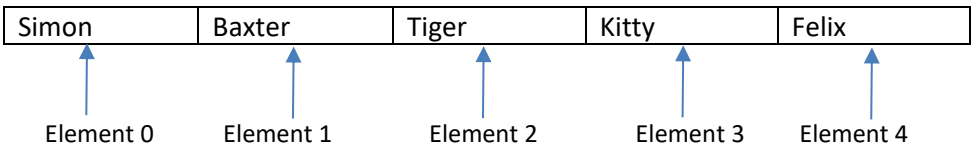
CIS 125 Principles of Programming Logic

Python Lists

- Many programming languages have a data structure known as an "**array**". Python has both "**arrays**" and "**lists**". We will cover lists here. They share many **similar characteristics** as arrays:
    - Both used to store (related) data in memory
    - Both are mutable (can be changed)
    - Both are indexed and iterated through (typically with a loop)
    - Both can be sliced and manipulated/managed with functions
        - Count, min, max, average, sort, etc.

- Lists are commonly used because they are **built into Python** and flexible.
    - Arrays must be brought in through either: a) the array module or b) the external library - numpy – that must also be installed.
    - Arrays can, however, perform more sophisticated numerical computations (sch as in data science) and can be more efficient for storing large amounts of data.
    - Lists can store heterogenous (different) data types; arrays store homogeneous (same) data types.
    - Lists can be resized/grow.

- Below is a visual depiction of a list cat names:



Example: A list of cats

```
cats = ['Simon', 'Baxter', 'Tiger', 'Kitty', 'Felix']

print(cats)
print(cats[2])
```

Output (raw/unformatted):

```
['Simon', 'Baxter', 'Tiger', 'Kitty', 'Felix']
Tiger
```

Python has tuples and dictionaries. Their differences are:

- **Lists** - mutable/values can be changed.
- **Tuples** – Tuples are just like lists, but non-mutable/can't change their values.
- **Dictionaries** - Dictionaries are similar to what some languages refer to as an associate array or map. The values in a dictionary aren't numbered but accessed by a key and value. They are unordered and changeable.

Example: A tuple of months (note use of parentheses instead of brackets)

```
months = ('January','February','March','April','May','June',\
'July','August','September','October','November','  December')

print (months)
print (months[1])
```

Output:

```
('January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', '
December')
February
```

Example: Python list – append and extend

```
cats = ['Simon', 'Baxter', 'Tiger', 'Kitty', 'Felix']
print(cats)
cats.extend(['George'])
cats.append('Catherine')
print(cats)
```

Output:

```
['Simon', 'Baxter', 'Tiger', 'Kitty', 'Felix']
['Simon', 'Baxter', 'Tiger', 'Kitty', 'Felix', 'George', 'Catherine']
```

Note:

- **Append** adds an element to the end of the list, **extend** concatenates the first list with another list (or another iterable not necessarily a list).
    - Append appends a single element. Extend appends a list of elements. If you pass a list to append, it still adds one element.

Example: Python list – Creating, displaying, adding to, deleting from, sorting, counting, min, etc.

```
scores=[90,85,73,94,85,88]
print (scores)
print ("Scores: ", scores)
print (scores[2])
print (scores[1:4])
print (scores[:4])
print (scores[3:])
print ("High score: ", max(scores))
print ("Low score: ", min(scores))
print ("Elements in array: ", len(scores))
print ("Sorted scores: ", sorted(scores))    # this displays the items sorted
print ("Scores: ", scores)
scores.sort()                # this actually changes the list order
print ("Scores: ", scores)
scores.extend([98, 65])    # add/append two scores to end of list
print ("Scores: ", scores)
print ("High score: ", max(scores))
scores.insert(1, 81)         # insert a new score at position 1
print ("Scores: ", scores)
del scores[7:]               # also works on dictionaries
print ("Scores: ", scores)
count1 = scores.count(85)
print ("# of 85 scores: ", count1)
```

Output:

```
[90, 85, 73, 94, 85, 88]
Scores:  [90, 85, 73, 94, 85, 88]
73
[85, 73, 94]
[90, 85, 73, 94]
[94, 85, 88]
High score:  94
Low score:  73
Elements in array:  6
Sorted scores:  [73, 85, 85, 88, 90, 94]
Scores:  [90, 85, 73, 94, 85, 88]
Scores:  [73, 85, 85, 88, 90, 94]
Scores:  [73, 85, 85, 88, 90, 94, 98, 65]
High score:  98
Scores:  [73, 81, 85, 85, 88, 90, 94, 98, 65]
Scores:  [73, 81, 85, 85, 88, 90, 94]
# of 85 scores:  2
```

Example: **Looping** through list with two types of for loops

```
employees=["Smith, Dan", "Jones, Mary", "Williams, Nick"]
print (employees)
for employee in employees:
      print("Employee: ", employee)
for i in range(len(employees)):
    print(i+1, employees[i])
```

Output:

```
['Smith, Dan', 'Jones, Mary', 'Williams, Nick']
Employee:  Smith, Dan
Employee:  Jones, Mary
Employee:  Williams, Nick
1 Smith, Dan
2 Jones, Mary
3 Williams, Nick
```

Example: Using a **while loop** to output contents of list/array

```
foods = ["Cheesecake","Steak","Donuts","Apples","Bread","Lettuce"]

x = 0
while x < len(foods):
    print(foods[x])
    x += 1
```

Output:

Cheesecake
Steak
Donuts
Apples
Bread
Lettuce

Example: Arrays – **Removing items**, looping through, searching for certain items (one dimensional)

```
scores=[90,85,73,94,85,88]

print ("Scores: ", scores)
scores += [97]   # add an item to end
print ("Scores: ", scores)
scores.remove(85)  # remove first instance of 85
scores.remove(85)  # remove next instance of 85
print ("Scores: ", scores)

i = 0
count = len(scores)
while i < count:
   if scores[i] >=90:
      print ("Scores:", scores[i])
   i+=1

scores2 = [i for i in scores if i >= 90] # create new list containing only
                                        #    scores >= 90
print ("Scores: ", scores2)
average = sum(scores) / len(scores)
print ("Average: ", average)
```

Output:

```
Scores:  [90, 85, 73, 94, 85, 88]
Scores:  [90, 85, 73, 94, 85, 88, 97]
Scores:  [90, 73, 94, 88, 97]
Scores: 90
Scores: 94
Scores: 97
Scores:  [90, 94, 97]
Average:  88.4
```

Example: Arrays – Alternate method of **finding an element** in an array with an IF statement

```
foods = ["Cheesecake","Steak","Donuts","Apples","Bread","Lettuce","Cottage cheese",
"Pears"]

choice = input("Enter food to search for: ")
if choice in foods:
    print("We have that food in stock.")
else:
    print("We do not have that food in stock.")
```

Example: More methods of **removing items** from list

```
employees=["Dan", "Mary", "Nick", "Jane", "Ali"]
del(employees[1])     # delete particular item at index location
employees.remove('Dan')
for employee in employees:
    print("Employee: ", employee)
print()
employees.pop(0)   # returns item removed
for employee in employees:
    print("Employee: ", employee)
```

Output:

```
Employee:   Nick
Employee:   Jane
Employee:   Ali

Employee:   Jane
Employee:   Ali
```

Question: What is the output of this program?

```
sports=["Soccer", "Football", "Volleyball", "Tennis", "Baseball", "Skiing"]
print(sports[3])
```

Write output here:

Question: What is the output of this program?

```
sports=["Soccer", "Football", "Volleyball", "Tennis", "Baseball", "Skiing"]
sports.sort()
del sports[3:]
for sport in sports:
    print(sport)
```

Write output here:

Question: What is the output of this program?

```
scores=[100, 88, 55, 99]
for score in scores:
    if score >= 60:
        print("Pass")
    else:
        print("Fail")
```

Write output here:

Task: Write a program that does the following:

- Utilizes a while loop
- Asks for user input of a student first name
- Adds the first name to a list of student names
- Stops the loop when the word 'stop' is entered, i.e. a sentinel value
- Once the loop has ended, outputs the list of student names one per line

```
names = []
name = input("Enter employee name: ").capitalize()

while name != "Stop":
    names += [name]
    name = input("Enter employee name: ").capitalize()

for name in names:
    print(name)
```

Example: Arrays – **Adding** and removing items from a list and calculating an **average**

```
scores = [90,85,73,94,85,88]

print ("Scores: ", scores)
scores += [97]                          # add an item to end
print ("Scores: ", scores)
scores.remove(85)                       # remove first instance of 85
scores.remove(85)                       # remove next instance of 85
print ("Scores: ", scores)
average = sum(scores) / float(len(scores))
print ("Average: ", average)
```

Output:

```
Scores:    [90, 85, 73, 94, 85, 88]
Scores:    [90, 85, 73, 94, 85, 88, 97]
Scores:    [90, 73, 94, 88, 97]
Average:    88.4
```

Multi-Dimensional Lists

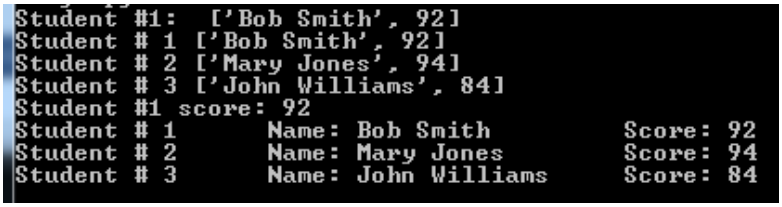
Example: **Two Dimensional Arrays**

```
quiz1=[["Bob Smith",92],["Mary Jones",94],["John Williams",84]]
print ("Student #1: ", quiz1[0])

count = len(quiz1)
for x in range(count):
        print ("Student #", x+1, quiz1[x])

print ("Student #1 score:", quiz1[0][1])
for x in range(count):
        print ("Student #", x+1, "\t Name:", quiz1[x][0], "\t Score:", quiz1[x][1])
```

Output:



Example: **Multi-dimensional array** (list of lists)

```
friends = []
friends.append(["Tom Smith","Blue",33])
friends.append(["Karen Jones","Red",35])
friends.append(["Alex Dillon","Black",41])

print(friends)
print(friends[1][0],"'s favorite color is",friends[1][1])
```

Output:

```
[['Tom Smith', 'Blue', 33], ['Karen Jones', 'Red', 35], ['Alex Dillon', 'Black', 41]]
Karen Jones 's favorite color is Red
```

Input Validation

Input validation is a critical area of programming. It serves the following purposes:

- Ensure valid data entered into program/system.
    - Famous saying: GIGO -> Garbage In, Garbage out

- Ensure inputs do not present or allow malicious security threats into the program/system.
    - SQL injections/attacks
    - XSS attacks
    - Buffer overflow attacks
    - Brute force/dictionary attacks
    - Preventing the program/system from crashing
    - Etc.

The input validation may be performed by **loops, decision statements, and/or try… catch statements**.

- Some languages have **built-in library functions** for input validation. PHP, for example, has functions to look for special characters that might be inputted/used to hack into a web site.
- We performed input validation earlier with our password checker program.
- Placing the input validation in a user-defined function may organize the code well.

Common input validation and types of errors to consider:

- Empty input, where a user accidentally hits enter before entering data
- The user enters the wrong type of data, e.g. string vs. integer
- State abbreviations should be 2-character strings
- Zip codes should be in the proper format of 5 or 9 digits
- Hourly wages and salary amounts should be numeric values and within ranges
- Dates should be checked, e.g. format, min/max date, etc.
- Time measurements should be checked
- Check for reasonable numbers
- Password does not contain letters, numbers, mixed case and 10 or more characters
- Patient blood pressure must be numeric and within valid numeric range

Example: **While loop user input validation**

```
while True:
    pass1 = input("Enter a password: ")
    if len(pass1) < 10:
        print("Invalid password.  Must be at least 10 characters.")
    else:
        print("Valid password.")
        break
```

Output:

```
Enter a password: dd
Invalid password.
Enter a password: 33e33e
Invalid password.
Enter a password: DDdddd33333
Valid password.
```

Example: Password check with function call

```
def checkPass():
    while True:
        pass1 = input("Enter a password: ")
        if len(pass1) < 10:
            print("Invalid password.  Must be at least 10 characters.")
        else:
            print("Valid password.")
            break
    return pass1

pass1 = checkPass()
```

Method #2:

```
def checkPass(pass1):
    while True:
        if len(pass1) < 10:
            print("Invalid password.  Must be at least 10 characters.")
            pass1 = checkPass(input("Enter a password: "))
        else:
            break
    return pass1

pass1 = checkPass(input("Enter a password: "))
```

Example: Checking password for mixed case

```
while True:
    pass1 = input("Please enter a password mixed case: ")
    if pass1.islower() or pass1.isupper():
        print("Invalid input.  Please enter a password mixed case: ")
    else:
        print("Valid password.  Thank you.")
        break
```

Output:

```
Please enter a password mixed case: dd
Invalid input.    Please enter a password mixed case:
Please enter a password mixed case: DD
Invalid input.    Please enter a password mixed case:
Please enter a password mixed case: ddDD
Valid password.    Thank you.
```

Example: Alternate method of mixed case password user input validation

```
pass1 = input("Enter password: ")
while (pass1.islower() or pass1.isupper()):
    pass1 = input("  >> Invalid password.  Please re-enter password (must be
mixed-case): ")
print(pass1, "is a valid password")
```

Output:

```
Enter password: dfljfljdk
   >> Invalid password.    Please re-enter password (must be mixed-case): FDFFFF
   >> Invalid password.    Please re-enter password (must be mixed-case): dfkdfjFDFDDF
dfkdfjFDFDDF is a valid password
```

Example: **Numeric Range Check** in a Function

```
def checkSalary(salary):
    while salary < 1 or salary > 200000:
        salary = float(input("Invalid salary amount; please re-enter amount
between 1 and 200000: "))
    return salary

amount = checkSalary(float(input("Enter salary amount (1-200000): ")))
print("%.2f" % amount, "is a valid salary amount")
```

Output:

Enter salary amount (1-200000): -1
Invalid salary amount; please re-enter amount between 1 and 200000: 350000
Invalid salary amount; please re-enter amount between 1 and 200000: 35000
35000.00 is a valid salary amount

Input Validation (with lists and correct data type)

Example: String valid choice in list (Python array)

```
choice = "Y"
valid = ("Y","YES","N", "NO")
yes_list = ("Y","y","yes","Yes","YES")
while choice in yes_list:
    weight = float(input("How much do you weight? "))
    height = float(input("How tall are you in inches? "))
    bmi = 703 * (weight / (height * height))
    print ("Your BMI is: %.2f" % bmi)
    choice = input("Would you like to make another BMI calculation (Y/N)?
").upper()
    while choice not in valid:
        choice = input("Invalid choice.  Enter a Y or N? ").upper()
```

Output:

How much do you weight? 120
How tall are you in inches? 62
Your BMI is: 21.95
Would you like to make another BMI calculation (Y/N)? dog
Invalid choice. Enter a Y or N? y
How much do you weight? 125
How tall are you in inches? 64
Your BMI is: 21.45
Would you like to make another BMI calculation (Y/N)? n

Example: String valid choice in list (Python array)

```
yes_list = ["Y","N","YES","NO"]
no_list = ["N","NO"]

def calcAreaCircle(rad):
      area = 3.14159 * (rad ** 2)
      print("Area of circle is %.2f" % area)

while True:
    rad = int(input("Enter a radius: "))
    calcAreaCircle(rad)

    againYN = input("Perform another calculation (y/n?) ").upper()
    while againYN not in yes_list:
        againYN = input("Invalid choice.  Please enter y or n: ").upper()
    if againYN in no_list:
        break
```

Example: Menu Choice Validation

```
def main_menu():
    while True:
        print("MAIN MENU")
        print("-----------------")
        print("1. Print pay check")
        print("2. Change benefits")
        print("3. Exit")
        choice = input("Chose menu option: ")
        if choice == "1" or choice == "2":
            input("This is a stub - complete later")
        elif choice == "3":
            break
        else:
            print("\n Not Valid Choice Try again \n")

main_menu()
```

Example: Menu Choice Validation (Method 2)

```
def main_menu():
    while True:
        print("MAIN MENU")
        print("-----------------")
        print("1. Print pay check")
        print("2. Change benefits")
        print("3. Exit")
        choice = input("Chose menu option: ")
        while choice not in ("1", "2", "3"):
            choice = input("\nInvalid choice.  Please enter 1-3: ")
        return choice

choice = main_menu()
```

Try … Catch

Input Validation that can address Incorrect Data Types

Try… catch statements are like an IF…. ELSE

```
a=input("Amount: ")

try:                                          # IF
    int(a)
except ValueError:
    print(a, "is not an integer number")      # ELSE
else:
    print (a, "is an integer number")         # THEN
```

Example: Simple try… catch to validate integer number

```
a=input("Amount: ")

try:
    int(a)
except ValueError:
    print(a, "is not an integer number")
else:
    print (a, "is an integer number")
```

Output:

Amount: dog
dog is not an integer number

Example: try… catch to validate integer number inside while loop with **casting (convert data type)**

```
while True:
    a=input("Amount: ")
    try:
        int(a)
    except ValueError:
        print(a, "is not an integer number")
    else:
        print (a, "is an integer number")
        a = int(a)
        break
print("\nTotal: ", a * 5)
```

Output:

```
Amount: f
f is not an integer number
Amount: 3.3
3.3 is not an integer number
Amount: 2
2 is an integer number

Total:  10
```

Example: Try … catch in a user-defined function with returned value casted in return

```
def validAmount():
    while True:
        a=input("Amount: ")
        try:
            int(a)
        except ValueError:
            print(a, "is not an integer number")
        else:
            return int(a)

a = validAmount()
print(a, "squared =", a * a)
```

Output:

```
Amount: 9.9
9.9 is not an integer number
Amount: house
house is not an integer number
Amount: 10
10 squared = 100
```

Example: Try catch differentiating all three data types (int, float, string)

```
a=input("Amount:")

try:
   int(a)
except ValueError:
   try:
      float(a)
   except ValueError:
      print ("This is not a number")
   else:
      print (a, "is a float number")
else:
   print (a, "is an integer number")
```

Example: Float and numeric range validation with loop (includes type casting)
Program also calls function sending user input into it.

```
def validHeight(cm):
    try:
        cm = float(cm)
        return 100 <= cm <= 250
    except ValueError:
        return False

checkHeight = validHeight(input("Please enter height: "))
while not checkHeight:
    print("Invalid height")
    checkHeight = validHeight(input("Please enter height: "))

print("Valid height")
```

Output:

Please enter height: 99
Invalid height
Please enter height: mouse
Invalid height
Please enter height: 120
Valid height

Example: Validate Data Type Using a Function (try integer combined with user input)

```
while True:
    try:
        x = int(input("Please enter a number: "))
        print("Total test: ", x * 10)
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

Output:

```
Please enter a number: 3.3
Oops!  That was no valid number.  Try again...
Please enter a number: d
Oops!  That was no valid number.  Try again...
Please enter a number: 3
Total test:  30
```

Example: Previous example with function call

```
def validNum():
    while True:
        try:
            x = int(input("Please enter a number: "))
            print("Total test: ", x * 10)
            break
        except ValueError:
            print("Oops!  That was no valid number.  Try again...")
    return x

x = validNum()
print("Total test: ", x)
```

Example: User input validation loop to ensure: a) integer number, b) between 1 and 100

```
while True:
    try:
        num = int(input("Enter a number between 1 and 100: "))
        if num in range(1,100):
            break
        else:
            print("Invalid number. Try again.")
    except:
        print("That is not a number.  Try again.")
```

Example: User input validation function

```
def valid_user_input():
    try:
        return int(input("Enter a number less than 100: ")) > 100
    except ValueError:
        return False

while valid_user_input():
    print('You have entered an invalid number')

print("Valid number. Thank you.")
```

Output:

```
Enter a number less than 100: 101
You have entered an invalid number
Enter a number less than 100: 99
Valid number. Thank you.
```

Example: User input validation function

```
def inputNum(prompt):
  while True:
    try:
        userInput = int(input(prompt))
    except ValueError:
        print("Not an integer! Try again.")
    else:
        return userInput

age = inputNum("How old are you? ")

if (age >= 18):
    print("You are old enough to vote.")
else:
    print("You will be able to vote in " + str(18-age) + " year(s).")
```

Example: Alternate method for numeric range (age) check

```
def validAge():
    while True:
        try:
            age = int(input("Enter your age: "))
        except:
            print("Input input.")
        else:
            if age < 18:
                print("You are not old enough to vote.")
            else:
                print("You are old enough to vote.")
                break

validAge()
```

Example: Checking for Null (empty) Value

```
while True:
    try:
        fname= input("Please enter your first name: ")
        if not fname:
            raise ValueError('You did not enter anything.  Please re-enter.')
    except ValueError as error:
        print(error)
    else:
        print("Thank you for entering your name", fname)
        break

print("Done.")
```

Output:

Please enter your first name:
You did not enter anything. Please re-enter.
Please enter your first name: James
Thank you for entering your name James
Done.

Example: User input validation with inputted prompt

```
def validateCredits1(prompt, min, max):
    while True:
        try:
            type = int(input(prompt))
        except ValueError:
            print("Value must be between", min, "and", max, ". Please re-enter.")
            continue
        else:
            while type < 0 or type > 25:
                print("Value must be between", min, "and", max, ". Please re-
enter.")
                type = validateCredits1(prompt, min, max)
            return type
            break

credits1 = validateCredits1("How many 100-200 level credits do you plan to
register for (0-25)?", 0, 25)
```

Output:

How many 100-200 level credits do you plan to register for (0-25)?dddtt
Value must be between 0 and 25. Please re-enter.
How many 100-200 level credits do you plan to register for (0-25)?44
Value must be between 0 and 25. Please re-enter.
How many 100-200 level credits do you plan to register for (0-25)?:2

Example: Checking for Positive Integer with raise command

```
while True:
    try:
        a = int(input("Enter a positive integer: "))
        if a <= 0:
            raise ValueError("That is not a positive number.  Please re-enter.")
    except ValueError as error:
        print("Invalid input.  Please enter positive integer.")
    else:
        print("Thank you for entering a positive number.")
        break

print("Done.")
```

Output:

Enter a positive integer: cat
Invalid input. Please enter positive integer.
Enter a positive integer: -1
Invalid input. Please enter positive integer.
Enter a positive integer: 2
Thank you for entering a positive number.
Done.

Example: Try … Catch in user defined function

```
def inputNum(prompt):
    while True:
        try:
            userInput = int(input(prompt))
        except ValueError:
            print("Not an integer.  Please enter an integer.")
        else:
            return userInput

age = inputNum("Enter your age: ")
print("Your age is: ", age)
```

Example: Check for non-zero numbers

```
def checkZero():        # test for non zero numbers
    while True:
        try:
            num = float(input("Please enter number (except 0): "))
            test = 10 / num
        except:
            print("Not a valid number.")
        else:
            return num

num = checkZero()
result = 120 / num
print("120 / %.2f = %2.f" % (num, result))
```

Example: Check if file exists

```
try:     # tests if file exists (for read access)
    file = open('test22.py', "r+")
except (IOError, EOFError) as e:
    print("Unable to open file.")
else:
    print("File open.")
```

Example: Check for range

```
def validateType(prompt,lower,upper):
    while True:
        try:
            num = int(input(prompt))
        except ValueError:
            print("Invalid input.  Please re-enter.")
        else:
            if num < lower or num > upper:
                print("Invalid input.  Please re-enter.")
            else:
                print("Valid number.")
                break
    return num

num = validateType("Enter integer between 1 and 10: ",1,10)
```

Example: Menu choice input validation with user-defined function and two while loops

```
def main_menu():
    while True:
        print("MAIN MENU")
        print("-----------------")
        print("1. Print pay check")
        print("2. Change benefits")
        print("3. Exit")
        choice = input("Chose menu option: ")
        while choice not in ("1", "2", "3"):
            choice = input("Invalid menu choice.  Please enter 1-3: ")
        else:
            break
    return choice

choice = main_menu()
print("You chose menu option", choice)
```

Sample run:

```
MAIN MENU
-----------------
1. Print pay check
2. Change benefits
3. Exit
Chose menu option: cat
Invalid menu choice.    Please enter 1-3: 5
Invalid menu choice.    Please enter 1-3: 5.5
Invalid menu choice.    Please enter 1-3: 2
You chose menu option 2
```