Codingbat Python Questions and Answers Section 2

If you have any questions, *please email me* via samet2@gmail.com

Have fun with Python!
Samet Atdag

This document contains 27 questions in these sections:

Warmup-2

1. **string_times**

Given a string and a non-negative int n, return a larger string that is n copies of the original string.

string_times('Hi', 2) → 'HiHi'
string_times('Hi', 3) → 'HiHiHi'
string_times('Hi', 1) → 'Hi'

My solution:

```
def string_times(str, n):
 result = ''
 i=0
 while i<n:
   result += str
   i += 1
 return result
```

2. **front_times**

Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;

front_times('Chocolate', 2) → 'ChoCho'
front_times('Chocolate', 3) → 'ChoChoCho'
front_times('Abc', 3) → 'AbcAbcAbc'

My solution:

```
def front_times(str, n):
 if len(str) < 3:
   front = str
 else:
   front = str[:3]
 result = ''
 for i in range(n):
   result += front
 return result
```

3. **string_bits**

Given a string, return a new string made of every other char starting with the first, so "Hello" yields "Hlo".

string_bits('Hello') → 'Hlo'
string_bits('Hi') → 'H'
string_bits('Heeololeo') → 'Hello'

My solution:

```
def string_bits(str):
 result = ""
 for i in range(0, len(str)):
   if i % 2 == 0:
     result = result + str[i]
 return result
```

4. **string_splosion**

Given a non-empty string like "Code" return a string like "CCoCodCode".

string_splosion('Code') → 'CCoCodCode'
string_splosion('abc') → 'aababc'
string_splosion('ab') → 'aab'

My solution:

```
def string_splosion(str):
 result = ''
 for i in range(len(str)+1):
   result += str[:i]
 return result
```

5. **last2**

Given a string, return the count of the number of times that a substring length 2 appears in the string and also as the last 2 chars of the string, so "hixxxhi" yields 1 (we won't count the end substring).

last2('hixxhi') → 1
last2('xaxxaxaxx') → 1
last2('axxxaaxx') → 2

My solution:

```
def last2(str):
 if len(str) < 2:
   return 0
 last2 = str[len(str)-2:]
 count = 0
 for i in range(len(str)-2):
   sub = str[i:i+2]
   if sub == last2:
     count = count + 1
 return count
```

6. **array_count9**

Given an array of ints, return the number of 9's in the array.

array_count9([1, 2, 9]) → 1
array_count9([1, 9, 9]) → 2
array_count9([1, 9, 9, 3, 9]) → 3

My solution:

```
def array_count9(nums):
 result = 0
 for i in nums:
   if i == 9:
     result += 1
 return result
```

7. **array_front9**

Given an array of ints, return True if one of the first 4 elements in the array is a 9. The array length may be less than 4.

array_front9([1, 2, 9, 3, 4]) → True
array_front9([1, 2, 3, 4, 9]) → False
array_front9([1, 2, 3, 4, 5]) → False

My solution:

```
def array_front9(nums):
 result = False
 for i in range(4):
   if len(nums) > i:
     if nums[i] == 9:
       result = True
 return result
```

8. **array123**

Given an array of ints, return True if .. 1, 2, 3, .. appears in the array somewhere.

array123([1, 1, 2, 3, 1]) → True
array123([1, 1, 2, 4, 1]) → False
array123([1, 1, 2, 1, 2, 3]) → True

My solution:

```
def array123(nums):
 if len(nums)<3:
   return False
 for i in range(len(nums)-2):
   if nums[i] == 1 and nums[i+1] == 2 and nums[i+2] == 3:
     result = True
     break
   else:
     result = False
 return result
```

9. **string_match**

Given 2 strings, a and b, return the number of the positions where they contain the same length 2 substring. So "xxcaazz" and "xxbaaz" yields 3, since the "xx", "aa", and "az" substrings appear in the same place in both strings.

string_match('xxcaazz', 'xxbaaz') → 3
string_match('abc', 'abc') → 2
string_match('abc', 'axc') → 0

My solution:

```
def string_match(a, b):
  r = len(a) if len(a)<len(b) else len(b)
  result = 0
  for i in range(r):
    if a[i:i+2] == b[i:i+2] and len(a[i:i+2]) == 2 and len(b[i:i+2]) == 2:
      result += 1
  return result
```

