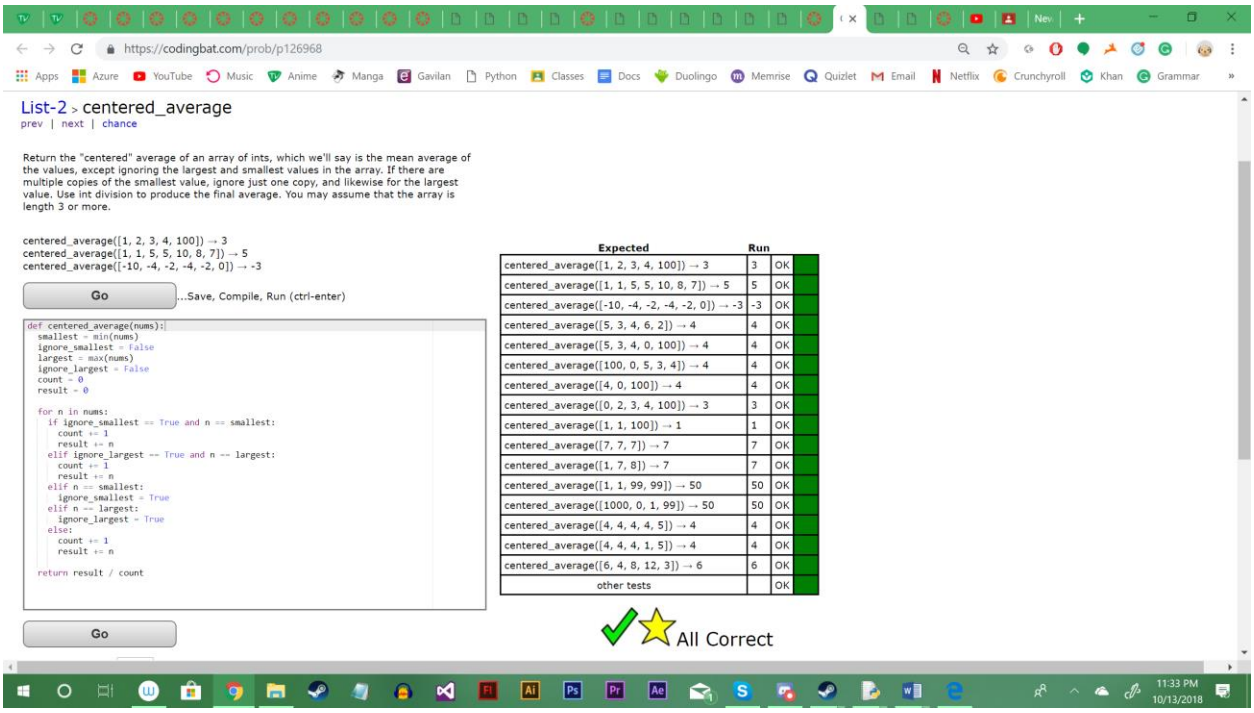
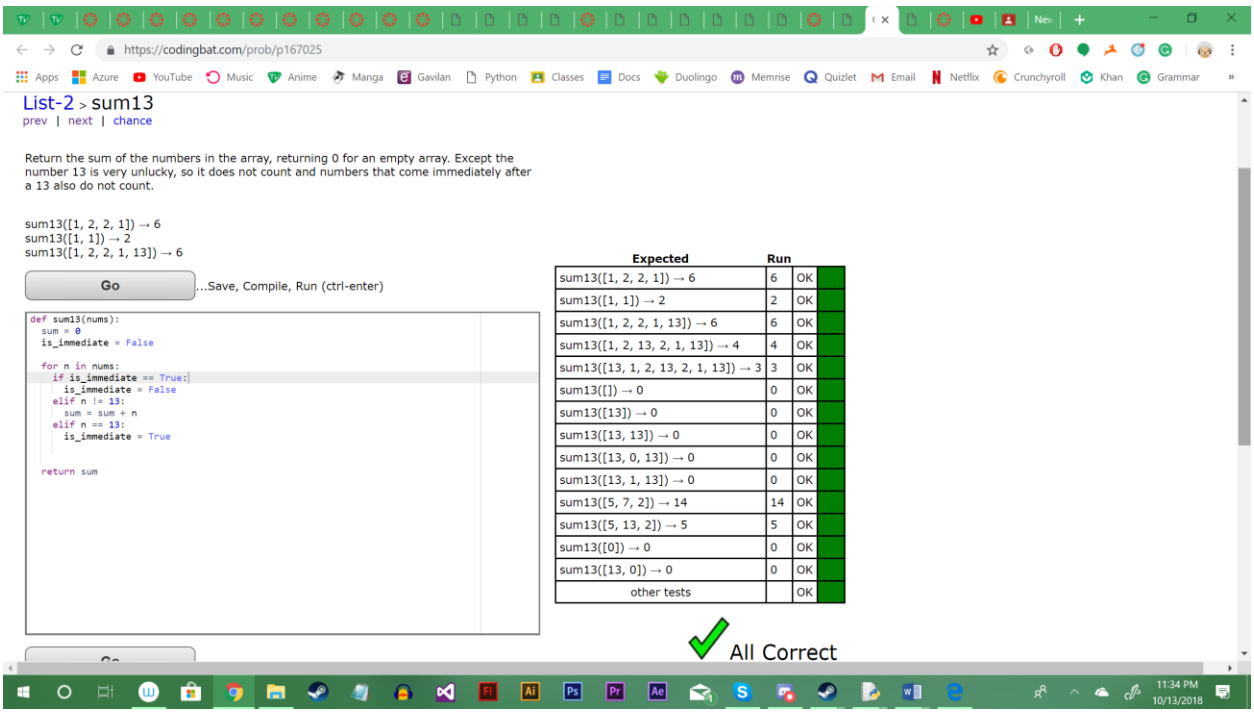
P37

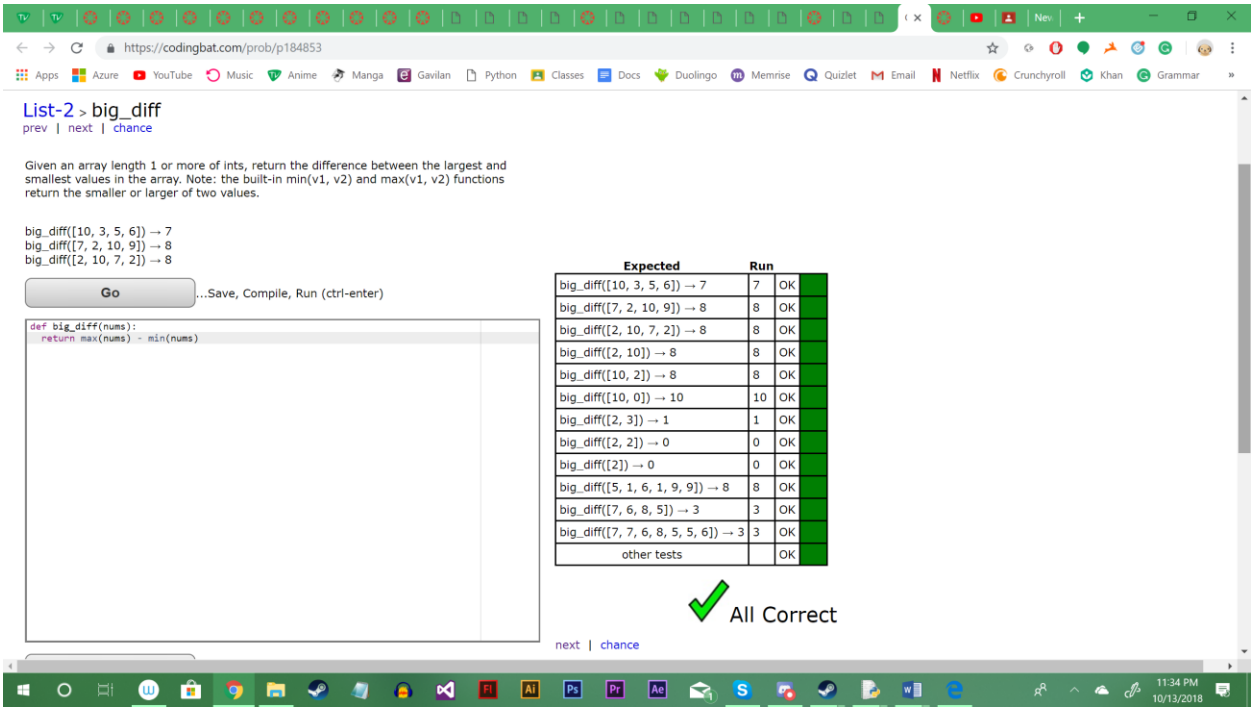
1.


