1 GBA 465 Lab 04 - Skill-Building with Advanced Collections (Voltron) (Starter)

In [1]:
```
# your implementation:

print ("Hello, World!")
```

```
Hello, World!
```

1.1 Part A

Action: Create a list called `lionColorsList` that contains the colors for The Lions (Blue, Yellow, Green, Red, Black).

In [2]:
```
# your implementation:

lionColorList = ['Blue','Yellow','Green','Red','Black']
```

Action: Print the value in `lionColorsList` to the screen.

In [3]:
```
# your implementation:
# Pinrt lionColorList
print(lionColorList)
```

```
['Blue', 'Yellow', 'Green', 'Red', 'Black']
```

Action: Convert `lionColorsList` into tuple and assign the value to a new variable called `lionColorsTuple`.

In [4]:
```
# your implementation:
# Convert lionColorsList into tuple and assign the value to a new variable calle
lionColorsTuple = tuple(lionColorList)
```

Action: Print the value in `lionColorsTuple` to the screen.

In [5]:
```
# your implementation:
# Print the value in lionColorsTuple
print(lionColorsTuple)
```

```
('Blue', 'Yellow', 'Green', 'Red', 'Black')
```

Action: Iterate over the items in `lionColorsTuple` in order to print each color on a separate line.

In [6]:
```
# your implementation:
# Using for loop to iterate over the items in lionColorsTuple and print
for i in lionColorsTuple:
    print(i)
```

```
Blue
Yellow
Green
Red
Black
```

1.2 Part B

Near the beginning of the show, Shiro becomes Paladin of the Black Lion and wears the matching Black uniform color. This data is captured in the variables in the following cell:

In [7]:
```
# Black Paladin Data
name = "Shiro"
isPaladin = True
uniformColor = lionColorsTuple [4]
lionColor = lionColorsTuple [4]
```

Action: Create a dictionary of key-value pairs based on the variables above and store in a variable called `blackPaladinData`.

- For the key of each key-value pair, use the variable's name.
- For the value of each key-value pair, reference the variable's value.

In [8]:
```
# Your implementation:
# Create a dictionary of key-value pairs based on the variables above and store i
blackPaladinData = {
    'name':"Shiro",
    'isPaladin': True,
    'uniformColor':lionColorsTuple[4],
    'lionColor':lionColorsTuple[4]}
```

Action: Print out the entire dictionary in `blackPaladinData`.

In [9]:
```
# Your implementation:
# Print out the entire dictionary
print(blackPaladinData)
```

```
{'name': 'Shiro', 'isPaladin': True, 'uniformColor': 'Black', 'lionColor': 'Bla
ck'}
```

In Jupyter Notebook, you don't always need a print function for quick data dump; you can just write an expression on the last line of a code cell.

Action: Print out the entire dictionary in `blackPaladinData` without using a print function.

In [10]:
```
# Your implementation:
# Print out the entire dictionary in blackPaladinData without using a print funct
blackPaladinData
```

Out[10]:
```
{'name': 'Shiro',
 'isPaladin': True,
 'uniformColor': 'Black',
 'lionColor': 'Black'}
```

To keep track of Shiro's age, we need to add a new key-value pair to the `blackPaladinData` dictionary.

Shiro is 25 years old.

Actions:

- Add a key named `age`.
- Add an integer value `25`.
- Print out the entire dictionary in `blackPaladinData` to verify if Shiro's age was added successfully.

In [11]:
```
# Your implementation:
# Add a key named age
# Add an integer value 25
# Print
blackPaladinData['age'] = 25
blackPaladinData
```

Out[11]:
```
{'name': 'Shiro',
 'isPaladin': True,
 'uniformColor': 'Black',
 'lionColor': 'Black',
 'age': 25}
```

Later in the show, Shiro no longer pilots the Black Lion, but continues to wear the Black uniform color.

Actions:

- Change the value associated with the `isPaladin` key to `False`.
- Change the value associated with the `lionColor` key to `None`.
- Print out the entire dictionary in `blackPaladinData` to verify these changes were made successfully.

