Data Structures

- data structures are different ways of storing a data set * * *
- three data structures in this notebook: * * *

Lists (ordered collection) [ ]

In [85]:

```python
# Create a list of colors
# Similar to arrays from JavaScript
colors = ["Blue", "Red", "White"]
print(colors)
```

```
['Blue', 'Red', 'White']
```

In [86]:

```python
# Add a single element to the list
# (using the 'append' method for the 'list' class)
colors.append("Purple")
print(colors)
```

```
['Blue', 'Red', 'White', 'Purple']
```

In [87]:

```python
# Add multiple elements to the list
# (using the 'extend' method for lists):
colors.extend(["Pink", "Magenta"])
print(colors)
```

```
['Blue', 'Red', 'White', 'Purple', 'Pink', 'Magenta']
```

In [88]:

```python
# Get the length of the list:
length = len(colors)
print(colors)
```

```
['Blue', 'Red', 'White', 'Purple', 'Pink', 'Magenta']
```

In [129]:

```python
# loop over the list:
sock_list = []
import random

for i in range(length):
    sock_color = colors[i]
    sock_quantity = str(random.randint(0, 10))
    # Don't forget to use str in paranthesis
    sock_message = "I have " + sock_quantity + " pairs of " + sock_color + " sock
    sock_list.append(sock_message)

print(sock_message)
```

```
I have 8 pairs of Magenta socks.
```

Searching

- Sytax
  - "item" in "list"
  - Search for this "item" in this "list"
- Examines list one-by-one (linear)
- Typically slow for long lists

In [133]:

```
yellow = "Yellow" in colors
blue = "Blue" in colors
print(blue, yellow)
```

```
True False
```

Slicing Lists

- Syntax for slicing a list named "myList" - myList[start, end, skip]
  - default value for start is 0 (inclusive)
  - default value for end is myList.length (exclusive)
  - skip is how many elements to jump by
    - print(myList[::2]) would print every other value from beginning to end
    - print(myList[::-1]) would print all values backwards from end to beginning

In [ ]:

```
print(colors)
```

In [135]:

```
# Print the element indexed at -2
print(colors[-6])
```

```
Blue
```

In [140]:

```
# Print the elements from index 0 (inclusive) to the index 2 (exclusive)
print(colors[:2])
```

```
['Blue', 'Red']
```

In [142]:

```
# Print the elements from index 3 (inclusive) to the end of the list
print(colors[3:])
```

```
['Purple', 'Pink', 'Magenta']
```

In [143]:

```
# Print the elements from index 1 (inclusive) to index 4 (exclusive)
print(colors[1:4])
```

```
['Red', 'White', 'Purple']
```

In [147]:

```
# Print the list backwards
print(colors[::-1])
```

```
['Magenta']
```

In [148]:
```
# Check the data type
type(colors)
```

Out[148]:
```
list
```

In [151]:
```
# Slicing also works on Strings
sentence = "It was a dark and stormy night."

# Slice out "dark and stormy night"
print(sentence[9:])
```

```
dark and stormy night.
```

In [1]:
```
# Reveal the secret message in an Acrostic
    # Hint: Look at the first letter of each line

# An Acrostic by Edgar Allan Poe
acrostic = """Elizabeth it is in vain you say
Love not — thou sayest it in so sweet a way:
In vain those words from thee or L.E.L.
Zantippe’s talents had enforced so well:
Ah! if that language from thy heart arise,
Breath it less gently forth — and veil thine eyes.
Endymion, recollect, when Luna tried
To cure his love — was cured of all beside —
His follie — pride — and passion — for he died."""

secret_message = ""                  # acrostic = ["Elizabeth it is in vain you say
acrostic = acrostic.split('\n')      #              "Love not — thou sayest it in so
for line in acrostic:                #              "In vain those words from thee o
    secret_message += line[0]        #              "Zantippe’s talents had enforced
                                     #               ... ]
print(secret_message)
```

```
ELIZABETH
```

Tuples( )

- Use ( ) or nothing, but stay consistent
    - (a,b) <=> a,b
- Similar to lists, but cannot modify them

In [4]:
```
# Use tuples for storing/assigning multiple values in the same line
a, b, c = 82, 47, 8247
print(a)
print(b)
print(c)
```

```
82
47
8247
```

In [11]:

```python
# How to swap elements in Python
# Start with variables x and y
x, y = "Red", "Ballon"
print(x)
print(y)
```

```
Red
Ballon
```

In [12]:

```python
# Swap using tuples
x,y = y,x
print(x)
print(y)

# In other languages, this would take 3 lines of code and a temporary variable:
    # temp = x
    # x = y
    # y = temp
```

```
Ballon
Red
```

In [31]:

```python
# Define a function that takes two numbers as parameters, and returns the tuple
import random
def sum_product(x, y):
    s = x + y
    p = x * y
    return s, p

sp = sum_product(4, 7)
print(sp)
```

```
(53, 0)
```

In [17]:

```python
# If you don't know what type of data you are working with, just check!
print(type(sp))
```

```
<class 'tuple'>
```

In [33]:

```python
# Assigning tuples with return values
a, b = sum_product(4, 7)
print(a, b)
```

```
11 28
```

In [34]:

```python
# Check what type of data the variables are
print(type(a))
print(type(b))
```

```
<class 'int'>
<class 'int'>
```

Dictionaries

- Use { }
- Associate "keys" with "values"
    - key:value
- Don't take the word dictionary too literally

Keys are immutable (unchangable) and must be unique (no duplicates)

In [41]:

```
# Dictionary keys: Student ID numbers
# Dictionary values: Student names
    # Are the keys unique in this example?
idNames = {8717:"J. Cole", 2309: "Kendrick Lamar", 4008:"Drake", 9904:"Cardi B.",
print(idNames)
```

```
{8717: 'J. Cole', 2309: 'Kendrick Lamar', 4008: 'Drake', 9904: 'Cardi B.', 157
6: 'J. Cole'}
```

In [39]:

```
# Dictionary lookup is very fast
    # Use the "key" to retrieve its associated "value"
print(idNames[9904])
print(idNames[8717])
```

```
Cardi B.
J. Cole
```

In [42]:

```
# Assign new value pairs
idNames[1822] = "Post Malone"
idNames[3310] = "Logic"
print(idNames)
```

```
{8717: 'J. Cole', 2309: 'Kendrick Lamar', 4008: 'Drake', 9904: 'Cardi B.', 157
6: 'J. Cole', 1822: 'Post Malone', 3310: 'Logic'}
```

In [43]:

```
# Check the length of the dictionary
length = len(idNames)
print(length)
```

```
7
```

In [48]:

```
# Search through a dictionary similar to lists
    # Typically slow for a list
    # Typically fast for a dictionary!
print(2309 in idNames)
```

```
True
```

In [ ]:

```
# Try a key that is not in our dictionary:
```

Name: Saipranav venkatakrishnan

Date:2/16/2020

Task 1

- use a for loop to create a list of 31 random numbers between -5 and 35
    - Example: [-4, 31, 0, 1, 8, 19, 8,..., 22]

In [52]:

```
import random
numList = []
for i in range(31):
    numList.append(random.randint(-5, 35))
print(numList)
```

```
[30, 31, 19, 12, 32, -5, 35, 9, 7, -1, 28, -1, 9, 34, 9, -4, 0, 22, 32, 2, 22,
31, 35, 13, 25, 6, -4, 14, 26, -5, 7]
```

Task 2

- 1) Check if 32 degrees appears in your list of temperatures
- 2) Pick a temperature that does appear in your list. Write some code that shows that this temperature appears in your list of temperatures
- 3) Verify that the length of your list is 31

In [53]:

```
length = len(numList)
print("The length of the tempreture list is " + str(length))
print(str(numList[random.randint(0, 31)]) + " degrees appears on the list")
if 32 in numList:
    print("32 degrees appears on the list")
else:
    print("32 doesn't degrees appears on the list")
```

```
The length of the tempreture list is 31
7 degrees appears on the list
32 degrees appears on the list
```

Task 3

- use print and your list to write a for loop in the following form:
    - The temperature on January 1 was -4 degrees F.
    - The temperature on January 2 was 31 degrees F.
    - The temperature on January 3 was 0 degrees F.
    - ...
    - The temperature on January 31 was 22 degrees F.

In [63]:

```
temp = []
for i in range(31):
    number = numList[i]
    new_line = "\n"
    temp.append("The tempreture on January " + str(i+1) + " is " + str(number) +
print(str(temp))
```

```
['The tempreture on January 1 is 30 degrees Farenheit', 'The tempreture on Janu
ary 2 is 31 degrees Farenheit', 'The tempreture on January 3 is 19 degrees Fare
nheit', 'The tempreture on January 4 is 12 degrees Farenheit', 'The tempreture
on January 5 is 32 degrees Farenheit', 'The tempreture on January 6 is -5 degre
es Farenheit', 'The tempreture on January 7 is 35 degrees Farenheit', 'The temp
reture on January 8 is 9 degrees Farenheit', 'The tempreture on January 9 is 7
degrees Farenheit', 'The tempreture on January 10 is -1 degrees Farenheit', 'Th
e tempreture on January 11 is 28 degrees Farenheit', 'The tempreture on January
12 is -1 degrees Farenheit', 'The tempreture on January 13 is 9 degrees Farenhe
it', 'The tempreture on January 14 is 34 degrees Farenheit', 'The tempreture on
January 15 is 9 degrees Farenheit', 'The tempreture on January 16 is -4 degrees
Farenheit', 'The tempreture on January 17 is 0 degrees Farenheit', 'The tempret
ure on January 18 is 22 degrees Farenheit', 'The tempreture on January 19 is 32
degrees Farenheit', 'The tempreture on January 20 is 2 degrees Farenheit', 'The
tempreture on January 21 is 22 degrees Farenheit', 'The tempreture on January 2
2 is 31 degrees Farenheit', 'The tempreture on January 23 is 35 degrees Farenhe
it', 'The tempreture on January 24 is 13 degrees Farenheit', 'The tempreture on
January 25 is 25 degrees Farenheit', 'The tempreture on January 26 is 6 degrees
Farenheit', 'The tempreture on January 27 is -4 degrees Farenheit', 'The tempre
ture on January 28 is 14 degrees Farenheit', 'The tempreture on January 29 is 2
6 degrees Farenheit', 'The tempreture on January 30 is -5 degrees Farenheit',
'The tempreture on January 31 is 7 degrees Farenheit']
```

Challenge

- use lists and math operators to write a function hex_converter(input) that takes a decimal number as input and returns its hexadecimal equivalent as output
  - show the output for hex_converter(255), hex_converter(10), hex_converter(851) to show that your algorithm works

In [ ]: