Instruction

- Write code to discover the extent of the puzzling behavior - creating and modifying pairs of variables with different types in Python.

    - Include types like int, float, string, list, tuple, and dictionary.
    - Create one variable, then create a second variable that is equal to the first.
    - Change one of the two variables and see what happens. If the type allows it try changing both the contents
      (an element in a list or dictionary) as well as setting it equal to a new object.

Tuple

In [50]:

```
# 1) Using Tuples

# 1.1) Creating a variable tuple
tuple1 = (57, "birthday", 1964, "year", "weight", 69.8,[1965, 1964, 1963])

# 1.2) Creating a second variable tuple equal to the first
tuple2 = tuple1

# 1.3) Changing the second variable
tuple2[3] = 89

# 1.4) Getting the result
print(f"The value of tuple1 is {tuple1} and the value of tuple2 is {tuple2}")
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-50-ba0e091914e9> in <module>
      8
      9 # 1.3) Changing the second variable
---> 10 tuple2[3] = 89
     11
     12 # 1.4) Getting the result

TypeError: 'tuple' object does not support item assignment
```

As we can see in the error message above, tuple does not support item assignment.

In [98]:

```
# 2) Using Tuples

# 2.1) Creating a variable tuple
tuple1 = (57, "birthday", 1964, "year", "weight", 69.8,[1965, 1964, 1963])

# 2.2) Creating a second variable tuple equal to the first
tuple2 = tuple1

# 2.3) Changing the second variable
tuple2 = [63.2, "testing"]

# 2.4) Getting the result

print(f" The value of tuple1 is {tuple1} \n The value of tuple2 is {tuple2} \n The valu
e of tuple2[1] is {tuple2[1]} ")
```

```
 The value of tuple1 is (57, 'birthday', 1964, 'year', 'weight', 69.8, [19
65, 1964, 1963])
 The value of tuple2 is [63.2, 'testing']
 The value of tuple2[1] is testing
```

As we can see above, the tuple is immutable

Dictionary

In [76]:

```
# 3) Using Dictionary

# 3.1) Creating the dictionary
Dict1 = {"key1": 1, "key2": "2", "key3": [3, 3, 3], "key4": (4, 4, 4), ('key5'): 5, (0,
1): 6}

# 3.2) Creating a second variable dictionay equal to the first
Dict2 = Dict1

# 3.3) Changing the second dictionay variable
Dict2[3] = {"notkey":[14,20,99]} # it needs curly bracket

# 3.4) Getting the result
print(f" The value of Dict1 is {Dict1} \n The value of Dict2 is {Dict2} \n The value of
Dict2[3] is {Dict2[3]}")
```

```
 The value of Dict1 is {'key1': 1, 'key2': '2', 'key3': [3, 3, 3], 'key4':
(4, 4, 4), 'key5': 5, (0, 1): 6, 3: {'notkey': [14, 20, 99]}}
 The value of Dict2 is {'key1': 1, 'key2': '2', 'key3': [3, 3, 3], 'key4':
(4, 4, 4), 'key5': 5, (0, 1): 6, 3: {'notkey': [14, 20, 99]}}
 The value of Dict2[3] is {'notkey': [14, 20, 99]}
```

As we can see above, both Dict1 and Dict2 are pointing to the same object, hence changes to the object will be perceived by any of the variables.

In [77]:

```
# 4) Using Dictionary

# 4.1) Creating the dictionary
Dict1 = {"key1": 1, "key2": "2", "key3": [3, 3, 3], "key4": (4, 4, 4), ('key5'): 5, (0,
1): 6}

# 4.2) Creating a second variable dictionay equal to the first
Dict2 = Dict1

# 4.3) Changing the second dictionay variable
Dict2= {"notkey":[14,20,99]}

# 4.4) Getting the result
print(f" The value of Dict1 is {Dict1} \n The value of Dict2 is {Dict2}")
```

```
 The value of Dict1 is {'key1': 1, 'key2': '2', 'key3': [3, 3, 3], 'key4':
(4, 4, 4), 'key5': 5, (0, 1): 6}
 The value of Dict2 is {'notkey': [14, 20, 99]}
```

As we can see above, Dict2 was pointing to the same object as Dict1, but the second assignment to Dict2 created a new reference (pointer) to another object, so from now on, Dict1 and Dict2 are referencing (pointing) to different objects.

LIST

In [86]:

```
 # 5) Using list

# 5.1) Creating the dictionary
List1 = [1, 5.2, 9, 12, 24, 33]

# 5.2) Creating a second variable List equal to the first
List2 = List1

# 5.3) Changing the second dictionay variable
List2[4]= 38, 45

# 5.4) Getting the result
print(f" The value of List1 is {List1} \n The value of List2 is {List2[4]}")
```

```
The value of List1 is [1, 5.2, 9, 12, (38, 45), 33]
The value of List2 is (38, 45)
```

In [97]:

```
 # 6) Using List

# 6.1) Creating the List
List1 = [1, 5.2, 9, 12, 24, 33]

# 6.2) Creating a second variable List equal to the first
List2 = List1

# 6.3) Changing the second List variable
List2 = 38,52

# 6.4) Getting the result
print(f" The value of List1 is {List1}  \n The value of List2 is {List2} \n The value o
f List2[1] is {List2[1]}")
```

```
 The value of List1 is [1, 5.2, 9, 12, 24, 33]
 The value of List2 is (38, 52)
 The value of List2[1] is 52
```

In [100]:

```
 # 6) Using List

# 6.1) Creating the List
List1 = [1, 5.2, 9, 12, 24, 33]

# 6.2) Creating a second variable List equal to the first
List2 = List1

# 6.3) Changing the second List variable
List2 = 38,52

# 6.4) Getting the result
print(f" The value of List1 is {List1}  \n The value of List2 is {List2} \n The value o
f List2[1] is {List2[1]}")
```

```
 The value of List1 is [1, 5.2, 9, 12, 24, 33]
 The value of List2 is (38, 52)
 The value of List2[1] is 52
```

In [90]:

```
# 7) Using List

# 7.1) Creating the List
List1 = ["a", "b", "c", "d", "f"]

# 7.2) Creating a second variable List equal to the first
List2 = List1

# 7.3) Changing the second List variable
List2[3] = "k","A"

# 7.4) Getting the result
print(f" The value of List1 is {List1} \n The value of List2 is {List2} \n The value of
List2[3] is {List2[3]}" )
```

```
 The value of List1 is ['a', 'b', 'c', ('k', 'A'), 'f']
 The value of List2 is ['a', 'b', 'c', ('k', 'A'), 'f']
 The value of List2[3] is ('k', 'A')
```