In [12]:
```
# Your implementation:
# Change the value associated with the isPaladin key to False
# Change the value associated with the lionColor key to None
blackPaladinData['isPaladin'] = False
blackPaladinData['lionColor'] = None
blackPaladinData
```

Out[12]:
```
{'name': 'Shiro',
 'isPaladin': False,
 'uniformColor': 'Black',
 'lionColor': None,
 'age': 25}
```

Actions:

- Write a conditional statement to check if the `blackPaladinData` dictionary contains a key called `isPaladin`.
- If it does, print `The key 'isPaladin' is in the dictionary.`.
- If it does not, print `The key 'isPaladin' is not in the dictionary.`.

In [13]:
```
# Your implementation:
# Writing a if statement to check
if isPaladin in blackPaladinData:
    print("The key 'isPaladin' is in the dictionary.")
else:
    print("The key 'isPaladin' is not in the dictionary.")
```

```
The key 'isPaladin' is not in the dictionary.
```

A Bayard is the traditional weapon of a Voltron Paladin.

We don't currently have any data on Shiro's Bayard stored in `blackPaladinData`.

Actions:

- Use the dictionary's `get` method to attempt to access the non-existent key `bayard` in `blackPaladinData`.
- Print this value to the screen.

In [14]:
```
# Your implementation:
# Use the dictionary's get method to attempt to access the non-existent key bayar

print(blackPaladinData.get("bayard"))
```

```
None
```

Actions:

- Use the dictionary's `get` method (again) to attempt to access the non-existent key `bayard` in `blackPaladinData`.
- This time, specify a value of `Sword` as a default value for `bayard`.
- Print this value to the screen.

In [15]:
```
# Your implementation:
# blackPaladinData['bayard'] = 'Sword'
print(blackPaladinData.get('bayard',"Sword"))
```

```
Sword
```

When using a default value and the method `get`, does the `bayard`:`Sword` key-value pair get added to `blackPaladinData`?

Actions:

- Write a conditional statement to check if the `blackPaladinData` dictionary contains a key called `bayard`.
- If it does, print `The key 'bayard' is in the dictionary.`.
- If it does not, print `The key 'bayard' is not in the dictionary.`.

In [16]:
```
# Your implementation:
# using if conditional statment to check if the blackPaladinData dictionary conta
# and print matching content

key = 'bayard'
if key in blackPaladinData:
    print("The key" + key + "is in the dictionary.")
else:
    print("The key" + key + "is not in the dictionary.")
blackPaladinData
```

```
The keybayardis not in the dictionary.
```

Out[16]:
```
{'name': 'Shiro',
 'isPaladin': False,
 'uniformColor': 'Black',
 'lionColor': None,
 'age': 25}
```

Actions:

- Remove the key-value pair for `uniformColor` from `blackPaladinData`.
- Print out the entire dictionary in `blackPaladinData` to verify these changes were made successfully without using the print function.

In [17]:
```
# Your implementation:
# using del to remove uniformColor from blackPaladinData
blackPaladinData['uniformColor'] = lionColorsTuple[4]
del blackPaladinData['uniformColor']
blackPaladinData
```

Out[17]:
```
{'name': 'Shiro', 'isPaladin': False, 'lionColor': None, 'age': 25}
```

1.3 Part C

Action: Execute the following code:

In [18]:
```
# create data dictionary for Lance

bluePaladinData = {
    "name": "Lance",
    "age": 17,
    "isPaladin": True,
    "uniformColor": "Blue",
    "lionColor": "Blue"
}
```

Action: Using the `keys` method, iterate over the `bluePaladinData` dictionary's **keys** to print each **key** on a separate line.

In [19]:
```
# Your implementation:
# Using the keys method, iterate over the bluePaladinData dictionary's keys to pr
for key in bluePaladinData.keys():
    print(key)
```

```
name
age
isPaladin
uniformColor
lionColor
```

Action: Using the `values` method, iterate over the `bluePaladinData` dictionary's **values** to print each **value** on a separate line.