https://codingbat.com/prob/p126968
List-2 > centered_average
prev | next | chance
Return the "centered" average of an array of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use int division to produce the final average. You may assume that the array is length 3 or more.
centered_average([1, 2, 3, 4, 100]) → 3
centered_average([1, 1, 5, 5, 10, 8, 7]) → 5
centered_average([-10, -4, -2, -4, -2, 0]) → -3
Go
...Save, Compile, Run (ctrl-enter)
def centered_average(nums):
smallest = min(nums)
ignore_smallest = False
largest = max(nums)
ignore_largest = False
count = 0
result = 0
for n in nums:
if ignore_smallest == True and n == smallest:
count += 1
result += n
elif ignore_largest == True and n == largest:
count += 1
result += n
elif n == smallest:
ignore_smallest = True
elif n == largest:
ignore_largest = True
else:
count += 1
result += n
return result / count
Go
Expected
Run
centered_average([1, 2, 3, 4, 100]) → 3 3 OK
centered_average([1, 1, 5, 5, 10, 8, 7]) → 5 5 OK
centered_average([-10, -4, -2, -4, -2, 0]) → -3 -3 OK
centered_average([5, 3, 4, 6, 2]) → 4 4 OK
centered_average([5, 3, 4, 0, 100]) → 4 4 OK
centered_average([100, 0, 5, 3, 4]) → 4 4 OK
centered_average([4, 0, 100]) → 4 4 OK
centered_average([0, 2, 3, 4, 100]) → 3 3 OK
centered_average([1, 1, 100]) → 1 1 OK
centered_average([7, 7, 7]) → 7 7 OK
centered_average([1, 7, 8]) → 7 7 OK
centered_average([1, 1, 99, 99]) → 50 50 OK
centered_average([1000, 0, 1, 99]) → 50 50 OK
centered_average([4, 4, 4, 4, 5]) → 4 4 OK
centered_average([4, 4, 4, 1, 5]) → 4 4 OK
centered_average([6, 4, 8, 12, 3]) → 6 6 OK
other tests OK
All Correct

2.


https://codingbat.com/prob/p167025
List-2 > sum13
prev | next | chance
Return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and numbers that come immediately after a 13 also do not count.
sum13([1, 2, 2, 1]) → 6
sum13([1, 1]) → 2
sum13([1, 2, 2, 1, 13]) → 6
Go
...Save, Compile, Run (ctrl-enter)
def sum13(nums):
sum = 0
is_immediate = False
for n in nums:
if is_immediate == True:
is_immediate = False
elif n != 13:
sum = sum + n
elif n == 13:
is_immediate = True
return sum
Expected
Run
sum13([1, 2, 2, 1]) → 6 6 OK
sum13([1, 1]) → 2 2 OK
sum13([1, 2, 2, 1, 13]) → 6 6 OK
sum13([1, 2, 13, 2, 1, 13]) → 4 4 OK
sum13([13, 1, 2, 13, 2, 1, 13]) → 3 3 OK
sum13([]) → 0 0 OK
sum13([13]) → 0 0 OK
sum13([13, 13]) → 0 0 OK
sum13([13, 0, 13]) → 0 0 OK
sum13([13, 1, 13]) → 0 0 OK
sum13([5, 7, 2]) → 14 14 OK
sum13([5, 13, 2]) → 5 5 OK
sum13([0]) → 0 0 OK
sum13([13, 0]) → 0 0 OK
other tests OK
All Correct

3.


https://codingbat.com/prob/p184853
List-2 > big_diff
prev | next | chance
Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array. Note: the built-in min(v1, v2) and max(v1, v2) functions return the smaller or larger of two values.
big_diff([10, 3, 5, 6]) → 7
big_diff([7, 2, 10, 9]) → 8
big_diff([2, 10, 7, 2]) → 8
Go
...Save, Compile, Run (ctrl-enter)
def big_diff(nums):
return max(nums) - min(nums)
Expected
Run
big_diff([10, 3, 5, 6]) → 7 | 7 | OK
big_diff([7, 2, 10, 9]) → 8 | 8 | OK
big_diff([2, 10, 7, 2]) → 8 | 8 | OK
big_diff([2, 10]) → 8 | 8 | OK
big_diff([10, 2]) → 8 | 8 | OK
big_diff([10, 0]) → 10 | 10 | OK
big_diff([2, 3]) → 1 | 1 | OK
big_diff([2, 2]) → 0 | 0 | OK
big_diff([2]) → 0 | 0 | OK
big_diff([5, 1, 6, 1, 9, 9]) → 8 | 8 | OK
big_diff([7, 6, 8, 5]) → 3 | 3 | OK
big_diff([7, 7, 6, 8, 5, 5, 6]) → 3 | 3 | OK
other tests | | OK
All Correct
next | chance