Codingbat Python Questions and Answers Section 1

This document is prepared and can be used only for educational purposes. All questions are taken from http://codingbat.com which contains great questions about Python and Java. Please go to original website and solve questions there. Codingbat also presents a report tool which educators can see students' results.

The answers are done by me in my spare times, Codingbat says all answers are true. I used some of questions in internship application reviews. **You can use questions**, see http://codingbat.com for usage details. **You can use/modify/distribute answers** of me or this document it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Document is prepared with Google Docs and syntax highlighter is GeSHi, which is a great open source tool.

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

Have fun with Python!
Samet Atdag

This document contains 44 questions in these sections:

- Warmup-1
- String-1
- List-1
- Logic-1

Warmup-1

1. **sleep_in**

The parameter weekday is True if it is a weekday, and the parameter vacation is True if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Return True if we sleep in.

sleep_in(False, False) → True
sleep_in(True, False) → False
sleep_in(False, True) → True

My solution:

```
def sleep_in(weekday, vacation):
   if(not weekday or vacation):
       return True
   else:
       return False
```

2. **monkey_trouble**

We have two monkeys, a and b, and the parameters a_smile and b_smile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return True if we are in trouble.

monkey_trouble(True, True) → True
monkey_trouble(False, False) → True
monkey_trouble(True, False) → False

My solution:

```
def monkey_trouble(a_smile, b_smile):
    if((a_smile and b_smile) or ((not a_smile) and (not b_smile))):
        return True
    else:
        return False
```

3. **sum_double**

Given two int values, return their sum. Unless the two values are the same, then return double their sum.

sum_double(1, 2) → 3
sum_double(3, 2) → 5
sum_double(2, 2) → 8

My solution:

```
def sum_double(a, b):
    result = a+b
    if(a == b):
        result = 2*result
    return result
```

4. **diff21**

Given an int n, return the absolute difference between n and 21, except return double the absolute difference if n is over 21.

diff21(19) → 2
diff21(10) → 11
diff21(21) → 0

My solution:

```
def diff21(n):
 if n>21:
   result = 2*(n-21)
 else:
   result = 21-n
 return result
```

5. **parrot_trouble**

We have a loud talking parrot. The "hour" parameter is the current hour time in the range 0..23. We are in trouble if the parrot is talking and the hour is before 7 or after 20. Return True if we are in trouble.

parrot_trouble(True, 6) → True
parrot_trouble(True, 7) → False
parrot_trouble(False, 6) → False

My solution:

```
def parrot_trouble(talking, hour):
 if talking:
   if hour<7 or hour>20:
     return True
   else:
     return False
 else:
   return False
```

6. **makes10**

Given 2 ints, a and b, return True if one if them is 10 or if their sum is 10.

makes10(9, 10) → True
makes10(9, 9) → False
makes10(1, 9) → True

My solution:

```
def makes10(a, b):
 if a==10 or b==10 or a+b==10:
   return True
 else:
   return False
```

7. **near_hundred**

Given an int n, return True if it is within 10 of 100 or 200. Note: abs(num) computes the absolute value of a number.

near_hundred(93) → True
near_hundred(90) → True
near_hundred(89) → False

My solution:

```
def near_hundred(n):
 if abs(100-n)<=10 or abs(200-n)<=10:
   return True
 else:
   return False
```

8. **pos_neg**

Given 2 int values, return True if one is negative and one is positive. Unless the parameter "negative" is True, then they both must be negative.

pos_neg(1, -1, False) → True
pos_neg(-1, 1, False) → True
pos_neg(1, 1, False) → False

My solution:

```
def pos_neg(a, b, negative):
 if negative:
   return (a < 0 and b < 0)
 else:
   return ((a < 0 and b > 0) or (a > 0 and b < 0))
```

9. **not_string**

Given a string, return a new string where "not " has been added to the front. However, if the string already begins with "not", return the string unchanged.

not_string('candy') → 'not candy'
not_string('x') → 'not x'
not_string('not bad') → 'not bad'

My solution:

```
def not_string(str):
 a = str.split('not')
 if len(a) > 1 and a[0]=="":
   return str
 else:
   return "not " + str
```

10. **missing_char**

Given a non-empty string and an int n, return a new string where the char at index n has been removed. The value of n will be a valid index of a char in the original string (i.e. n will be in the range 0..len(str)-1 inclusive).

missing_char('kitten', 1) → 'ktten'
missing_char('kitten', 0) → 'itten'
missing_char('kitten', 4) → 'kittn'

My solution:

```
def missing_char(str, n):
 return str[0:n] + str[n+1:]
```

11. **front_back**

Given a string, return a new string where the first and last chars have been exchanged.

front_back('code') → 'eodc'
front_back('a') → 'a'
front_back('ab') → 'ba'

My solution:

```
def front_back(str):
 if len(str) == 1:
   return str
 elif len(str) == 2:
   return str[1] + str[0]
 else:
   return str[-1:] + str[1:-1] + str[:1]
```

12. **front3**

Given a string, we'll say that the front is the first 3 chars of the string. If the string length is less than 3, the front is whatever is there. Return a new string which is 3

copies of the front.

front3('Java') → 'JavJavJav'
front3('Chocolate') → 'ChoChoCho'
front3('abc') → 'abcabcabc'

My solution:

```
def front3(str):
 if len(str) < 3:
   return str+str+str
 else:
   return str[:3] + str[:3] + str[:3]
```

String-1

13. **hello_name**

Given a string name, e.g. "Bob", return a greeting of the form "Hello Bob!".

hello_name('Bob') → 'Hello Bob!'
hello_name('Alice') → 'Hello Alice!'
hello_name('X') → 'Hello X!'

My solution:

```
def hello_name(name):
 return "Hello " + name + "!"
```

14. **make_abba**

Given two strings, a and b, return the result of putting them together in the order abba, e.g. "Hi" and "Bye" returns "HiByeByeHi".

make_abba('Hi', 'Bye') → 'HiByeByeHi'
make_abba('Yo', 'Alice') → 'YoAliceAliceYo'
make_abba('x', 'y') → 'xyyx'

My solution:

```
def make_abba(a, b):
 return a + b + b + a
```

15. **make_tags**

The web is built with HTML strings like "Yay" which draws Yay as italic text. In this example, the "i" tag makes and which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "Yay".

make_tags('i', 'Yay') → 'Yay'
make_tags('i', 'Hello') → 'Hello'
make_tags('cite', 'Yay') → '<cite>Yay</cite>'

My solution:

```
def make_tags(tag, word):
 return "<" + tag + ">" + word + "</" + tag + ">"
```

16. **make_out_word**

Given an "out" string length 4, such as "<<>>", and a word, return a new string where the word is in the middle of the out string, e.g. "<<word>>".

make_out_word('<<>>', 'Yay') → '<<Yay>>'
make_out_word('<<>>', 'WooHoo') → '<<WooHoo>>'
make_out_word('[[]]', 'word') → '[[word]]'

My solution:

```
def make_out_word(out, word):
 return out[0:2] + word + out[2:]
```

17. **extra_end**

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

extra_end('Hello') → 'lololo'
extra_end('ab') → 'ababab'
extra_end('Hi') → 'HiHiHi'

My solution:

```
def extra_end(str):
 return 3*(str[-2:])
```

18. **first_two**

Given a string, return the string made of its first two chars, so the String "Hello"

yields "He". If the string is shorter than length 2, return whatever there is, so "X" yields "X", and the empty string "" yields the empty string "".

first_two('Hello') → 'He'
first_two('abcdefg') → 'ab'
first_two('ab') → 'ab'

My solution:

```
def first_two(str):
 if len(str) < 2:
   return str
 else:
   return str[0:2]
```