In [20]:
```
# Your implementation:
# Using the values method, iterate over the bluePaladinData dictionary's values t
for key in bluePaladinData.values():
    print(key)
```

```
Lance
17
True
Blue
Blue
```

Action: Using the `keys` method, iterate over the `bluePaladinData` dictionary's **keys** and print each **value** out on a separate line.

In [21]:
```
# Your implementation:
# Using the keys method, iterate over the bluePaladinData dictionary's keys
for key in bluePaladinData.keys():
    print(bluePaladinData[key])
```

```
Lance
17
True
Blue
Blue
```

Action: Using the `items` method, iterate over the `bluePaladinData` dictionary's **keys** and **values**, and print each **key** and **value** out on a separate line.

In [22]:
```
# Your implementation:
# Using the items method, iterate over the bluePaladinData dictionary's keys and
for key in bluePaladinData.items():
    print(key)
```

```
('name', 'Lance')
('age', 17)
('isPaladin', True)
('uniformColor', 'Blue')
('lionColor', 'Blue')
```

1.4 Part D

Now you are going to make a dictionary representing a larger set of structured data.

It will be a "dictionary of dictionaries".

Like any dictionary, the top-level dictionary will contain key-values pairs. The key will be a lion color. The value will be a dictionary containing additional key-value pairs for the Paladin who pilots that lion.

Action: Create an empty dictionary called `lionsData`.

In [23]:
```
# Your implementation:
# Create an empty dictionary called lionsData
lionsData = {}
```

Actions:

- Iterate over `lionColorsTuple` to create a key-value pair in `lionsData` for each color.
- The key will be the color.
- The value will be an empty dictionary.
- After you are done iterating, print `lionsData` to the screen.

In [24]:
```
# Your implementation:
# Iterate over lionColorsTuple to create a key-value pair in lionsData
for lionColor in lionColorsTuple:
     lionsData[lionColor] = {}

lionsData
```

Out[24]:
```
{'Blue': {}, 'Yellow': {}, 'Green': {}, 'Red': {}, 'Black': {}}
```

Actions:

- Add the following key-value pair to the `Blue` key in `lionsData`.
  - key: `name`
  - name: `Lance`
- Print `lionsData` to the screen.

In [25]:
```
# Your implementation:
# Add the following key-value pair to the Blue key in lionsData and print
lionsData["Blue"]["name"] = "Lance"
print(lionsData)
```

```
{'Blue': {'name': 'Lance'}, 'Yellow': {}, 'Green': {}, 'Red': {}, 'Black': {}}
```

Actions:

- Add the following key-value pairs to the appropriate key in `lionsData`.

- For `Yellow` : key is `name` , value is `Hunk`
- For `Green` : key is `name` , value is `Pidge`
- For `Red` : key is `name` , value is `Keith`
- For `Black` : key is `name` , value is `Shiro`

- Print `lionsData` to the screen.

In [26]:
```
# Your implementation:
# Add the following key-value pairs to the appropriate key in lionsData
# For Yellow: key is name, value is Hunk
# For Green: key is name, value is Pidge
# For Red: key is name, value is Keith
# For Black: key is name, value is Shiro
lionsData["Yellow"]["name"] = "Hunk"
lionsData["Green"]["name"] = "Pidge"
lionsData["Red"]["name"] = "Keith"
lionsData["Black"]["name"] = "Shiro"

lionsData
```

Out[26]:
```
{'Blue': {'name': 'Lance'},
 'Yellow': {'name': 'Hunk'},
 'Green': {'name': 'Pidge'},
 'Red': {'name': 'Keith'},
 'Black': {'name': 'Shiro'}}
```

Let's add the uniform colors.

Actions:

- Add the following key-value pairs to the appropriate key in `lionsData` .
  - For `Blue` : key is `uniformColor` , value is `Blue`
  - For `Yellow` : key is `uniformColor` , value is `Yelow`
  - For `Green` : key is `uniformColor` , value is `Green`
  - For `Red` : key is `uniformColor` , value is `Red`
  - For `Black` : key is `uniformColor` , value is `Black`
- Print `lionsData` to the screen.

