And if you want your function to return a result so that you can do something else with the result, then you can return that result like in line 3 above. We "call" a function (make a function do the code inside it) by writing the name of the function, and giving it what it needs in the brackets.

```
1.  name_of_function(parameters)
```

Generally you put all your functions at the top of the page. And the rest of your code at the bottom of your page. Your computer reads from the top, downwards. So if you call your function (tell the function to do what it does) before you have defined what your function does, then the computer will not know what you are talking about.

```
1.  def function1(parameters):
2.    #your code hear
3.    return(answer)
4.
5.
6.  def function2(parameters):
7.    #your code hear
8.    return(answer)
9.
10.
11. a = function1(parameter)
12. b = function2(parameter)
```

What ever is indented (belongs to the function) the rest of your program will not know about. If you run the code below, you will see that the computer does not know what x is because it belongs to the function. It says "x is not defined". So we have to return it in order to use it.

```
1.  def function1():
2.    x = 3
3.    return x
4.
5.  newx = function1()
6.  print(newx)
7.  print(x)
```

So we need to return the value inside the function so that the rest of the program can access it.
Step 1:

Open up a console from here https://trinket.io/python3

Copy paste this code into trinket and write your algorithm in the add_numbers_my_algorithm() function (remember # means that it is a comment and the computer ignores it). Your function should be able to take a particular number parameter (pn) and add up all the numbers from 1 up until and including that particular number (using your newly learned computational thinking skills). Assign the result to the answer variable (instead of it being None) so that the answer can be returned when you call the function.

```
1.  def add_numbers_my_algorithm(pn):
2.    #your algorithm in code here
3.    # e.g step1= pn+50
4.    #     step2= pn*50
5.    #     step3= pn/5
6.    #     step4= step1 + step2 + step3
7.    #     answer = step4
8.    answer = None
9.    return(answer)
10.
11.
12. #call your function
13. my_algorithm_answer = add_numbers_my_algorithm(3)
14. print(my_algorithm_answer)
```

When we return the result of the function, we give the result the name my_algorithm_answer so that we can print it out. Test you function by calling it and passing it a number parameter. Like in line 13 above. Then click the run button to print the answer/result.

Step 2:
Once your function is able to return and print an answer, it easy to manually check it for small numbers that you can quickly do yourself (or 200). However, it is really difficult to make sure it does it for very big numbers. Let's add the numbers in a different way so that we can check our algorithm. Don't delete your code!

We will do this by adding a add_numbers_computer_algorithm() function at the top of our page. Computers have their own ways of adding many numbers. This function will also take the parameter pn and return an answer. This problem we will break down into 2 sub-steps.

1. Using google, search the internet on how to create an array of numbers in the range between 1 and pn. Assign the array to the answer vaiable(make answer equal the array) so it can be printed out.

```
def add_numbers_computer_algorithm(pn):
  #create array of numbers in range between 1 and pn
  answer = None
  return(answer)


computer_algorithm_answer = add_numbers_computer_algorithm(3)
print(computer_algorithm_answer)
```

2. Using google, search for how to add up all the numbers in a python array. Assign this result to answer so that we can print it out. Your function should now look like this (with your code where the # comments are)

```
def add_numbers_computer_algorithm(pn):
  #create array of numbers in range between 1 and pn
  #sum all values in array
  answer = None
  return(answer)


computer_algorithm_answer = add_numbers_computer_algorithm(3)
print(computer_algorithm_answer)
```

Step 3

Now we are ready to test our function. The below function will help you do this. Paste this function at the top of your page. At the bottom of your page (after all your code) you can call this function by writing its name and passing it a particular number. See if you can read the lines of code and understand what it is doing. Don't forget to call it to make it run!

```
def testing_function(pn):
  my_algorithm = add_numbers_my_algorithm(pn)
  computer = add_numbers_computer_algorithm
  if my_algorithm == computer:
    print("Well done, looks like your algorithm works")
  else:
    print("Sorry, it doesnt look like your algorithm is working")
```

Note: This function does not return anything, functions don't always have to return stuff

CHALLENGE!
Does are program work for odd numbers? Can you make it work for odd numbers too?