Logic-2

10. **make_bricks**

We want to make a row of bricks that is **goal** inches long. We have a number of small bricks (1 inch each) and big bricks (5 inches each). Return True if it is possible to make the goal by choosing from the given bricks. This is a little harder than it looks and can be done without any loops.

make_bricks(3, 1, 8) → True
make_bricks(3, 1, 9) → False
make_bricks(3, 2, 10) → True

My solution:

```
def make_bricks(small, big, goal):
  number_of_fives = goal/5
  number_of_ones = goal - (5*number_of_fives)
  if number_of_fives <= big and number_of_ones <= small:
    return True
  elif (number_of_ones+5<=small) and (big*5 + number_of_ones + 5 == goal):
    return True
  else:
    return False
```

11. **lone_sum**

Given 3 int values, a b c, return their sum. However, if one of the values is the same as another of the values, it does not count towards the sum.

lone_sum(1, 2, 3) → 6
lone_sum(3, 2, 3) → 2
lone_sum(3, 3, 3) → 0

My solution:

```
def lone_sum(a, b, c):
  if a == b == c:
    return 0
  elif a == b:
    return c
  elif a == c:
    return b
  elif b == c:
    return a
  else:
    return a + b + c
```

12. **lucky_sum**

Given 3 int values, a b c, return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.

lucky_sum(1, 2, 3) → 6
lucky_sum(1, 2, 13) → 3
lucky_sum(1, 13, 3) → 1

My solution:

```
def lucky_sum(a, b, c):
  if a == 13:
    return 0
  elif b == 13:
    return a
  elif c == 13:
    return a + b
  else:
    return a + b + c
```

13. **no_teen_sum**

Given 3 int values, a b c, return their sum. However, if any of the values is a teen -- in the range 13..19 inclusive -- then that value counts as 0, except 15 and 16 do

not count as a teens. Suggestion: decompose a separate "def fix_teem(n):" below no_teen_sum() that takes in an int value and returns that value fixed for the teen rule. In this way, you avoid repeating the teen code 3 times.

no_teen_sum(1, 2, 3) → 6
no_teen_sum(2, 13, 1) → 3
no_teen_sum(2, 1, 14) → 3

My solution:

```
def no_teen_sum(a, b, c):
  return fix_teem(a) + fix_teem(b) + fix_teem(c)

def fix_teem(n):
  if n>=13 and n<=19:
    if n == 15 or n == 16:
      return n
    else:
      return 0
  else:
    return n
```

14. **round_sum**

For this problem, we'll round an int value up to the next multiple of 10 if its rightmost digit is 5 or more, so 15 rounds up to 20. Alternately, round down to the previous multiple of 10 if its rightmost digit is less than 5, so 12 rounds down to 10. Given 3 ints, a b c, return the sum of their rounded values. To avoid code repetition, decompose a separate "def round10(num):" below your round_sum() method and call it 3 times.

round_sum(16, 17, 18) → 60
round_sum(12, 13, 14) → 30
round_sum(6, 4, 4) → 10

My solution:

```
def round_sum(a, b, c):
  return round10(a) + round10(b) + round10(c)

def round10(num):
  if num%10>=5:
    return num/10*10 + 10
  else:
    return num/10*10
```

15. **close_far**

Given three ints, a b c, return True if one of b or c is "close" (differing from a by

at most 1), while the other is "far", differing from both other values by 2 or more. Note: abs(num) computes the absolute value of a number.

close_far(1, 2, 10) → True
close_far(1, 2, 3) → False
close_far(4, 1, 3) → True

My solution:

```
def close_far(a, b, c):
  if abs(b-a) <= 1:
    close = b
  elif abs(c-a) <= 1:
    close = c
  else:
    return False

  if (close == b) and (abs(c-a) >= 2) and (abs(c-b)>=2):
    return True
  elif (close == c) and (abs(b-a) >= 2) and (abs(b-c)>=2):
    return True
  else:
    return False
```

String-2

16. **double_char**

Given a string, return a string where for every char in the original, there are two chars.

double_char('The') → 'TThhee'
double_char('AAbb') → 'AAAAbbbb'
double_char('Hi-There') → 'HHii--TThheerree'

My solution:

```
def double_char(str):
  result = ''
  for i in str:
    result += i
    result += i
  return result
```

17. **count_hi**

Return the number of times that the string "hi" appears anywhere in the given

string.

count_hi('abc hi ho') → 1
count_hi('ABChi hi') → 2
count_hi('hihi') → 2

My solution:

```
def count_hi(str):
  a = str.split('hi')
  return len(a) - 1
```

