1. Input and output:

We put an end=' ' at the end of print line. This tells print to not end the line with a newline character and go to the next line.

print("How old are you?"') # will add the newline

Write a code that will put the input and answers in the same line:

"How tall are you?"

"How much do you weigh?"

Now put show the answers in the following format:

So, you're \_\_\_\_\_\_\_\_(10 character wide, center) old, \_\_\_\_\_\_\_\_\_\_(10 character wide, center) tall and \_\_\_\_\_\_\_\_\_ (10 character wide, center) heavy."

2. We did not discuss this in the class, but I think you can figure this out.

from sys import argv

# google this part

script, first, second, third = argv

print("The script is called:", script)

print("Your first variable is:", first)

print("Your second variable is:", second)

print("Your third variable is:", third)

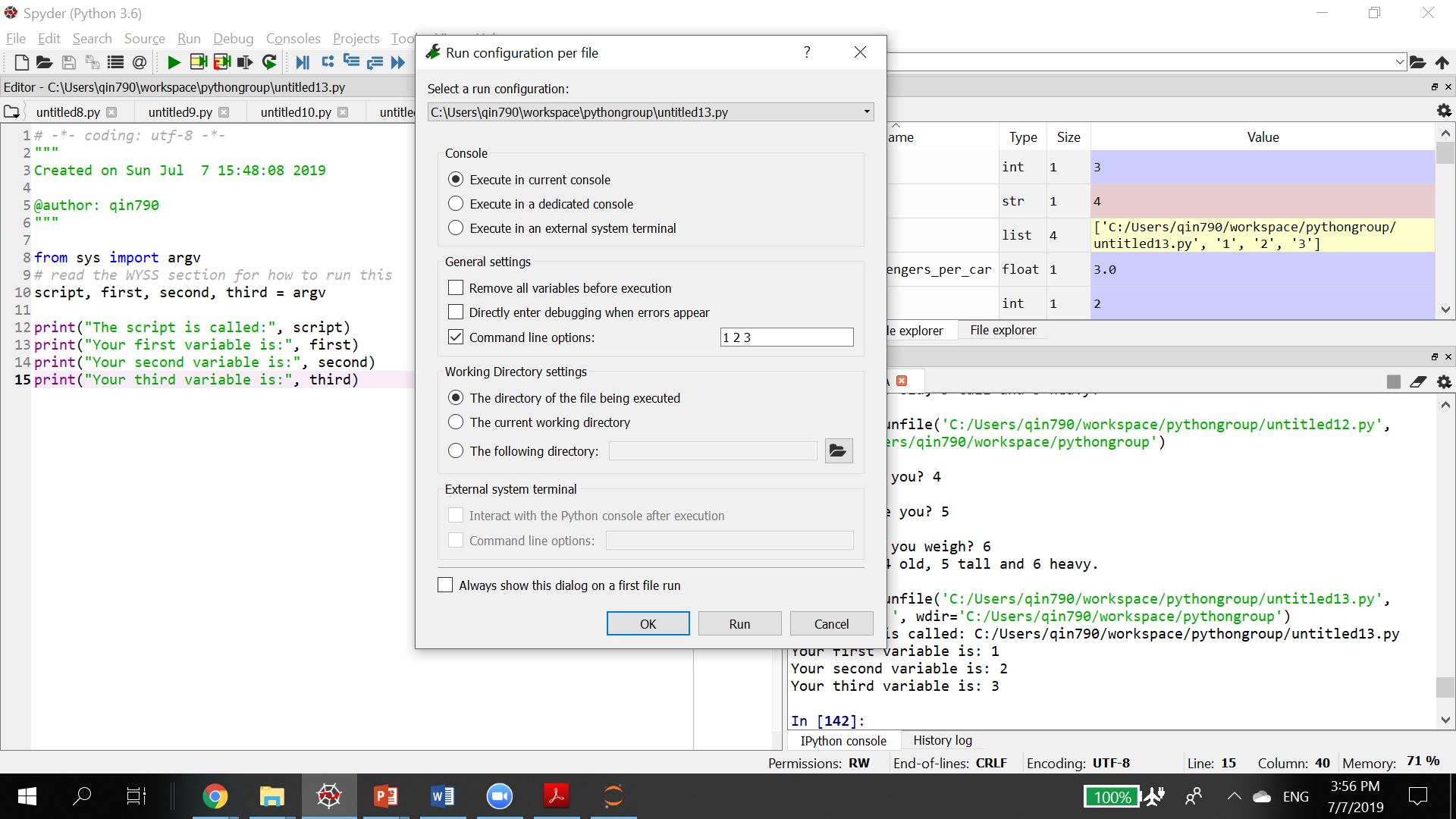
Note:

