Python Worksheet I

Welcome to your first python worksheet! Each week you will be given a new worksheet that will cover the topics we covered in class, as well as going over some previous stuff, just to make sure you are getting all this!

Part I - Computer Components

1) Write a couple of sentences to explain what each of these computer components is for
    a) CPU
    b) Graphics card / GPU
    c) RAM / memory
    d) HDD / Hard drive
    e) Motherboard

Part II - Commands

1) So far we've covered a few python commands, but there are lots of others. Research the following commands and try to explain what each is for. How many arguments does each take? Give an example for each one. **Bonus - what *types* are the arguments for each command?**
    a) `print()`
    b) `abs()`
    c) `dir()` hint: make a variable and then try dir()
    d) `pow()`
    e) `round()`
    f) `type()`
    g) `help()`

Part III - Numbers

1) Python is also a calculator. Use the python shell to find the answers to the following questions:
    a) 23 * 56 =
    b) 23 / 56 =
    c) 573.36 + 289.03 =
    d) -9 + 20 =
    e) 3.14 + 2.5 * 82.13 =
    f) 4.7 + 2.3 =

2) Remember, we have 2 types of numbers in python: **int**, which stands for integer or whole number, and **float**, which is a number with a decimal point. Here are some examples:

```
int : 2, 45, -23, 0, 256
float : 4.3, 6.09, -78.83, 0.0034
```

For each of the answers in question 1) say whether it is an **int** or a **float**.

3) Python can tell us whether a number is an int or a float when we use the `type()` command. Here is an example when we use `type()` on a float. Try it.

```
>>> type(3.14)
<class 'float'>
>>>
```

The keyword `class` is just another name for `type`. It is saying the type of **3.14** is a `float`. Check your answers for question 2 using the `type()` command. Were you correct?

4) Python can also handle more advanced math functions. For each question below, find the answer using a python command. Also give the command you used. The first is done for you. Remember to use import math first to tell python we are going to be using math commands. If you are having trouble, the math commands are listed below.
   a) Round up 4.6
      i) `>>> math.ceil(4.6)`
   b) Round down 76.43
   c) Round up 89.98
   d) Round 10.5
   e) $\sqrt{49}$ =
   f) cos(12) =
   g) sin(45) =

```
math.ceil(), math.floor(), math.sqrt(), round(), math.cos(),
math.sin()
```

5) So far we have been using commands which only take one argument, but some commands take more, and others don't have any arguments. For example the `help()` command doesn't need any arguments at all. Now let's have a look at the `pow()` command. Remember from your math classes the exponent of a number says how many times to multiply that number by itself. For example

$$8^2 = 8 \times 8 = 64$$

Here, 8 is the number and 2 is the exponent. We can do this in python with the `pow()`

command:

```
>>> pow(8, 2)
64
```

All we need to do is separate the arguments with a comma. Use the `pow()` command to find the answer to these:

a) 3^3
b) 5^2
c) 7^4
d) 2^8
e) 2^{-1}
f) $16^{0.5}$

Part IV - Variables

1) A variable is a container we can put things in. We can store all types of stuff in a variable, but for now we'll just look at storing numbers. To make a variable, we need to choose a name and put something in it.

   Make a variable apples and assign it the number 7. Use the `print()` command to see how many apples we have:

   ```
   >>> apples = 7
   >>> print(apples)
   ```

   Change the number of apples to 12 and `print()` again. Try it with some other numbers. Also, once we've created **`apples`**, we can just type **`apples`** to see what it holds.

2) A variable name must start with a letter, and can only contain letters, numbers or the '_' character. Which of the following are valid variable names? Try to create them in python by copying them into the python shell and assigning the number 3 to them.
   a) **`Apples`**
   b) **`appLES`**
   c) **`apples3`**
   d) **`apples 3`**
   e) **`3apples`**
   f) **`Apples_3`**
   g) **`applesAreNice`**
   h) **`I_have_3_apples`**
3) Create 3 variables, **`apples`**, **`oranges`** and **`bananas`** and assign them the values 3, 4 and 5 respectively:

   ```
   >>> apples = 3
   >>> oranges = 4
   ```

```
>>> bananas = 5
```

Use `print(apples, oranges, bananas)` to see how many of each we have. We can assign a variable another variable. In python, type

```
>>> apples = oranges
```

How many apples do we have? Now, in the following examples, type the code and answer the questions. Try to answer the questions first and then use the print command to check your answer.

```
>>> apples = 4
>>> oranges = 5
>>> bananas = apples
```

a) How many apples do we have?
b) How many bananas do we have?

```
>>> oranges = 7 - 4
>>> apples = 2 * 8
```

c) How many oranges do we have?
d) How many apples do we have?

```
>>> apples = 7
>>> oranges = apples + 2
```

e) How many oranges do we have?

```
>>> oranges = 4
>>> apples = oranges * 2
```

f) How many oranges do we now have?

```
>>> bananas = 8
>>> bananas = bananas + 2
```

g) How many bananas do we have?

```
>>> oranges = 8
>>> apples = 3
>>> bananas = oranges - apples
```

h) How many bananas do we have?

```
>>> oranges = 8
>>> apples = 3
>>> bananas = oranges * apples
```

i) How many bananas do we have now?

Part V - Strings

1) A string is a list of characters. We declare a string by using double quotes (“”). Have a go at printing these strings:
    a) `print(“Hello world!”)`
    b) `print(“I am another string”)`
    c) `print(“You can put nearly anything you like in a string”)`
    d) `print(“But some characters like the “ require special care”)`
2) What happened in 1) d) ? The quotation is used to begin and end a string, but what if we want to use in inside the string? To do this we need to precede it with a backslash \. Try it!

```
>>> print(“I can use a \” in a string!”)
I can use a “ in a string!
```

Use the print() command to display the following strings
    a) `Coding is fun`
    b) `Coding “is” fun`
    c) `“Welcome” the sign read, as we entered the rundown mansion.`
    d) `“Hi Emily,” I said. “How was your day?”`

3) We can even join strings using the (+) operator. Try it!

```
>>> print(“Hello ” + “World”)
```

Notice the space at the end of the first string. If we didn’t have that, the words would run together and you’ll end up with `HelloWorld`.

4) We can also repeat a string using the (*) operator. Try this:

```
>>> print(“bouncy, ” * 10)
```

5) In the python shell, assign these string to the following variables:

```
>>> str_1 = “I want the ”
>>> str_2 = “bouncy ”
>>> str_3 = “ball”
```

Here we use str which is short for string. Now, using the (+) and (*) operators, write a print command that will display the following text. The first is done for you.
    a) `I want the ball`
        i) Answer: `print(str_1 + str_3)`
    b) `bouncy ball`
    c) `I want the bouncy ball`

d) `I want the bouncy bouncy ball`
e) `I want the bouncy bouncy bouncy bouncy bouncy bouncy ball`

Part VI - Errors and Debugging

1) In each of these lines of code, an error has been made. If we copy the code into python and try to run it we see it gives us red text. Find and fix the error for each example. Write out what the code should be
    a) `prints("I so got this!")`
    b) `myColour = "Red`
    c) `print("One potato ", "two potato  "three potato more!")`
    d) `theNumberTen = 1 0`
    e) `Print("Almost finished!")`
    f) `prinT(""Oh,\" I said. \"This is one's a little harder/"")`