18. **cat_dog**

Return True if the string "cat" and "dog" appear the same number of times in the given string.

cat_dog('catdog') → True
cat_dog('catcat') → False
cat_dog('1cat1cadodog') → True

My solution:

```
def cat_dog(str):
  a = str.split('cat')
  b = str.split('dog')
  if len(a) == len(b):
    return True
  else:
    return False
```

19. **count_code**

Return the number of times that the string "code" appears anywhere in the given string, except we'll accept any letter for the 'd', so "cope" and "cooe" count.

count_code('aaacodebbb') → 1
count_code('codexxcode') → 2
count_code('cozexxcope') → 2

My solution:

```
def count_code(str):
  result = 0
  for i in range(len(str)-3):
    if str[i:i+2] == 'co' and str[i+3] == 'e':
      result += 1
  return result
```

*20.***end_other**

Given two strings, return True if either of the strings appears at the very end of the other string, ignoring upper/lower case differences (in other words, the computation should not be "case sensitive"). Note: s.lower() returns the lowercase version of a string.

end_other('Hiabc', 'abc') → True
end_other('AbC', 'HiaBc') → True
end_other('abc', 'abXabc') → True

My solution:

```
def end_other(a, b):
  a = a.lower()
  b = b.lower()
  return (b.endswith(a) or a.endswith(b))
```

*21.***xyz_there**

Return True if the given string contains an appearance of "xyz" where the xyz is not directly preceeded by a period (.). So "xxyz" counts but "x.xyz" does not.

xyz_there('abcxyz') → True
xyz_there('abc.xyz') → False
xyz_there('xyz.abc') → True

My solution:

```
def xyz_there(str):
  str = str.replace('.xyz', '')
  if 'xyz' in str:
    return True
  else:
    return False
```

List-2

*22.***count_evens**

Return the number of even ints in the given array. Note: the % "mod" operator computes the remainder, e.g. 5 % 2 is 1.

count_evens([2, 1, 2, 3, 4]) → 3

count_evens([2, 2, 0]) → 3
count_evens([1, 3, 5]) → 0

My solution:

```
def count_evens(nums):
  result = 0
  for i in nums:
    if i%2 == 0:
      result += 1
  return result
```

23. **big_diff**

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

My solution:

```
def big_diff(nums):
  maxx = nums[0]
  minn = nums[0]
  for i in range(len(nums)):
    if nums[i] > maxx:
      maxx = nums[i]
    if nums[i] < minn:
      minn = nums[i]
  return maxx - minn
```

24. **centered_average**

Return the "centered" average of an array of ints, which we'll say is the mean average of the values, except not counting the largest and smallest values in the array. 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

My solution:

```
def centered_average(nums):
  total = 0
  number_of_excepts = 2
  centered = nums
  centered.remove(max(nums))
  centered.remove(min(nums))
  for i in centered:
    total += i
  return total/len(centered)
```

*25.***sum13**

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

My solution:

```
def sum13(nums):
  sum = 0
  for i in range(0, nums.count(13)):
    if nums.count(13):
      after = nums.index(13)
      nums.remove(13)
      if after < len(nums):
        nums.pop(after)
  for i in nums:
    sum += i
  return sum
```

*26.***sum67**

Return the sum of the numbers in the array, except ignore sections of numbers starting with a 6 and extending to the next 7 (every 6 will be followed by at least one 7). Return 0 for no numbers.

sum67([1, 2, 2]) → 5
sum67([1, 2, 2, 6, 99, 99, 7]) → 5
sum67([1, 1, 6, 7, 2]) → 4

My solution:

```
def sum67(nums):
  dontadd = 0
  sum = 0
  for i in range(0, len(nums)):
    if dontadd == 0:
      if nums[i] == 6:
        dontadd = 1
      else:
        sum += nums[i]
    else:
      if nums[i] == 7:
        dontadd = 0
      else:
        pass
  return sum
```

27. **has22**

Given an array of ints, return True if the array contains a 2 next to a 2 somewhere.

has22([1, 2, 2]) → True
has22([1, 2, 1, 2]) → False
has22([2, 1, 2]) → False

My solution:

```
def has22(nums):
  indices = []
  for i in range(0, len(nums)):
    if nums[i] == 2:
      indices.append(i)
  for i in range(0, len(indices)-1):
    if indices[i+1] - indices[i] == 1:
      return True
  return False
```