**Programming Concepts review**

**0 Introduction**

1. A \_\_\_GUI\_\_\_\_\_\_ allows you to talk to your computer without knowing a programming language.

2. When working interactively in the \_command line or shell\_\_\_\_\_, each input line starts with a \_\_\_\_command prompt\_\_\_\_\_\_\_, which may look like >, >>>, or $ (or something else entirely) with a space after it.

3. You can also write a \_\_\_script\_\_\_\_: a file with many lines of code in it to be executed together.

**1 Files and the Filesystem**

1.

/Users/christina/Documents/my\_project/data.csv

/home/christina/my\_project/data.csv

C:\Users\christina\Documents\my\_project\data.csv

are examples of \_\_absolute\_\_ paths

2.

../my\_project/data/data.csv

data.csv

data/data.csv

are examples of \_\_\_relative\_\_ paths

3. In file paths:

.. means \_\_\_\_\_\_\_\_\_\_go up one directory\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

~ means \_\_\_\_\_\_\_\_\_\_\_\_home directory\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. To retrieve information from a file, \_\_\_\_\_read\_\_ from it. To put information into a file, \_\_\_write\_\_\_ to it.

5. CSV (comma separated values) files are one type of plain text file. Plain text files have no \_\_\_\_\_formatting\_\_\_\_ such as bold text, colors, or fonts.

**2 Data Types**

1. Boolean variables can be either \_\_\_true\_\_\_ or \_\_\_\_false\_\_\_. When converted to an integer, \_\_\_false\_\_\_ becomes 0 and \_\_true\_\_\_ becomes 1.

2. Tabs, spaces, and new line characters are examples of \_\_\_whitespace\_ characters.

\n is a \_\_\_new line\_\_\_

\t is a \_\_\_tab\_\_\_\_

3. A string without any characters in it (length 0) is called an \_\_\_empty\_\_ string.

4. Strings must be surrounded by \_\_quotation marks\_\_. In R and Python, single or double \_\_quotation marks\_\_ can be used, but they must match. Pick one style and be consistent where possible!

5. "north" is a \_\_\_substring\_\_ of "northwestern".

6. Concatenating strings means to \_\_\_join or add\_\_\_ the strings together.

**3 Variables**

1. <- in R, and = in Python, are \_\_assignment\_\_\_ operators. The name of the variable goes on the \_\_\_left\_\_\_ side, and the value goes on the \_\_\_right\_\_\_. Everything on the \_\_\_right\_\_\_ side is evaluated first before the value is assigned to the variable.

2. In R and Python, a variable with name age\_list is **[not the same ]** as a variable with name Age\_list.

3. If you run this code:

x = 3

x + 2

the value of x at the end will be \_3\_\_\_\_.

4. If you run this code:

x = 3

y = x

x = x + 1

y = y + 2

the value of y at the end will be \_\_\_5\_\_\_\_\_\_.

**4 Lists, Vectors, Arrays**

1. Elements in a list are stored **[in the order given ]**

2. The first element of a Python list has the index of \_\_\_0\_\_\_ .

3. The \_\_\_length\_\_ of a list, vector, or array is the number of elements in it. An empty list has a \_\_\_\_length\_\_ of 0.

4. You can \_\_\_\_prepend\_\_\_ an item to the beginning of a list or vector or \_\_\_append\_\_\_ an item to the end.

5. Sometimes, lists can be \_\_\_nested\_\_ inside other lists.

**5 Conditions**

1. The operator to test for equality is \_\_\_==\_\_\_.

2. Is the following **[ True]**; (TRUE and FALSE) or (not FALSE and TRUE)

**6 Flow Control**

1. If statements determine what to do based on a condition that evaluates to **[ a single ]** True or False value(s).

2. Where will the \* in cell B3 of Figure 1 below end up if you execute the following statements?

if the space to your right is occupied

move one space up

move one space right

else

move one space right

if you are in column B

move one space to your left

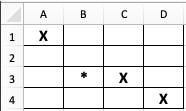
else if you are in column C

move one space up

else

move one space down

FIGURE 1.



**End in C1**

**7 For Loops**

1. Loops are used to \_repeat or run\_\_\_ the same code for \_\_multiple\_\_\_ values.

2. The following code will print \_\_4\_\_ numbers.

x = [1, 4, 3, 6, 7, 2]

for i in x

if i < 5

print i

**8 Functions**

1. The values you send to a function are called \_\_arguments\_\_\_, while the variables that are defined in a function definition are called \_\_\_parameters\_\_\_.

2. The \_\_order\_\_\_ of non-keyword arguments must match the \_\_order\_\_\_ of parameters in the function definition.

3. Parameters without default values in a function definition are **[ required ]**.

4. You can **[ always ]** use the parameter name with all of the arguments when calling a function.

5. The output of a function is called the \_\_\_return\_\_\_ value.

6. Packages/libraries/modules need to be \_\_\_imported\_\_ before using them in every script or session. Some are built-in, while others need to be \_\_\_\_installed\_\_\_ first.