19. **first_half**

Given a string of even length, return the first half. So the string "WooHoo" yields "Woo".

first_half('WooHoo') → 'Woo'
first_half('HelloThere') → 'Hello'
first_half('abcdef') → 'abc'

My solution:

```
def first_half(str):
 return str[0:len(str)/2]
```

20. **without_end**

Given a string, return a version without the first and last char, so "Hello" yields "ell". The string length will be at least 2.

without_end('Hello') → 'ell'
without_end('java') → 'av'
without_end('coding') → 'odin'

My solution:

```
def without_end(str):
 return str[1:-1]
```

21. **combo_string**

Given 2 strings, a and b, return a string of the form short+long+short, with the

shorter string on the outside and the longer string on the inside. The strings will not be the same length, but they may be empty (length 0).

combo_string('Hello', 'hi') → 'hiHellohi'
combo_string('hi', 'Hello') → 'hiHellohi'
combo_string('aaa', 'b') → 'baaab'

My solution:

```
def combo_string(a, b):
 if len(a)<len(b):
   return a+b+a
 else:
   return b+a+b
```

22. **non_start**

Given 2 strings, return their concatenation, except omit the first char of each. The strings will be at least length 1.

non_start('Hello', 'There') → 'ellohere'
non_start('java', 'code') → 'avaode'
non_start('shotl', 'java') → 'hotlava'

My solution:

```
def non_start(a, b):
 return a[1:] + b[1:]
```

23. **left2**

Given a string, return a "rotated left 2" version where the first 2 chars are moved to the end. The string length will be at least 2.

left2('Hello') → 'lloHe'
left2('java') → 'vaja'
left2('Hi') → 'Hi'

My solution:

```
def left2(str):
 return str[2:] + str[0:2]
```

List-1

24. **first_last6**

Given an array of ints, return True if 6 appears as either the first or last element in the array. The array will be length 1 or more.

first_last6([1, 2, 6]) → True
first_last6([6, 1, 2, 3]) → True
first_last6([3, 2, 1]) → False

My solution:

```
def first_last6(nums):
 if (nums[0] == 6) or (nums[len(nums)-1] == 6):
   return True
 else:
   return False
```

25. **same_first_last**

Given an array of ints, return True if the array is length 1 or more, and the first element and the last element are the same.

same_first_last([1, 2, 3]) → False
same_first_last([1, 2, 3, 1]) → True
same_first_last([1, 2, 1]) → True

My solution:

```
def same_first_last(nums):
 if len(nums)>0:
   if nums[0] == nums[len(nums) - 1]:
     return True
   else:
     return False
 else:
   return False
```

26. **make_pi**

Return an int array length 3 containing the first 3 digits of pi, {3, 1, 4}.

make_pi() → [3, 1, 4]

My solution:

```
def make_pi():
 return [3,1,4]
```

27. **common_end**

Given 2 arrays of ints, a and b, return True if they have the same first element or they have the same last element. Both arrays will be length 1 or more.

common_end([1, 2, 3], [7, 3]) → True
common_end([1, 2, 3], [7, 3, 2]) → False
common_end([1, 2, 3], [1, 3]) → True

My solution:

```
def common_end(a, b):
 if (a[0] == b[0]) or (a[len(a)-1] == b[len(b)-1]):
   return True
 else:
   return False
```

28. **sum3**

Given an array of ints length 3, return the sum of all the elements.

sum3([1, 2, 3]) → 6
sum3([5, 11, 2]) → 18
sum3([7, 0, 0]) → 7

My solution:

```
def sum3(nums):
 return nums[0]+nums[1]+nums[2]
```

29. **rotate_left3**

Given an array of ints length 3, return an array with the elements "rotated left" so {1, 2, 3} yields {2, 3, 1}.

rotate_left3([1, 2, 3]) → [2, 3, 1]
rotate_left3([5, 11, 9]) → [11, 9, 5]
rotate_left3([7, 0, 0]) → [0, 0, 7]

