**Unit 5:** Data Structures

**Lab 1: Tuples**

| **Name:** |  | | |
| --- | --- | --- | --- |

| **1.**  Open the **U5L1 Tuples** Replit project and clear out the code from the class exploration. | |
| --- | --- |
| **2.**  Create a variable named fruits and set it to a **tuple** containing the strings "banana", "grape", "kiwi", and "pear", in that order.  Then print the tuple to make sure you got everything. [check](#_ybiyq7zd160a) | |
| **a.** What is fruits[0]?  **b.** What is the third element of fruits?  **c.** What element is at index 3?  **d.** What happens if you try *updating* the element at index 1, like this: fruits[1] = "apple" and why?  **e.** What will happen if you try to print(fruits[4]) and why? | a.  b.  c.  d.  e. |
| **3. Write some test code to check your answers:**  For **a), b),** and **c)**, print out each element to see if you’re correct, e.g: print(fruits[0])  For **d)**, try modifying the element at index 2 as shown *(then remove this code).*  For **e)**, try it and see what happens *(then remove this code).* | |
| **Copy/paste your test code below:** [Let me confirm my test code](#_nwufkl7is2mn)[Check answers for a-e](#_t7bpz59z8t5w) | |

| **5.** Delete or comment out your code so far. Start again with this **tuple**:  fruits = ("banana", "grape", "kiwi", "pear")  Add some code below this list to print out each element in the **tuple** using a **for loop**; it should result in printed output that looks like this: | |
| --- | --- |
| **Copy/paste the for loop you wrote below:** [check](#_ezwnfb5dha4l) | |

| **Remove or comment out all code so far in main.py**  **6.** Write a function named distance that takes *two* parameters, p1 and p2, which are both **tuples** that represent (x, y) coordinates.  Here is the function header you should use: def distance(p1, p2):  Your function should calculate and return the **distance** between the two points. Do a quick google search for "distance formula" if you don't remember it!  Also, friendly reminder that you have access to a sqrt function by importing math  **TEST CODE:**  dist1 = distance((3, 4), (-5, -10))  print(dist1)  dist2 = distance((-16, 5), (17, -2))  print(dist2)  **Expected output:** |
| --- |
| **copy/paste your distance function here:**[sample solution](#_czhvgp2hpugz) |

| **Clear out the *test code* from above (but leave your distance function).**  **7.** Now write a small interactive program in which you ask the user to input the two coordinates as x and y values, store the input as two tuples, and use your function to tell the user the distance between their chosen points.  **Sample input/output:** |
| --- |
| **copy/paste your full code here**[sample solution](#_d9tw7115f16g) |

| **8.** Recall that a color can be represented as a mixture of 3 values -- red, green, and blue, or **RGB** -- each ranging between 0 and 255.  Here some examples of colors and their respective "RGB" values:    Write a function, random\_RGB, that has no parameters and returns a **tuple** containing three elements, each a random number between 0 and 255.  **TEST CODE:**  rand\_rgb\_1 = random\_RGB()  rand\_rgb\_2 = random\_RGB()  rand\_rgb\_3 = random\_RGB()  print(rand\_rgb\_1)  print(rand\_rgb\_2)  print(rand\_rgb\_3)  **Expected output (will be random, but should look something like this):**    **Run it again and you should get different values:** |
| --- |
| **copy/paste your random\_RGB function here:**[sample solution](#_kwid5awmya5n) |

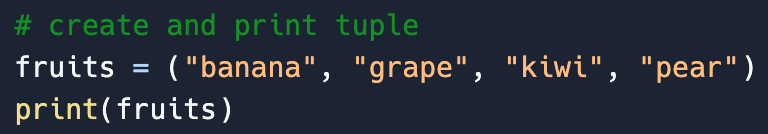
| **9.** Using both functions that you wrote to write a small program that asks the user for two coordinates, then uses turtle to write the coordinate in random colors, draw a line segment between the two points in a random color, and write the distance in a random color in some desired location (the examples below show the distance written at (-100, 100))  Here is some code you can copy/paste and use to work with turtle:   | import turtle  t = turtle.Turtle()  **t.screen.colormode(255) # needed in order to use RGB tuples for pencolor**  turtle.Screen().setup(500, 400)  t.penup()  t.goto(-50, 50)  **color = (30, 95, 150) # a tuple**  **t.pencolor(color)**  t.write("Hello!")  t.pendown() | | --- |   **Importantly**, the pencolor function for turtle can take a *string* (e.g. pencolor("red")) **or** a ***tuple*** with three numbers between 0 and 255 that represents the desired pen color in RGB; the example code above shows the pencolor function being used with a tuple instead of a string. However, you must set turtle's "colormode" to "255" in order to use tuples in this way; this line of code is included above.  **Your finished program should look something like this:** |
| --- | --- |
| **copy/paste your full code here**[sample solution](#_79bunhqmb7u5) |