In [27]:
```
# Your implementation:
# Add the following key-value pairs to the appropriate key in lionsData.
# For Blue: key is uniformColor, value is Blue
# For Yellow: key is uniformColor, value is Yelow
# For Green: key is uniformColor, value is Green
# For Red: key is uniformColor, value is Red
# For Black: key is uniformColor, value is Black
lionsData["Blue"]["uniformColor"] = "Blue"
lionsData["Yellow"]["uniformColor"] = "Yellow"
lionsData["Green"]["uniformColor"] = "Green"
lionsData["Red"]["uniformColor"] = "Red"
lionsData["Black"]["uniformColor"] = "Black"

lionsData
```

Out[27]:
```
{'Blue': {'name': 'Lance', 'uniformColor': 'Blue'},
 'Yellow': {'name': 'Hunk', 'uniformColor': 'Yellow'},
 'Green': {'name': 'Pidge', 'uniformColor': 'Green'},
 'Red': {'name': 'Keith', 'uniformColor': 'Red'},
 'Black': {'name': 'Shiro', 'uniformColor': 'Black'}}
```

During the show, some Paladins pilot different lions, while continuing to wear their original uniform color.

- Keith pilots the Black Lion, but continues to wear the red uniform color.
- Lance pilots the Red Lion, but continues to wear the blue uniform color.

Actions:

- Change the values for the Black Lion so that it is now piloted by Keith wearing a Red uniform.
- Change the values for the Red Lion so that it is now piloted by Lance wearing a Blue uniform.
- Print `lionsData` to the screen.

In [33]:
```
# Your implementation:
# Change the values for the Black Lion so that it is now piloted by Keith wearing
lionsData['Black']['name'] = 'Keith'
lionsData['Black']['uniformColor'] = 'Red'
# Change the values for the Red Lion so that it is now piloted by Lance wearing a
lionsData['Red']['name'] = 'Lance'
lionsData['Red']['uniformColor'] = 'Blue'
```

During the show, some characters start being Paladins, wearing a new uniform color that does not match any lion.

Now that Lance is piloting the Red Lion, we need Allura to pilot the Blue Lion. Her uniform color is `Pink` .

Actions:

- Modify `lionsData` so that Allura pilots the Blue Lion while wearing a Pink uniform.
- Print `lionsData` to the screen.

In [34]:
```
# Your implementation:
# Modify lionsData so that Allura pilots the Blue Lion while wearing a Pink unifo
lionsData['Blue']['name'] = 'Allura'
lionsData['Blue']['uniformColor'] = 'Pink'

lionsData
```

Out[34]:
```
{'Blue': {'name': 'Allura', 'uniformColor': 'Pink'},
 'Yellow': {'name': 'Hunk', 'uniformColor': 'Yellow'},
 'Green': {'name': 'Pidge', 'uniformColor': 'Green'},
 'Red': {'name': 'Lance', 'uniformColor': 'Blue'},
 'Black': {'name': 'Keith', 'uniformColor': 'Red'}}
```

1.5 Part E

Action: Iterate over `lionsData` to achieve the following output dynamically. The order of the Lions does not matter.

```
Blue Lion: Allura (Pink Uniform)

Yellow Lion: Hunk (Yellow Uniform)

Green Lion: Pidge (Green Uniform)

Red Lion: Lance (Blue Uniform)

Black Lion: Keith (Red Uniform)
```

In [35]:
```
# Your implementation:
# Iterate over lionsData to achieve the following output dynamically.
for i in lionsData.keys():
    print(i + " Lion: " + lionsData[i]['name'] + " (" + lionsData[i]['uniformCol
```

```
Blue Lion: Allura (Pink Uniform)
Yellow Lion: Hunk (Yellow Uniform)
Green Lion: Pidge (Green Uniform)
Red Lion: Lance (Blue Uniform)
Black Lion: Keith (Red Uniform)
```

Action: Lastly, use chain bracket syntax to print the value verifying the uniform color for the Blue Lion.

In [36]:
```
# Your implementation:
# use chain bracket syntax to print the value verifying the uniform color for the
lionsData['Blue']['uniformColor']
```

Out[36]:
```
'Pink'
```