My solution:

```
def rotate_left3(nums):
 return [nums[1], nums[2], nums[0]]
```

30. **reverse3**

Given an array of ints length 3, return a new array with the elements in reverse order, so {1, 2, 3} becomes {3, 2, 1}.

reverse3([1, 2, 3]) → [3, 2, 1]
reverse3([5, 11, 9]) → [9, 11, 5]
reverse3([7, 0, 0]) → [0, 0, 7]

My solution:

```
def reverse3(nums):
 return [nums[2], nums[1], nums[0]]
```

31. **max_end3**

Given an array of ints length 3, figure out which is larger between the first and last elements in the array, and set all the other elements to be that value. Return the changed array.

max_end3([1, 2, 3]) → [3, 3, 3]
max_end3([11, 5, 9]) → [11, 11, 11]
max_end3([2, 11, 3]) → [3, 3, 3]

My solution:

```
def max_end3(nums):
 if nums[0]>nums[2]:
   return [nums[0],nums[0],nums[0]]
 else:
   return [nums[2],nums[2],nums[2]]
```

32. **sum2**

Given an array of ints, return the sum of the first 2 elements in the array. If the array length is less than 2, just sum up the elements that exist, returning 0 if the array is length 0.

sum2([1, 2, 3]) → 3
sum2([1, 1]) → 2
sum2([1, 1, 1, 1]) → 2

My solution:

```
def sum2(nums):
 if len(nums) == 0:
   return 0
 elif len(nums) == 1:
   return nums[0]
 else:
   return nums[0] + nums[1]
```

33. **middle_way**

Given 2 int arrays, a and b, each length 3, return a new array length 2 containing their middle elements.

middle_way([1, 2, 3], [4, 5, 6]) → [2, 5]
middle_way([7, 7, 7], [3, 8, 0]) → [7, 8]
middle_way([5, 2, 9], [1, 4, 5]) → [2, 4]

My solution:

```
def middle_way(a, b):
 return [a[1],b[1]]
```

34. **make_ends**

Given an array of ints, return a new array length 2 containing the first and last elements from the original array. The original array will be length 1 or more.

make_ends([1, 2, 3]) → [1, 3]
make_ends([1, 2, 3, 4]) → [1, 4]
make_ends([7, 4, 6, 2]) → [7, 2]

My solution:

```
def make_ends(nums):
 return [nums[0], nums[-1]]
```

35. **has23**

Given an int array length 2, return True if it contains a 2 or a 3.

has23([2, 5]) → True
has23([4, 3]) → True
has23([4, 5]) → False

My solution:

```
def has23(nums):
 if 2 in nums or 3 in nums:
   return True
 else:
   return False
```

Logic-1

36. **cigar_party**

When squirrels get together for a party, they like to have cigars. A squirrel party is successful when the number of cigars is between 40 and 60, inclusive. Unless it is the weekend, in which case there is no upper bound on the number of cigars. Return True if the party with the given values is successful, or False otherwise.

cigar_party(30, False) → False
cigar_party(50, False) → True
cigar_party(70, True) → True

My solution:

```
def cigar_party(cigars, is_weekend):
 if is_weekend:
   if cigars>=40:
     return True
   else:
     return False
 else:
   if cigars>=40 and cigars<=60:
     return True
   else:
     return False
```

37. **date_fashion**

You and your date are trying to get a table at a restaurant. The parameter "you" is the stylishness of your clothes, in the range 0..10, and "date" is the stylishness of your date's clothes. The result getting the table is encoded as an int value with 0=no, 1=maybe, 2=yes. If either of you is very stylish, 8 or more, then the result is 2 (yes). With the exception that if either of you has style of 2 or less, then the result is 0 (no). Otherwise the result is 1 (maybe).

date_fashion(5, 10) → 2
date_fashion(5, 2) → 0
date_fashion(5, 5) → 1

My solution:

```
def date_fashion(you, date):
 if you<=2 or date<=2:
   return 0
 elif you>=8 or date>=8:
   return 2
 else:
   return 1
```

38. **squirrel_play**

The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is between 60 and 90 (inclusive). Unless it is summer, then the upper limit is 100 instead of 90. Given an int temperature and a boolean is_summer, return True if the squirrels play and False otherwise.

squirrel_play(70, False) → True
squirrel_play(95, False) → False
squirrel_play(95, True) → True

My solution:

```
def squirrel_play(temp, is_summer):
 upper = 90
 if is_summer:
   upper = 100
 return (temp>=60 and temp<=upper)
```

39. **caught_speeding**

You are driving a little too fast, and a police officer stops you. Write code to compute the result, encoded as an int value: 0=no ticket, 1=small ticket, 2=big ticket. If speed is 60 or less, the result is 0. If speed is between 61 and 80 inclusive, the result is 1. If speed is 81 or more, the result is 2. Unless it is your birthday -- on that day, your speed can be 5 higher in all cases.

caught_speeding(60, False) → 0
caught_speeding(65, False) → 1
caught_speeding(65, True) → 0

My solution:

```
def caught_speeding(speed, is_birthday):
 gift = 0
 if is_birthday:
   gift = 5
 if speed <= 60+gift:
   return 0
 elif speed >= 81+gift:
   return 2
 else:
   return 1
```

*40.***sorta_sum**

Given 2 ints, a and b, return their sum. However, sums in the range 10..19 inclusive, are forbidden, so in that case just return 20.

sorta_sum(3, 4) → 7
sorta_sum(9, 4) → 20
sorta_sum(10, 11) → 21

My solution:

```
def sorta_sum(a, b):
 total = a+b
 if total > 9 and total < 20:
   return 20
 else:
   return total
```

*41.***alarm_clock**

Given a day of the week encoded as 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating if we are on vacation, return a string of the form "7:00" indicating when the alarm clock should ring. Weekdays, the alarm should be "7:00" and on the weekend it should be "10:00". Unless we are on vacation -- then on weekdays it should be "10:00" and weekends it should be "off".

alarm_clock(1, False) → '7:00'
alarm_clock(5, False) → '7:00'
alarm_clock(0, False) → '10:00'

My solution:

```
def alarm_clock(day, vacation):
 weekday_alarm = "7:00"
 weekend_alarm = "10:00"
 if vacation:
   weekday_alarm = "10:00"
   weekend_alarm = "off"
 if day>0 and day<6:
   return weekday_alarm
 else:
   return weekend_alarm
```

*42.***love6**

The number 6 is a truly great number. Given two int values, a and b, return True if either one is 6. Or if their sum or difference is 6. Note: the function abs(num) computes the absolute value of a number.

love6(6, 4) → True
love6(4, 5) → False
love6(1, 5) → True

My solution:

```
def love6(a, b):
 if a==6 or b==6 or a+b==6 or abs(a-b)==6:
   return True
 return False
```

*43.***in1to10**

Given a number n, return True if n is in the range 1..10, inclusive.
Unless "outsideMode" is True, in which case return True if the number is less or equal to 1, or greater or equal to 10.

in1to10(5, False) → True
in1to10(11, False) → False
in1to10(11, True) → True

My solution:

```
def in1to10(n, outside_mode):
  if not outside_mode:
    return (n>=1 and n<=10)
  else:
    return (n<=1 or n>=10)
```

44. **near_ten**

Given a non-negative number "num", return True if num is within 2 of a multiple of 10. Note: (a % b) is the remainder of dividing a by b, so (7 % 5) is 2.

near_ten(12) → True
near_ten(17) → False
near_ten(19) → True

My solution:

```
def near_ten(num):
 return (num%10==0 or num%10==1 or num%10==2 or abs(10-num%10)==2 or abs(10-
num%10)==1 or abs(10-num%10)==0)
```