Assignment - Dictionaries in Pokemon

Each question is worth 5 points

We will use a pockemon dataset to answer questions in this assignment. The dataset includes details of 800 imaginary pets, represented as a list of dictionaries.

Reference: calmcode.io

This code block fetches a JSON file from a website.

In [15]:
```
import requests
r = requests.get('https://calmcode.io/datasets/pokemon.json')
pokemon = r.json()

len(pokemon)
type(pokemon)
```

Out[15]:
```
list
```

This is the first item in the list - first imaginary pet. A Pokemon's dictionary include the following features:

HP (Hit Points) and Attack are stats in the Pokemon video game series that determine a Pokemon's health and strength in battle, respectively.

HP represents the amount of damage a Pokemon can take before fainting. A higher HP value means the Pokemon is more durable and can withstand more attacks.

Attack represents the strength of a Pokemon's physical or special attacks. A higher Attack value means the Pokemon is capable of dealing more damage with its attacks.

Total typically refers to the sum of a Pokemon's base stats, which includes its HP, Attack, and some other stats.

Type refers to a classification that determines the strengths and weaknesses of a Pokemon in battle. There are different types in the Pokemon universe, such as Normal, Fire, Water, Grass, Electric, and more.

In [16]:
```
pokemon[0]
```

Out[16]:
```
{'attack': 49,
 'hp': 45,
 'name': 'Bulbasaur',
 'total': 318,
 'type': ['Grass', 'Poison']}
```

In [17]:
```
first_dict = pokemon[0]
first_dict['name']
```

Out[17]:
```
'Bulbasaur'
```

Problem 0

This is an example of a Python function that returns a list of the names of the first ten pets in the Pokemon dataset.

In [4]:
```
def first_ten_names():
    list_of_names = []
    for pet in pokemon[0:10]:
        list_of_names.append(pet['name'])
    return list_of_names
```

In [5]:
```
first_ten_names()
```

Out[5]:
```
['Bulbasaur',
 'Ivysaur',
 'Venusaur',
 'VenusaurMega Venusaur',
 'Charmander',
 'Charmeleon',
 'Charizard',
 'CharizardMega Charizard X',
 'CharizardMega Charizard Y',
 'Squirtle']
```

Problem 1

Write a function that accepts a pet name as a parameter and returns a list, type(s) of that pet.

In [41]:
```
def get_type_by_name(name):
    for i in pokemon:
        if i['name'] == name:
            print(i['type'])
            return(i['type'])
    return[]
```

In [42]:
```
# DO NOT CHANGE THIS CELL
assert(get_type_by_name('Squirtle') == ['Water'])
assert(get_type_by_name('Uknown Pet') == [])
assert(get_type_by_name('Ivysaur') == ['Grass', 'Poison'])
```

```
['Water']
['Grass', 'Poison']
```

Problem 2

A. When you answered the previous question, you observed that some pets have multiple types. Can you calculate the number of pets with multiple types? Write a function that calculates the number of pets with multiple types and returns that count.

In [47]:
```
# your code here
def pets_with_multiple_types():
    count = 0
    for i in pokemon:
        if len(i['type']) > 1:
                count += 1

    return count
```

In [48]:
```
# test your function here - it should return an integer.
pets_with_multiple_types()
```

Out[48]:
```
414
```

Problem 3

Write a function that retrieves a list of pet names based on a specified range of Hit Points. The function takes minimum and maximum Hit Points as input and returns a list of Pokemon names.

In [73]:
```
def hit_points_range(min,max):
    names = []
    for i in pokemon:
        if i["hp"] >= min and i["hp"] <= max:
            names.append(i['name'])
    print(names)
    return names
```

In [74]:
```
# test your function here - DO NOT CHANGE THE CODE
assert (hit_points_range(0,10) == ['Diglett', 'Shedinja'])
```

```
['Diglett', 'Shedinja']
```

Problem 4

Write a function that returns the number of pets that belong to a specified type.

In [4]:
```
def match_type(typ):
    count = 0
    for i in pokemon:
        if isinstance(i['type'], list):
            types = i['type']
        for t in types:
            t = t.strip()
            if t == typ:
                count += 1
    print(count)
    return count
```

In [5]:
```
# test your function here - DO NOT CHANGE THE CODE
assert(match_type('Fire') == 64)
assert (match_type('Dragon') == 50)
```

```
64
50
```

Problem 5

Write a function that returns a list of pet names that belong to two specified types.

In [6]:
```
def match_types(typ1, typ2):
    names = []
    for i in pokemon:
        if isinstance(i['type'], list):
            types = i['type']
        if typ1 in types and typ2 in types:
            names.append(i['name'])
    print(names)
    return names
```

In [7]:
```
# test your function here - DO NOT CHANGE THE CODE
assert(match_types('Fire', 'Dragon') ==['CharizardMega Charizard X', 'Reshiram'])
assert(match_types('Rock', 'Fairy') == ['Carbink', 'Diancie', 'DiancieMega Diancie'])
```

```
['CharizardMega Charizard X', 'Reshiram']
['Carbink', 'Diancie', 'DiancieMega Diancie']
```

Problem 6

Write a function that takes a Pokemon type as input and returns the average Hit Points for that type.

In [19]:
```
def average_hp_type(typ):
    total_hp = 0
    count = 0
    for i in pokemon:
        if isinstance(i['type'], list):
            if typ in i['type']:
                total_hp += i['hp']
                count += 1
        elif i['type'] == typ:
            total_hp += i['hp']
            count += 1

    print(total_hp / count if count > 0 else 0)
    return total_hp / count if count > 0 else 0
```

In [20]:
```
# DO NOT CHANGE THIS BLOCK
assert (average_hp_type('Fire') == 70.15625)
assert(average_hp_type('Dragon') == 82.9)
assert(average_hp_type('No Type') == 0)
```

```
70.15625
82.9
0
```

Question 7 (Extra Bonus)

3 point

Create a dictionary that maps a Pokemon type to the count of pets of that type

In [18]:
```
pet_types = {}
def match_types(typ1):
    names = []
    for i in pokemon:
        if isinstance(i['type'], list):
            types = i['type']
        if typ1 in types:
            names.append(i['name'])
    for name in names:
        if typ1 in types:
            if typ1 in pet_types:
                pet_types[typ1] += 1
            else:
                pet_types[typ1] = 1
    print(pet_types)
    return pet_types
```

All Done! Submit your work.

In [19]:
```
assert (match_types('Fire') == 64)
```

```
{'Fire': 64}

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-19-464ea5679ba0> in <module>()
----> 1 assert (match_types('Fire') == 64)

AssertionError:
```

In [ ]:
```

```