Introducing Lists (Chapter 3)

In [2]:
```
# a list is a collection of items in a particular order.
# lists can contain any type of varible (letters, numbers, strings)
favorite_numbers = [1, 42, 18]
print(favorite_numbers)

favorite_foods = ['buffalo chicken sandwich','a second buffalo chicken sandwich']
print(favorite_foods)

# lists can even contain other lists
all_my_favorites = [favorite_numbers, favorite_foods]

# different types of items can be in the same list
random_stuff = [-5, 'Go Navy', 'A']

# TODO: make a list of all of your classes (like ['EW200', ...]) and print it
MyClasses = ['EW200', 'EM211', 'HH215', 'NN220', 'PE201', 'SM316', 'SP211']
print(MyClasses)
```

```
[1, 42, 18]
['buffalo chicken sandwich', 'a second buffalo chicken sandwich']
['EW200', 'EM211', 'HH215', 'NN220', 'PE201', 'SM316', 'SP211']
```

Accessing Elements in a List

In [4]:
```
# access items by their position in the list or 'index'
names = ['Alice', 'Bob', 'Charlie']
print(names[0]) # NOTE the first item has index 0

#Array elements act just like normal variables from Chapter 2
print(names[0].lower())
print(f"Hello my name is {names[1]}")

#TODO print the last name in the list
print(names[2])

#TODO what does names[-1] print?
print(names[-1])
```

```
Alice
alice
Hello my name is Bob
Charlie
Charlie
```

Changing, Adding, and Removing Elements

In [6]:
```
# Lists are dynamic. You can add, remove, and change elements
names = ['Alice','Bob', 'Charlie']
names[1] = 'Bart'
print(names)

#appending to the end of the list
names.append('Dan')
print(names)

# inserting elements into a list
names.insert(1, 'Beth')
print(names)

# combining lists
other_names = ['Xavier','Yolanda','Ziggy']
all_names = names+other_names
print(all_names)

# Removing Elements from a list
del names[0]
print(names)

best_friend = names.pop() # pop last item into a variable
print(best_friend)

other_best_friend = names.pop(1) # pop a particular item into a variable
print(other_best_friend)
# removing item by value
names.remove('Charlie')
print(names)

# CAREFUL: remove only removes the first occurence in the list
colors = ['red','green','blue','green']
colors.remove('green')
print(colors) # still has green
```

```
['Alice', 'Bart', 'Charlie']
['Alice', 'Bart', 'Charlie', 'Dan']
['Alice', 'Beth', 'Bart', 'Charlie', 'Dan']
['Alice', 'Beth', 'Bart', 'Charlie', 'Dan', 'Xavier', 'Yolanda', 'Ziggy']
['Beth', 'Bart', 'Charlie', 'Dan']
Dan
Bart
['Beth']
['red', 'blue', 'green']
```

In [9]:
```
# TODO create a list of the vowels in a random order but skip 'i'
vowels = ['a','o','u','e']
print(vowels)

#      insert i at the end of the list
vowels.append('i')
print(vowels)

#      pop your favorite vowel off into a new variable
favorite = vowels.pop(3)
print(favorite)
```

```
['a', 'o', 'u', 'e']
['a', 'o', 'u', 'e', 'i']
e
```

Organizing a List

In [10]:
```python
#there are several functions that can sort lists in particular order
names = ['Bob','Charlie','Alice']
names.sort() # in place (permanently sorts)
print(names)
names.sort(reverse=True)
print(names)

# create a new copy of the list (original is unchanged)
names = ['Bob','Charlie','Alice']
sorted_names = sorted(names)
print("Original:")
print(names)
print("Sorted:")
print(sorted_names)

# change order (not a sort)
print(names)
names.reverse()
print(names)
```

```
['Alice', 'Bob', 'Charlie']
['Charlie', 'Bob', 'Alice']
Original:
['Bob', 'Charlie', 'Alice']
Sorted:
['Alice', 'Bob', 'Charlie']
['Bob', 'Charlie', 'Alice']
['Alice', 'Charlie', 'Bob']
```

In [11]:
```python
# TODO create a list of the vowels in a random order
randVowels = ['i', 'o','e','a','u']
print(randVowels)
#      sort the list alphabetically
randVowels.sort()
print(randVowels)
#      reverse the list
randVowels.reverse()
print(randVowels)
```

```
['i', 'o', 'e', 'a', 'u']
['a', 'e', 'i', 'o', 'u']
['u', 'o', 'i', 'e', 'a']
```

Helpful Hints

In [12]:
```python
# negative indexes count from the back of the list
# -1 is the last item, -2 is the second-to-last, etc
print(names[-2])

# len returns the length of a list (number of elements)
print(f"There are {len(names)} names")
```

```
# index errors are common, especially off-by-one
names = ['Bob','Charlie','Alice']
print(names[3]) # three names but last element is index 2
```

```
Charlie
There are 3 names
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[12], line 10
      8 # index errors are common, especially off-by-one
      9 names = ['Bob','Charlie','Alice']
---> 10 print(names[3])

IndexError: list index out of range
```

Homework Problems

3-1. Names: Store the names of a few of your friends in a list called `names`. Print each person's name by acceessing each element in the list, one at a time

In [ ]:
```
names = ['Avery','Leah','Molly']
print(names[0])
print(names[1])
print(names[3])
```

3-2. Greetings: Start with the list you used above, but instead of just printing each person's name, print a message to them. The text of each message should be the same, but each message should be personalized with the person's name

In [15]:
```
names = ['Avery','Leah','Molly']
print(f"Hey, {names[0]} how are you?")
print(f"Hey, {names[1]} how are you?")
print(f"Hey, {names[2]} how are you?")
```

```
Hey, Avery how are you?
Hey, Leah how are you?
Hey, Molly how are you?
```

3-3. Your Own List: Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples. Use your list to print a series of statements about these items, such as "I would like to own a Honda motorcycle"

In [16]:
```
transpo = ['trolly', 'ostrich', 'pogo stick']
print(f'I like to ride in a {transpo[0]}')
print(f'I get places quickly when I ride an {transpo[1]}')
print(f'I am tired of taking my {transpo[2]} to work')
```

```
I like to ride in a trolly
I get places quickly when I ride an ostrich
I am tired of taking my pogo stick to work
```

3-4. Guest List: If you could invite any famous Naval Officer, living or deceased, to a dinner, who would you invite? Make a list that includes at least three officers you'd like to invite to dinner. Then use your list to print a message to each person, inviting them to dinner.

In [17]:
```
officers = ['Jimmy Carter','John McCain', 'Wendy B. Lawrence']
print(f'It would be an honor to eat dinner with {officers[0]}')
print(f'It would be an honor to eat dinner with {officers[1]}')
print(f'It would be an honor to eat dinner with {officers[2]}')
```

```
It would be an honor to eat dinner with Jimmy Carter
It would be an honor to eat dinner with John McCain
It would be an honor to eat dinner with Wendy B. Lawrence
```

3-5. Changing Guest List: One of your guests cannot make it, which is ok because you forgot to invite someone from the Marines.

- Start with the code from above. Add a `print()` call at the end of the program stating the name of the guest who can't make it.
- Modify the list, replacing the guest who can't make it with a famous Marine Officer.
- Print a second set of invitation messages, one for each person who is still in your list

In [20]:
```
officers = ['Jimmy Carter','John McCain', 'Wendy B. Lawrence']
print(f'It would be an honor to eat dinner with {officers[0]}')
print(f'It would be an honor to eat dinner with {officers[1]}')
print(f'It would be an honor to eat dinner with {officers[2]}')
print(f'{officers[1]} cannot attend dinner')
officers[1]='David Berger'
print(officers)
print(f'It would be an honor to eat dinner with {officers[0]}')
print(f'It would be an honor to eat dinner with {officers[1]}')
print(f'It would be an honor to eat dinner with {officers[2]}')
```

```
It would be an honor to eat dinner with Jimmy Carter
It would be an honor to eat dinner with John McCain
It would be an honor to eat dinner with Wendy B. Lawrence
John McCain cannot attend dinner
['Jimmy Carter', 'David Berger', 'Wendy B. Lawrence']
It would be an honor to eat dinner with Jimmy Carter
It would be an honor to eat dinner with David Berger
It would be an honor to eat dinner with Wendy B. Lawrence
```

3-6. More Guests: You just found a bigger dinner table, so now more space is available. Think of three famous Army Officers to invite to dinner. A joint operation!