On line 1 we have what’s called an *import*. This is how you add features to your script from the Python feature set. Rather than give you all the features at once, Python asks you to say what you plan to use. This keeps your programs small, but it also acts as documentation for other programmers who read your code later.

The argv is the *argument variable*, a very standard name in programming that you will find used in many other languages. This variable *holds* the arguments you pass to your Python script when you run it. In the exercises you will get to play with this more and see what happens.

Line 3 *unpacks* argv so that, rather than holding all the arguments, it gets assigned to four variables you can work with: script, first, second, and third. This may look strange, but “unpack” is probably the best word to describe what it does. It just says, “Take whatever is in argv, unpack it, and assign it to all of these variables on the left in order.”

To run this is spyder, you need to set up the run configuration file, under ‘run’ menu:



Then in the command line option, enter 1 2 3

The 1 2 3 will be passed to the argv, and get printed out.

3. Write a function named times\_ten. The function should accept an argument and display the

product of its argument multiplied times 10.

4. Examine the following function header, then write a statement that calls the function, passing 12 as an argument.

def show\_value(quantity):

5. Look at the following function header:

def my\_function(a, b, c):

Now look at the following call to my\_function:

my\_function(3, 2, 1)

When this call executes, what value will be assigned to a? What value will be assigned to b?

What value will be assigned to c?

6. What will the following program display?

def main():

x = 1

y = 3.4

print(x, y)

change\_us(x, y)

print(x, y)

def change\_us(a, b):

a = 0

b = 0

print(a, b)

main()

Exercise for more experienced students

1. Automobile Costs

Write a program that asks the user to enter the monthly costs for the following expenses

incurred from operating his or her automobile: loan payment, insurance, gas, oil, tires, and

maintenance. The program should then display the total monthly cost of these expenses, and the total annual cost of these expenses.

2. Stadium Seating

There are three seating categories at a stadium. Class A seats cost $20, Class B seats cost $15, and Class C seats cost $10. Write a program that asks how many tickets for each class of seats were sold, then displays the amount of income generated from ticket sales.

3. Prime Number List

Please write the is\_prime function. Write another program that displays all of the prime numbers from 1 to 100. The program should have a loop that calls the is\_prime function.

4. Random Number Guessing Game

Write a program that generates a random number in the range of 1 through 100, and asks the

user to guess what the number is. If the user’s guess is higher than the random number, the

program should display “Too high, try again.” If the user’s guess is lower than the random

number, the program should display “Too low, try again.” If the user guesses the number, the

application should congratulate the user and generate a new random number so the game can

start over.

Optional Enhancement: Enhance the game so it keeps count of the number of guesses that the

user makes. When the user correctly guesses the random number, the program should display

the number of guesses.

5. Passing by reference:

In class, we talked about passing by value. For the mutable objects, like list, Python is passing by reference. Run the sample code and try to understand why.

Here is a sample code:

def reassign(list):

list = [0, 1] #list is the local variable, will not change the arg value

def append(list):

list.append(1) #the arg is not changed, but the content changed

list = [0]

reassign(list)

print ("reassign, no content change on list=", list)

append(list)

print ("content changed", list)

Now write a function that passes the following arguments, and show if they are passing by reference:

1. Pass a tuple: like (1,2,3,4)
2. Pass a string: like “ hello world”
3. Pass a set, like {“apple”, “banna”, “orange”)
4. Pass a dictionary { “john”: 30, “Doe”: 40, “Henry”: 50}