**→ Done! ←**

Submit this document in Google Classroom



### Sample Solution ([back](#_xlabptciihh1))



**printed output:**



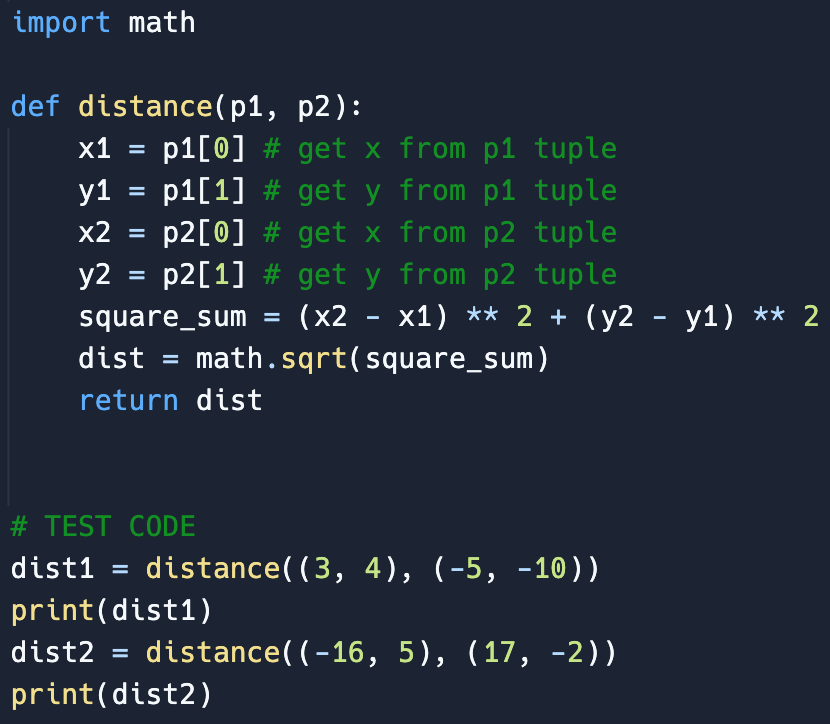
### Answers ([back](#_az0qjvq2xluh))



**Index: 0 1 2 3 \_\_\_**

| **a.** What is fruits[0]?  **b.** What is the third element of fruits?  **c.** What element is at index 3?  **d.** What happens if you try *updating* the element at index 1, like this: fruits[1] = "apple" and why?  **e.** What will happen if you try to print(fruits[4]) and why? | a. banana  b. kiwi (at index 2)  c. pear  d. **error** because you can't update a tuple's values (a tuple is **immutable**)  e. **out of bounds error** because the last index is 3 |
| --- | --- |

### Sample Solution ([back](#_pydtj8g5cegk))

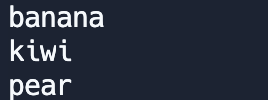


### Test code ([back](#_az0qjvq2xluh))

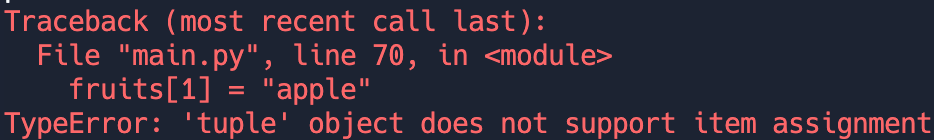


**Should print:**

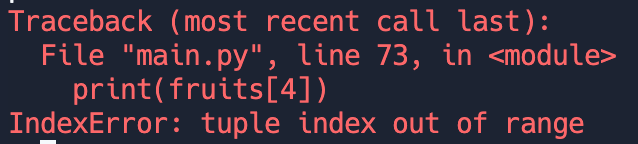
**a - c)**



**d)**  error because tuples are **immutable** which means you **can't** modify its values:

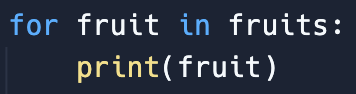


**e)** should print the following error (after commenting out test code for d above) because index 4 is **out of bounds** (just like for lists):

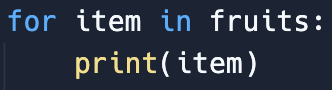
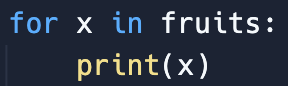


### Check ([back](#_p4sh4a6x0svb))

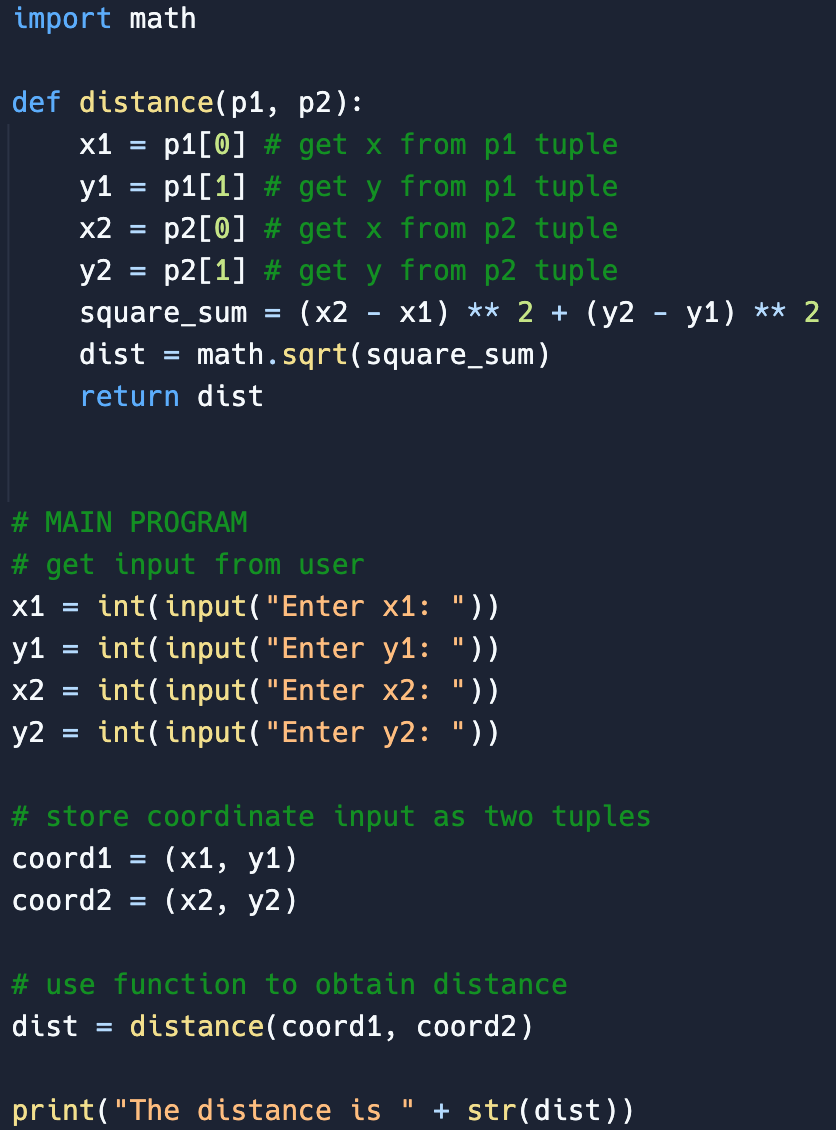
Using the singular "fruit" for the **loop variable** is typical, since the list is the plural word "fruits":



But you *can* name the loop variable whatever you want:

### Sample solution ([back](#_pqnf9wnq7ii9))



### Sample solution ([back](#_hwqfxv78yj1r))

Function + test code shown below:



### Sample solution ([back](#_m6vcdd2mfyqe))

