Exercise

Attempt the following exercises. When you see an output, **DO NOT WRITE IN THAT CELL**, A cell will be provided **below the output cell** where you can run your code.

IF YOU RESTART AND RUN ALL CELLS, THE OUTPUT WILL DISAPPEAR

Question 1

Create a list that contains **10 elements (3 integers, 2 strings, 2 boolean, 2 lists and 1 dictionary)** and attribute it to a variable called **new_var**

```
[3,4,7, 'string one','string 2', true, false, [4,5,6], [5,8,4], {'ten':taiwo}]
```

In [115]:
```
new_var = [1,2,3, 'string 1','string 2', True, False, [2,4], [2,6,4], {'number5':'taiwo'}]
```

Question 2

Concatenate the following variables:

x = 'This is a string'

y = 'that has been concatenated'

In [4]:
```
x='This is a string'
y='that has been concatenated'
x+' '+y
```

Out[4]:
```
'This is a string that has been concatenated'
```

In [ ]:
```
```

```
In [14]: # use the list below to answer question 3, 4, 5
         list_one = [1,2,3,4, ['list', 'in', 'list', {'k1': 9, 'k2':8,
                                           'k3':[4,5,(6,90,{'name':'Abigail', 'age':60}),7]}], 12, 48, 90]
```

Question 3

Using negative indexing alone, index out Abigail from the complex list

```
In [13]: list_one = [1,2,3,4, ['list', 'in', 'list', {'k1': 9, 'k2':8,
                                           'k3':[4,5,(6,90,{'name':'Abigail', 'age':60}),7]}], 12, 48, 90]
```

```
In [104]: list_one[-4][-1]['k3'][-2][-1]['name']

Out[104]: 'Abigail'
```

Question 4

Reassign the value age of the key 'age' to 65

```
In [48]: list_one [-4][-1]['k3'][-2][-1]['age']='65'
         list_one [-4][-1]['k3'][-2][-1]['age']

Out[48]: '65'
```

Question 5

Using slicing, index out the last 3 items within the list one

```
In [58]: list_one[5:]

Out[58]: [12, 48, 90]
```

Question 6

Using the appropriate dictionary method, print out the items of the dictionary in list_one

In [85]:
```
list_one[4][-1].items()
```

Out[85]:
```
dict_items([('k1', 9), ('k2', 8), ('k3', [4, 5, (6, 90, {'name': 'babatunde', 'age': '65'}), 7])])
```

Question 7

Merge the two lists created in the above questions to become one list.

In [126]:
```
print (new_var)
print (list_one)
list_one.extend (new_var)
```

```
[1, 2, 3, 'string 1', 'string 2', True, False, [2, 4], [2, 6, 4], {'number5': 'taiwo'}]
[1, 2, 3, 4, ['list', 'in', 'list', {'k1': 9, 'k2': 8, 'k3': [4, 5, (6, 90, {'name': 'Abigail', 'age': '65'}), 7]}], 1
2, 48, 90, 1, 2, 3, 'string 1', 'string 2', True, False, [2, 4], [2, 6, 4], {'number5': 'taiwo'}, 1, 2, 3, 'string 1',
'string 2', True, False, [2, 4], [2, 6, 4], {'number5': 'taiwo'}]
```

Question 8

Using a **for loop, print the data type of each element** in the list you **created in Question 1**

In [140]:
```
print(new_var)
for x in new_var:
    print (type(x))
```

```
[1, 2, 3, 'string 1', 'string 2', True, False, [2, 4], [2, 6, 4], {'number5': 'taiwo'}]
<class 'int'>
<class 'int'>
<class 'int'>
<class 'str'>
<class 'str'>
<class 'bool'>
<class 'bool'>
<class 'list'>
<class 'list'>
<class 'dict'>
```

Question 9

Create an function called **deduct** that takes two arguments (Salary and minutes). It should deduct #500, #700, #1000 from a person's salary if the person arrives work **above 30, above 45 and above 60 minutes late** respectively. The function should also always print *The number of minutes the person was late* and *Shun late coming everytime it reduces your salary*

In [3]:
```
deduct(9000, 69) # the output should look like this
```

```
You were late for 69 minutes
New salary:  8000
Shun late coming everytime it reduces your salary
```

In [164]:
```
def deduct (salary, minutes):
    print ('You were late for') + str(minutes + 'minutes')
    if minutes >30 and minutes <= 45:
        print ('new salary:'+ str(salary-500))
    elif minutes >45 and minutes <= 60:
        print ('new salary:'+ str(salary-700))
    elif minutes >60:
        print ('new salary:'+ str(salary-1000))
    print ('shun late coming because its reduces your salary everytime')
```

In [165]:
```
deduct(9000, 69)
```

```
You were late for

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-165-d49794a9090d> in <module>
----> 1 deduct(9000, 69)

<ipython-input-164-3e3be67df1a6> in deduct(salary, minutes)
      1 def deduct (salary, minutes):
----> 2     print ('You were late for') + str(minutes + 'minutes')
      3     if minutes >30 and minutes <= 45:
      4         print ('new salary:'+ str(salary-500))
      5     elif minutes >45 and minutes <= 60:

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Question 10

Create a function called **factor_check** that takes in two arguments. The first argument is a number while the second is a factor of the first argument. The function should return True if the number is a factor and False if the number is not a factor.

In [4]:
```
factor_check(90, 9) # Example
```

Out[4]:
```
True
```

In [174]:
```
def factor_check (number,x):
    if number%x==0:
        return (True)
    else:
        print (False)
```

In [176]:
```
def factor_check (firstnumber,secondnumber):
    if firstnumber&secondnumber==0:
        return (True)
    else:
        print (False)
```

Good luck !!!

In [ ]:
```
def
```