CS 111 - Winter 2020
Quiz 1 Practice Questions

The quiz will be worth 30 points, and each point should correspond roughly to one minute of your time. This list of practice questions is not necessarily representative of the length of the quiz – it is hopefully longer to give you more practice, but there might be more questions in a given category on the actual quiz.

True/False questions (approximately 2 points)

1) Computer science is the study of what can be computed.

2) In Python, the expression `3 + 4.0` evaluates to `7`.

3) In Python, the expression `"a" * "3"` evaluates to `"aaa"`.

4) The last character of a string `s` is at position `len(s)-1`.

5) The `split` method breaks a string into a list of substrings, and `join` does the opposite.

Multiple-choice questions (approximately 4 points)

1) Which of the following is *not* a valid identifier name?
(a) `main` **(b)** `_var` **(c)** `ch` **(d)** `in`

2) The binary (base-2) number `10110` corresponds to which decimal (base-10) number?
(a) `22` **(b)** `13` **(c)** `10110` **(d)** `11`

Code prediction (approximately 8 points)

1) Show the sequence of numbers that be generated by the following range expression.

```
range(15,5,-2)
```

2) Using the formula $a = (a//b)(b) + (a\%b)$, predict the output of the following two expressions:
(a) `8 // -3`
(b) `8 % -3`

3) Predict the output that would be generated by the following code.

```
res = 0
for i in range(1,5):
    res = res + i*i
    print(i)
print(res)
```

4) Predict the output that would be generated by the following code.

```
for w in "Mississippi".split("i"):
    print(w, end=" ")
```

5) The list method `insert` inserts a new element in the list at the specified position, pushing everything else back one position. Here is an example interaction using the shell (interpreter):

```
>>> words = ["apple", "banana", "cat", "dog"]
>>> words.insert(3, "coconut")
>>> print(words)
["apple", "banana", "cat", "coconut", "dog"]
```

Predict the output that would be generated by the `print` statement on the last line of the following code.

```
mylist = ["CS", 111, "applebananacatdog"]
mylist.insert(1, 3.14)
print(mylist)
```

6) As we saw in lesson 5, you can store a list as an element of a list. For example:

```
>>> wordPairs = [["apple", "ant"], ["banana", "baboon"], ["cat", "coconut"]]
>>> firstPair = wordPairs[0]
>>> print(firstPair)
["apple", "ant"]
>>> print(firstPair[1])
"ant"
```

Given the “nested” list `wordPairs`, predict the output that would be printed by the following code:

```
for i in range(len(wordPairs)):
    pair = wordPairs[i]
    print(i, pair[1], pair[0])
```

7) Recall that we can use the built-in functions `ord` and `chr` to convert between characters in strings and their numeric Unicode representations:

```
>>> val = ord("a")
>>> print(val)
97
>>> ch = chr(122)
>>> print(ch)
z
```

Predict the output that would be printed by the following code:

```
def main():
    # For now, we'll hard-code the message and key
    plaintext = "abcxyz"
    key = 2

    msg = ""
    for ch in plaintext:
        val = ord(ch)

        newval = (((val - 97) + key) % 26) + 97
        print("New value:", newval)

        msg = msg + chr(newval)

    print("The message is:", msg)

main()
```

Rewriting code (approximately 6 points)

1) The following code could be written much more concisely with a `for` loop. Write code to replace the code below, using a `for` loop. For full points, make sure not to hard-code (i.e., use the actual number in the code, when you can determine it using a variable instead) the length of the list.

```
total = 0
myList = [3, 1, 4, 1, 5, 9]
val = myList[0]
total = total + val
print("Value", val, "is at position", 0)
val = myList[1]
total = total + val
print("Value", val, "is at position", 1)
val = myList[2]
total = total + val
print("Value", val, "is at position", 2)
val = myList[3]
total = total + val
print("Value", val, "is at position", 3)
val = myList[4]
total = total + val
print("Value", val, "is at position", 4)
val = myList[5]
total = total + val
print("Value", val, "is at position", 5)
print("The total is", total)
```

Fixing bugs (approximately 4 points)

1) The following code is supposed to calculate the volume of a sphere from its radius, given as input. The formula for the volume of a sphere is given by $V = 4/3\pi r^3$. Unfortunately, there are three bugs in the code. What are they?

```
import math

def main():
    # Get the radius from the user as input
    radius = input("What is the radius of the sphere? ")

    # Calculate the volume
    volume = (4 // 3) * math.pi * (radius * radius * radius)

    # Report the answer to the user
    print("The volume of a sphere with radius " + radius + " is " + volume)

main()
```

Writing code (approximately 6 points)

1) A Fibonacci sequence is a sequence of numbers where each successive number is the sum of the previous two. The classic Fibonacci sequence begins: 1, 1, 2, 3, 5, 8, 13, ...

Write a program that computes the nth Fibonacci number where n is a value input by the user. For example, if $n = 6$, then the result is 8.

2) The `string` method `title` returns a copy of the provided string with the first character of each word capitalized, but does not modify the original string. For example:

```
>>> s = "this is my string"
>>> caps = s.title()
>>> print(s)
this is my string
>>> print(caps)
This Is My String
```

Given the string `cold`, write code that would generate the output shown below. You can use more than one statement (i.e., more than one line of code), but you should store the resulting string in the variable `message`.

```
>>> cold = "my nose is froze"
>>> message = <your code here>
>>> print(message)
My Frozen Nose
```