Data Bootcamp: Code Practice #2

Revised: February 10, 2016

Answer each of the questions below. What you hand in can be code or handwritten or something else, but it must be readable and professional. Hardcopy only.

1. *Review.* Does this code run without error? If so, what does it produce? If not, explain why.

```
x = [1, 2, 3]
y = 'bootcamp'
z = x + y
```

2. *Review.* For the same `x` and `y` as the previous question: What function tells us what type they are? What function tells us how many elements they contain?

3. What type is each expression? How can you tell?

```
2
'2'
2.0
"2.0"
2>1
'Itamar' > 'Chase'
[1, 2]
(1, 2)
{1: one, 2: two}
```

4. What value does each of these comparisons have?

```
1>=0
1 >= 1
1 > 1
1==1
1 == 1.0
'Spencer' == "Spencer"
2**3 > 3**2
1 >= 0 or 1 <= 2
1 >= 0 and 1 <= 2
```

5. Does this code run without error? If so, what does it produce? If not, how would you fix it?

```
if 2>1
   print('Yes, 2 is still greater than 1')
```

6. What is the result of running this code? Why?

```
if True:
    print('on the one hand')
else:
    print('but on the other hand')
```

   What happens if we replace `True` with `False` in the first line? What happens if we insert the word `not` after `if` in the first line?

7. What is the result of running this code?

```
cond = True
if cond:
    x = "Chase"
else:
    x = "Dave"
print(x)
```

8. Suppose we have two lists, `x = [1, 2, 3, 4]` and `y = ['x', 'y', 'z']`. Adapt the code below to determine which has more elements:

```
if <insert expression>:
    print('x has more')
else:
    print("y has at least as many")
```

9. Explain in words what slicing does.

10. How would you extract ("slice") the first element (the integer `1`) from the list `x` below? The last element? All but the last element?

```
x = [1, 2, 3, 4, 5]
```

11. Use slicing to extract each word from

```
sentence = 'This is a sentence; please slice it.'
```

   Suggestion: Number every character in `sentence` by hand.

12. Consider the list

```
x = [1, 2, "a", 'b', "fast", 'slow', 3, "Raghu", 'Liuren', 10]
```

   (a) How would you slice out the first item? The last item?

   (b) How would you slice out the items from `'b'` to `3` inclusive?

13. Using the same list `x`, write a loop that prints every element on a new line.

14. *Challenging.* Using the same list `x`, write a loop that prints every element of type `str`.

15. Use Spyder's help to find out what the range function does. How would you describe `range(3,12,2)`? Verify by converting to a list with `list(range(3,12,2))`.

16. *Challenging.* Write a loop that sums the integers from zero to thirty that are multiples of three: 3, 6, etc.

17. Define a function `pocket_change()` that takes four integers as inputs (numbers of pennies, nickels, dimes, and quarters in your pocket) and returns a floating point number (their dollar value). Run your program with the input `(1, 2, 3, 4)`. *Bonus (optional):* Report the value with a dollar sign.

18. *Challenging.* Write a function `notsix()` that takes a list of integers and returns a (shorter) list of only those that do not begin with a 6. Test it on the list `[1234, 6783, 6, 4321, 9876]`. *Hints:* You can create a blank list with `x = []`. You can append `item` to it with `x.append(item)`.

19. *Challenging.* Explain what this code does:

```
old_list = [1234, 6783, 6, 4321, 9876]
new_list = [x for x in old_list if str(x)[0] != "6"]
```

20. Consider the Python object

```
z = {1: 'one', 2: 'two', 3: 'three'}
```

    (a) What kind of object is `z`? What is its length?
    (b) Which components are keys? Which are values?
    (c) How would I get the value associated with the key 2?
    (d) Use Spyder's help facilities to figure out what `z.keys()` does. Ditto `z.values()`. Try them to verify.
    (e) What does `list(z.keys())` do?
    (f) What does `list(z.values())` do?
    (g) What does `list(z)` do?