- Start with the code from above. Add a `print()` call to the end of your program informing people that you found a bigger dinner table.
- Use `insert()` to add one new guest to the beginning of your list.
- Use `insert()` to add one new guest to the middle of your list.
- Use `append()` to add one new guest to the end of your list.
- Print a new set of invitation messages, one for each person in your list.

In [10]:
```
officers = ['Jimmy Carter','John McCain', 'Wendy B. Lawrence']
print(f'It would be an honor to eat dinner with {officers[0]}')
```

```
print(f'It would be an honor to eat dinner with {officers[1]}')
print(f'It would be an honor to eat dinner with {officers[2]}')
print(f'{officers[1]} cannot attend dinner')
officers[1]='David Berger'
print(officers)
print(f'It would be an honor to eat dinner with {officers[0]}')
print(f'It would be an honor to eat dinner with {officers[1]}')
print(f'It would be an honor to eat dinner with {officers[2]}')
### cant get the following line out of brackets? Have to reference individually?
print(f'{officers[0]}, {officers[1]}, {officers[2]} I found a bigger dinner table!')
#for name in officers:
    #strList = print(name)

#print(f'{strList[0]}, I found a bigger dinner table!')
```

```
It would be an honor to eat dinner with Jimmy Carter
It would be an honor to eat dinner with John McCain
It would be an honor to eat dinner with Wendy B. Lawrence
John McCain cannot attend dinner
['Jimmy Carter', 'David Berger', 'Wendy B. Lawrence']
It would be an honor to eat dinner with Jimmy Carter
It would be an honor to eat dinner with David Berger
It would be an honor to eat dinner with Wendy B. Lawrence
Jimmy Carter, David Berger, Wendy B. Lawrence I found a bigger dinner table!
```

3-7. Shrinking Guest List: You just found out that your new dinner table won't arrive in time for the dinner and you have space for only two guests.

- Start with the program above. Add a new line that prints a message saying that you can invite only two people for dinner.
- Use `pop()` to remove guests from your list one at a time until only two names remain in your list. Each time you pop a name from your list, print a message to that person letting them know you're sorry you can't invite them to dinner.
- Print a message to each of the two people still on your list, letting them know they are still invited.
- Use `del` to remove the last two names from your list, so you have an empty list. Print your list to make sure you actually have an empty list at the end of your program.

In [3]:
```
officers = ['Jimmy Carter','John McCain', 'Wendy B. Lawrence']
print(f'It would be an honor to eat dinner with {officers[0]}')
print(f'It would be an honor to eat dinner with {officers[1]}')
print(f'It would be an honor to eat dinner with {officers[2]}')
print(f'{officers[1]} cannot attend dinner')
officers[1]='David Berger'
print(officers)
print(f'It would be an honor to eat dinner with {officers[0]}')
print(f'It would be an honor to eat dinner with {officers[1]}')
print(f'It would be an honor to eat dinner with {officers[2]}')
### cant get the following line out of brackets? Have to reference individually?
print(f'{officers[0]}, {officers[1]}, {officers[2]} I found a bigger dinner table!')
print('I can only invite two people to dinner.')
print(f"I'm sorry {officers.pop()} I don't have room for you.")
print(f"{officers[0]} and {officers[1]}, you are still invited to dinner.")
del officers[1]
del officers[0]
print(officers)
```

```
It would be an honor to eat dinner with Jimmy Carter
It would be an honor to eat dinner with John McCain
It would be an honor to eat dinner with Wendy B. Lawrence
John McCain cannot attend dinner
['Jimmy Carter', 'David Berger', 'Wendy B. Lawrence']
It would be an honor to eat dinner with Jimmy Carter
It would be an honor to eat dinner with David Berger
It would be an honor to eat dinner with Wendy B. Lawrence
Jimmy Carter, David Berger, Wendy B. Lawrence I found a bigger dinner table!
I can only invite two people to dinner.
I'm sorry Wendy B. Lawrence I don't have room for you.
Jimmy Carter and David Berger, you are still invited to dinner.
[]
```

3-8. Seeing the World: Think of at least five places in the world you'd like to visit.

- Store the locations in a list. Make sure the list is not in alphabetical order.
- Print your list in its original order. Don't worry about printing the list neatly, just print it as a raw Python list.
- Use `sorted()` to print your list in alphabetical order without modifying the actual list.
- Show that your list is still in its original order by printing it.
- Use `sorted()` to print your list in reverse alphabetical order without changing the order of the original list.
- Show that your list is still in its original order by printing it again.
- Use `reverse()` to change the order of your list. Print the list to show that its order has changed.
- Use `reverse()` to change the order of your list again. Print the list to show it's back in its original order.
- Use `sort()` to change your list so it's stored in reverse alphabetical order. Print the list to show that its order has changed.

In [15]:
```
places = ["France", "Spain", "Budapest", "Australia", "Hawaii"]
print(places)
print(sorted(places))
print(places)
print(sorted(places,reverse=True))
print(places)
places.reverse()
print(places)
places.reverse()
print(places)
places.sort(reverse=True)
print(places)
```

```
['France', 'Spain', 'Budapest', 'Australia', 'Hawaii']
['Australia', 'Budapest', 'France', 'Hawaii', 'Spain']
['France', 'Spain', 'Budapest', 'Australia', 'Hawaii']
['Spain', 'Hawaii', 'France', 'Budapest', 'Australia']
['France', 'Spain', 'Budapest', 'Australia', 'Hawaii']
['Hawaii', 'Australia', 'Budapest', 'Spain', 'France']
['France', 'Spain', 'Budapest', 'Australia', 'Hawaii']
['Spain', 'Hawaii', 'France', 'Budapest', 'Australia']
```

3-9. Dinner Guests: Working with one of the programs from **3-4** through **3-7**, use `len()` to print a message indicating the number of people you are inviting to dinner.

In [18]:
```
officers = ['Jimmy Carter','John McCain', 'Wendy B. Lawrence']

print(f"I am inviting {len(officers)} people to dinner.")
```

```
I am inviting 3 people to dinner.
```

3-10. Every Function: Think of something you could store in a list. For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you'd like. Write a program that creates a list containing these items and then uses each function introduced in this chapter at least once.

In [30]:
```
## all functions introduced: indexing, change an element, .append, .insert, del, .pop,

states = ['Georgia', 'Flordia', 'Texas', 'Alabama', 'Ohio']
print(states)
print(f'I am from {states[0]}!')
states[4] = 'Virginia'
print(states)
states.append('Missouri')
print(states)
states.insert(3,'New York')
print(states)
del states[5]
print(states)
states.pop()
print(states)
states.remove('Texas')
print(states)
states.sort()
print(states)
states.sort(reverse=True)
print(states)
print(len(states))
```

```
['Georgia', 'Flordia', 'Texas', 'Alabama', 'Ohio']
I am from Georgia!
['Georgia', 'Flordia', 'Texas', 'Alabama', 'Virginia']
['Georgia', 'Flordia', 'Texas', 'Alabama', 'Virginia', 'Missouri']
['Georgia', 'Flordia', 'Texas', 'New York', 'Alabama', 'Virginia', 'Missouri']
['Georgia', 'Flordia', 'Texas', 'New York', 'Alabama', 'Missouri']
['Georgia', 'Flordia', 'Texas', 'New York', 'Alabama']
['Georgia', 'Flordia', 'New York', 'Alabama']
['Alabama', 'Flordia', 'Georgia', 'New York']
['New York', 'Georgia', 'Flordia', 'Alabama']
4
```

3-11. Intentional Error: Change an index in one of your programs to produce an index error. Learning how to read and understand Python error messages can significantly speed up your code debugging.

In [31]:
```
states = ['Georgia', 'Flordia', 'Texas', 'Alabama', 'Ohio']
states[8]= 'not a state'
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[31], line 2
      1 states = ['Georgia', 'Flordia', 'Texas', 'Alabama', 'Ohio']
----> 2 states[8]= 'not a state'

IndexError: list assignment index out of range
```

In [ ]: