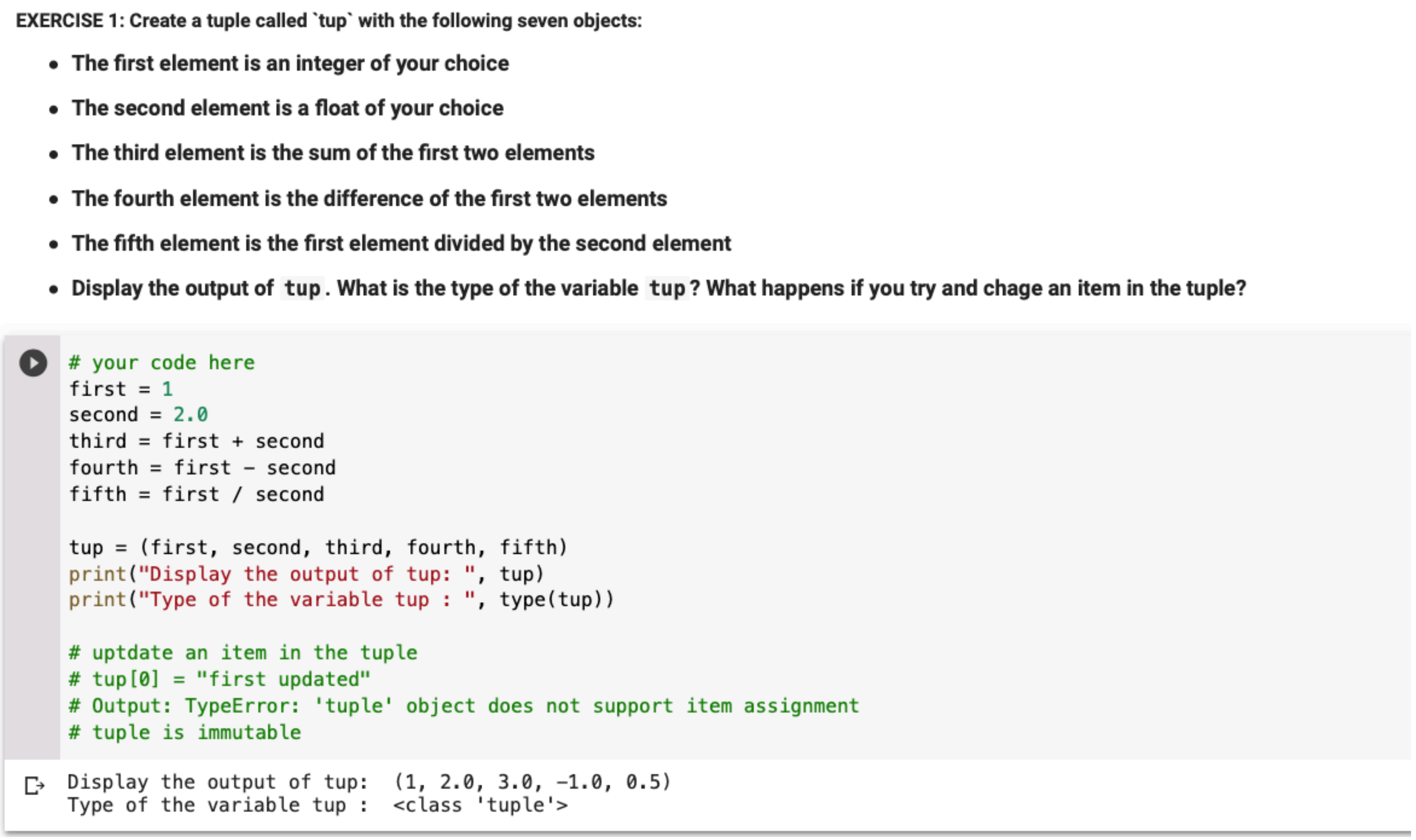

EXERCISE 1: Create a tuple called `tup` with the following seven objects:

- **The first element is an integer of your choice**
- **The second element is a float of your choice**
- **The third element is the sum of the first two elements**
- **The fourth element is the difference of the first two elements**
- **The fifth element is the first element divided by the second element**
- **Display the output of `tup`. What is the type of the variable `tup`? What happens if you try and chage an item in the tuple?**

```
# your code here
first = 1
second = 2.0
third = first + second
fourth = first - second
fifth = first / second

tup = (first, second, third, fourth, fifth)
print("Display the output of tup: ", tup)
print("Type of the variable tup : ", type(tup))

# uptdate an item in the tuple
# tup[0] = "first updated"
# Output: TypeError: 'tuple' object does not support item assignment
# tuple is immutable
```

```
Display the output of tup:  (1, 2.0, 3.0, -1.0, 0.5)
Type of the variable tup :  <class 'tuple'>
```

Exercise 2: Build a list that contains every prime number between 1 and 100, in two different ways:

- **2.1 Using for loops and conditional if statements.**
- **2.2 Using a list comprehension. You should be able to do this in one line of code. Hint: it might help to look up the function `all()` in the documentation.**

```python
### Your code here
# 2.1 Using for loops and conditional if statements.
prime_nums1 = []
for Number in range (1, 101):
    count = 0
    for i in range(2, (Number//2 + 1)):
        if(Number % i == 0):
            count = count + 1
            break

    if (count == 0 and Number != 1):
        prime_nums1.append(Number)

print(prime_nums1)


# 2.2 Using a list comprehension. You should be able to do this in one line of code. Hint: it might help to look up the function all() in the documentation.
prime_nums2 = [x for x in range(2, 101) if all(x % y != 0 for y in range(2, x))]
print(prime_nums2)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Comments

Exercise 3: generated a list of the prime numbers between 1 and 100

In Exercise 2, above, you wrote code that generated a list of the prime numbers between 1 and 100. Now, write a function called `isprime()` that takes in a positive integer N, and determines whether or not it is prime. Return `True` if it's prime and return `False` if it isn't. Then, using a list comprehension and `isprime()`, create a list `myprimes` that contains all the prime numbers less than 100.

```python
def isprime(n):
  if n == 1:
    return False
  if n == 2:
    return True

  for i in range(2, n//2+1):
      if (n % i == 0):
        return False;
  return True



myprimes = [x for x in range(1,101) if isprime(x)]
print(myprimes)







# your code here
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Exercise 4: Matrix multiplication

```
Using numpy, create a random 5X5 matrix and multiply is by the 5X5 unit matrix
```

```python
### Your code here
import numpy as np
randomMatrix = np.random.random((5,5))
unitMatrix =  np.identity(5)
productMatrix = np.matmul(randomMatrix, unitMatrix)

print("Random Matrix:")
print(randomMatrix)

print("Unit Matrix:")
print(unitMatrix)

print("Product Matrix:")
print(productMatrix)
```

```
Random Matrix:
[[0.79703239 0.53778255 0.23676865 0.40844029 0.58893061]
 [0.35387409 0.94917519 0.22580905 0.58551544 0.87155582]
 [0.13409773 0.60409796 0.17442179 0.38770237 0.0604519 ]
 [0.38270256 0.71002307 0.58291174 0.35253083 0.67311668]
 [0.53773362 0.78128022 0.10113762 0.49368499 0.53236811]]
Unit Matrix:
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
Product Matrix:
[[0.79703239 0.53778255 0.23676865 0.40844029 0.58893061]
 [0.35387409 0.94917519 0.22580905 0.58551544 0.87155582]
 [0.13409773 0.60409796 0.17442179 0.38770237 0.0604519 ]
 [0.38270256 0.71002307 0.58291174 0.35253083 0.67311668]
 [0.53773362 0.78128022 0.10113762 0.49368499 0.53236811]]
```

Exercise 5: Dictionary search

```
Given the dictionary we constructed 'my_dict', find where odd values are, print their keys, and assign to the same key the value multipl
ied by 2.
```

```python
## Your code here
my_dict = {'CS1': 500, 'CS2': 401, 'Stat1': 300, 'Stat2': 301, 'EE1': 400}
def evenOdd(dict):
  for key in dict:
    if dict[key] % 2 == 1:
      print(key)
      dict[key] *= 2
  return dict

new_dict = evenOdd(my_dict)

print("New dict: ", new_dict)
```

```
CS2
Stat2
New dict:  {'CS1': 500, 'CS2': 802, 'Stat1': 300, 'Stat2': 602, 'EE1': 400}
```