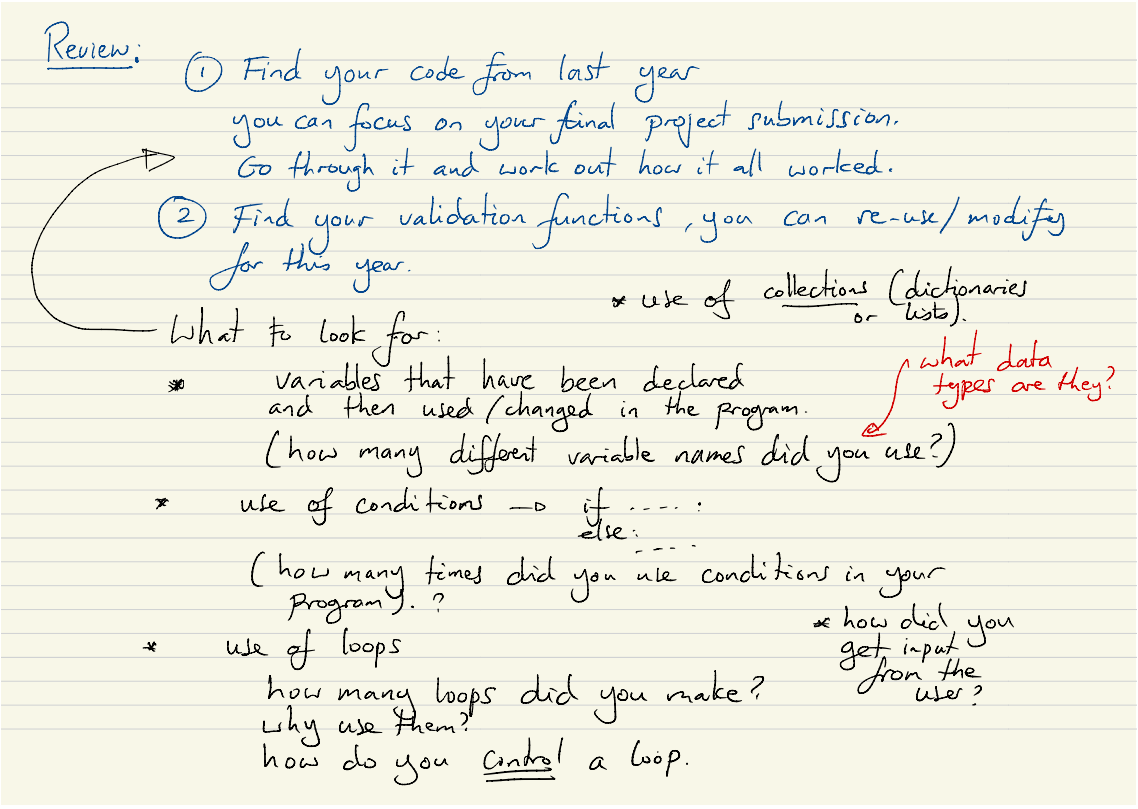
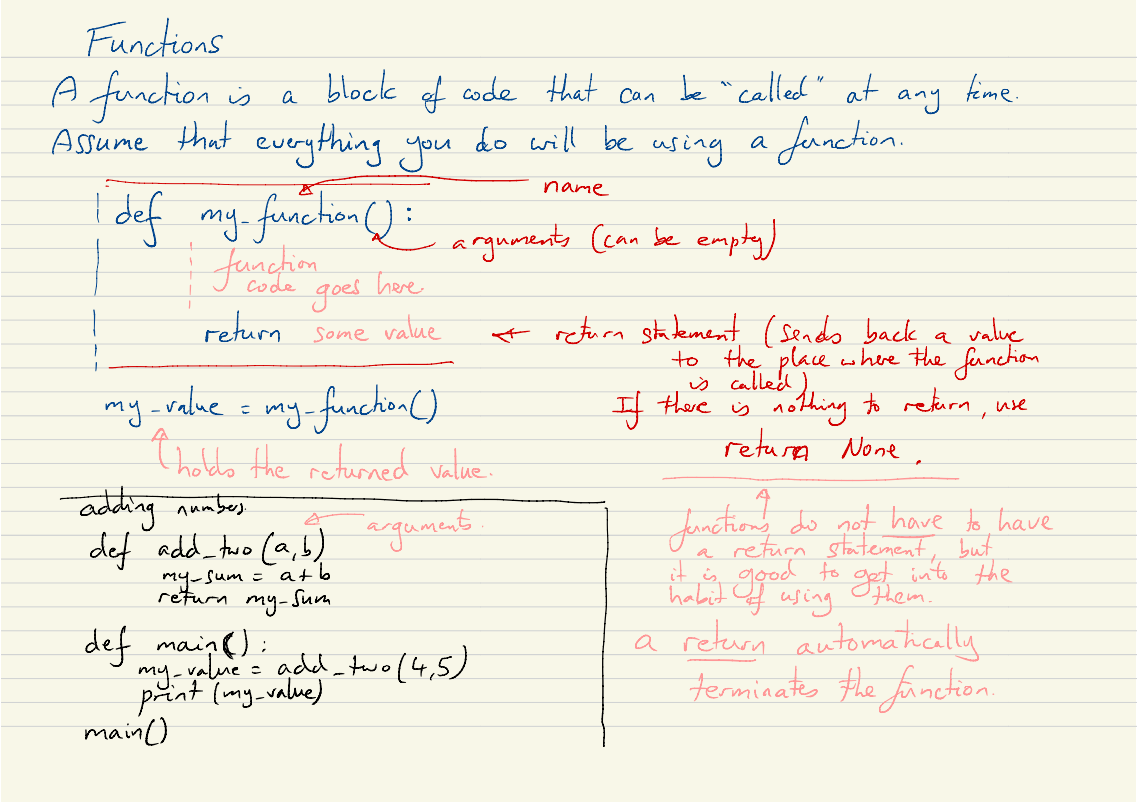
1 Review


Review:
① Find your code from last year
you can focus on your final project submission.
Go through it and work out how it all worked.
② Find your validation functions, you can re-use/modify for this year.
What to look for:
* use of collections (dictionaries or lists).
* variables that have been declared and then used/changed in the program.
(how many different variable names did you use?)
what data types are they?
* use of conditions → if ... :
else: ...
(how many times did you use conditions in your program?)
* use of loops
how many loops did you make?
why use them?
how do you control a loop.
* how did you get input from the user?

2 Functions

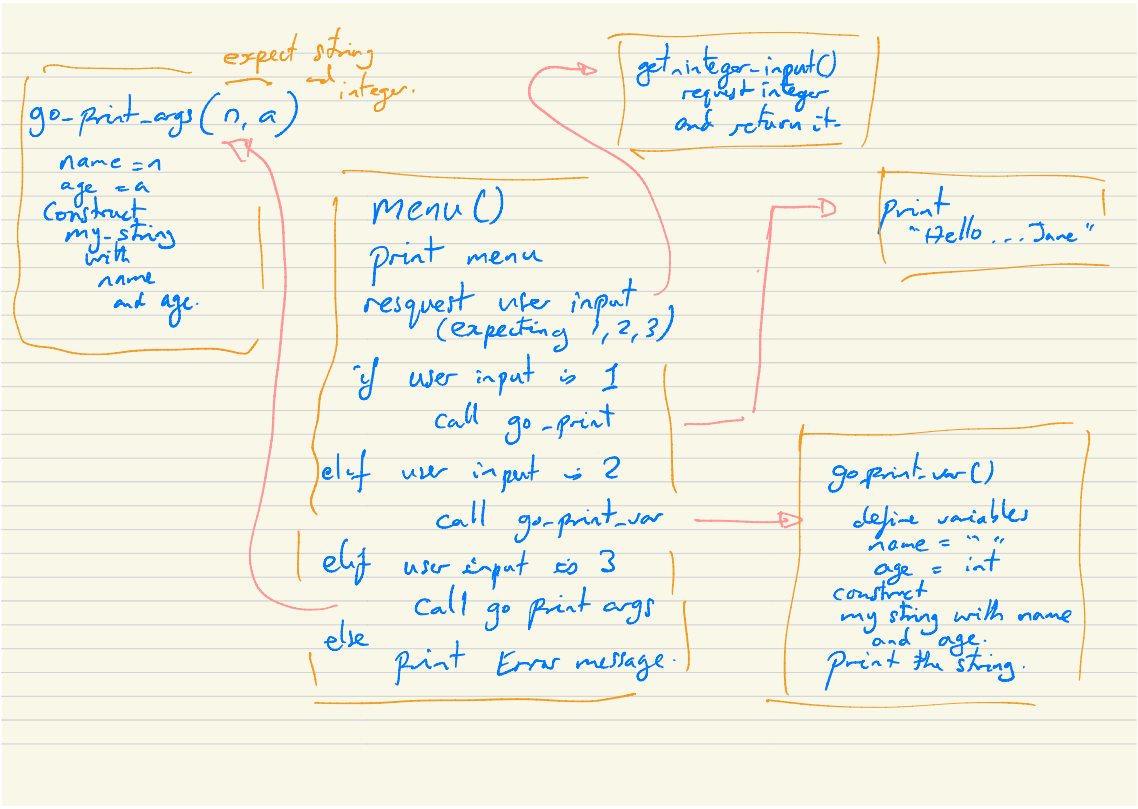


- Functions allow programmers to avoid creating code that has lots of repetitions.
- Functions allow programmers to break the structure of a program into blocks (or modules), which means that different problems can be separated out and dealt with independently.

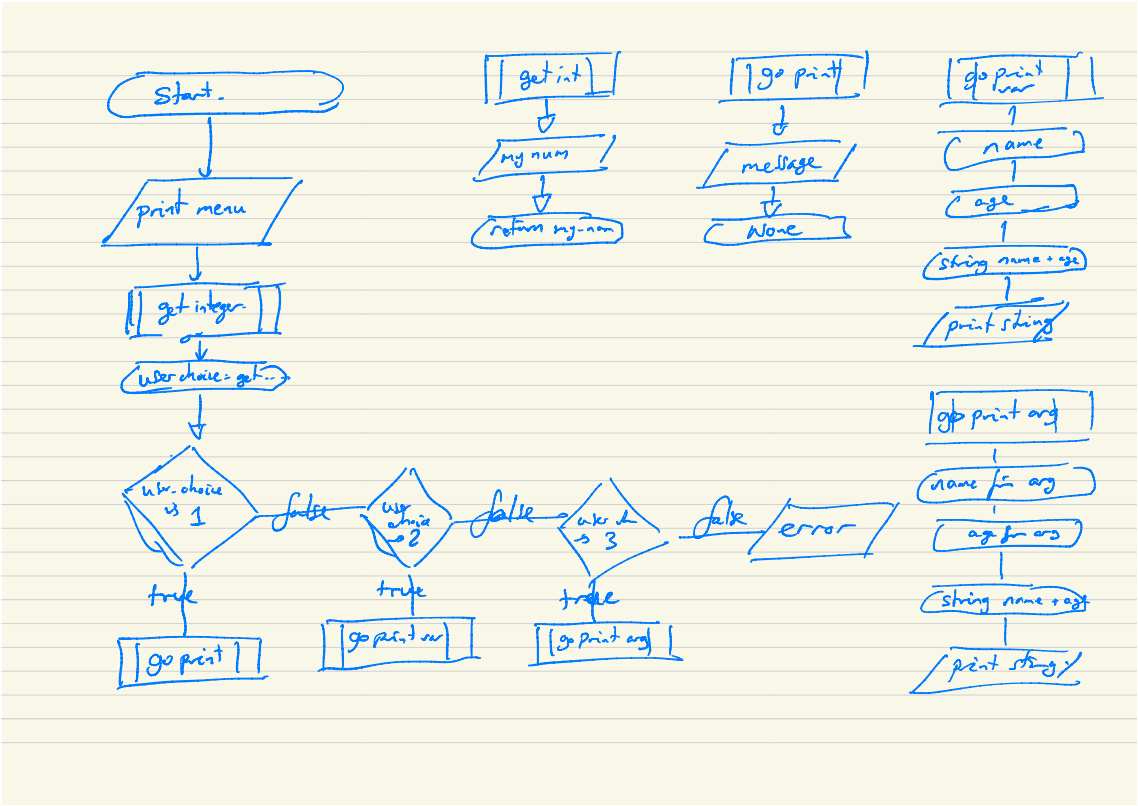
```
1
2
3  def go_print():
4      print("Hello my name is Jane")
5
6  def go_print_var():
7      name = "Bob"
8      age = 10
9      # put them together in a string
10     my_string = "My name is {} \nI am {} years old".format(name, age)
11     print(my_string)
12
13 def go_print_args(n,a):
14     name = n
15     age = a
16     my_string = "My name is {} \nI am {} years old".format(name, age)
17     print(my_string)
18
```

```
def get_integer_input():
     my_number = int(input("please enter number: -> "))
     return my_number

def menu():
     run = True
     while run is True:
          my_menu = '''
                    1: go print
                    2: go print var
                    3: go print args
                    0: quit
                    '''
          print(my_menu)
          choice = get_integer_input()
          if choice == 1:
               go_print()
          elif choice == 2:
               go_print_var()
          elif choice == 3:
               go_print_args("Jill", 0)
          elif choice == 0:
               run = False
               print("Thank you")
          else:
               print("you have not chose a valid option")

menu()
```

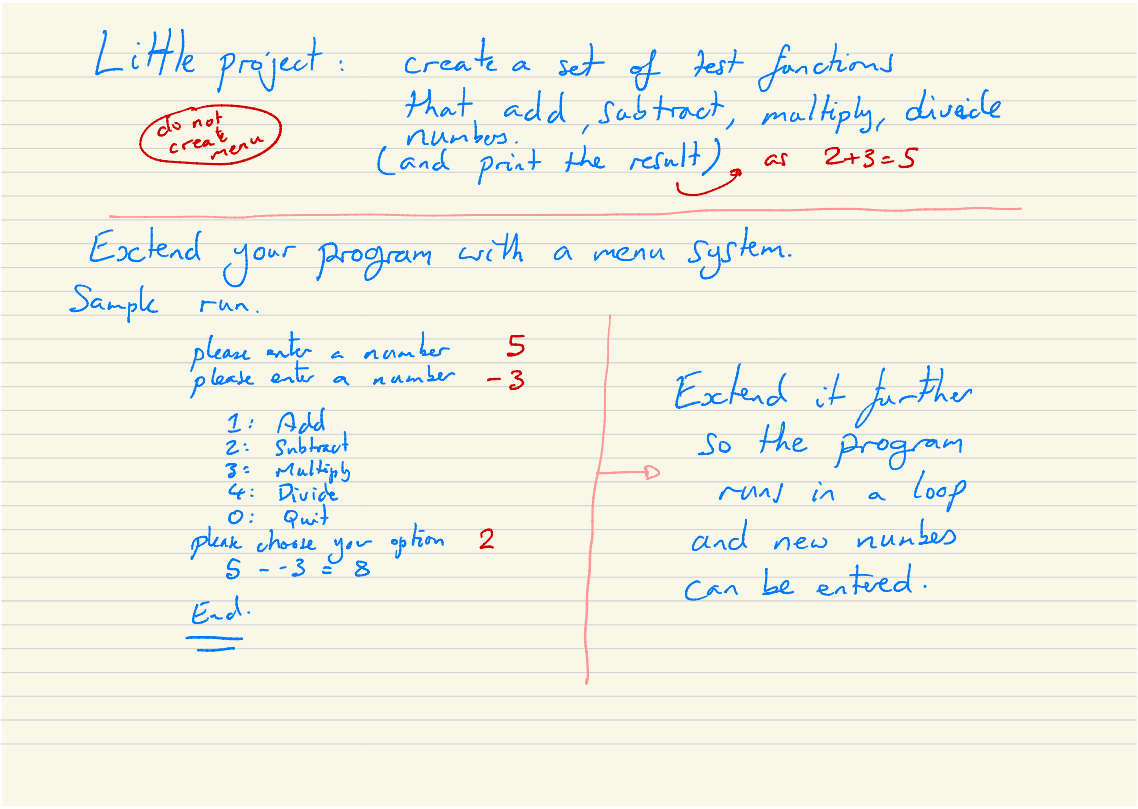
Listing 1: Test program - functions and menu



Simple Plan



Simple Plan as flow diagram



Little start project

3 String Formatting

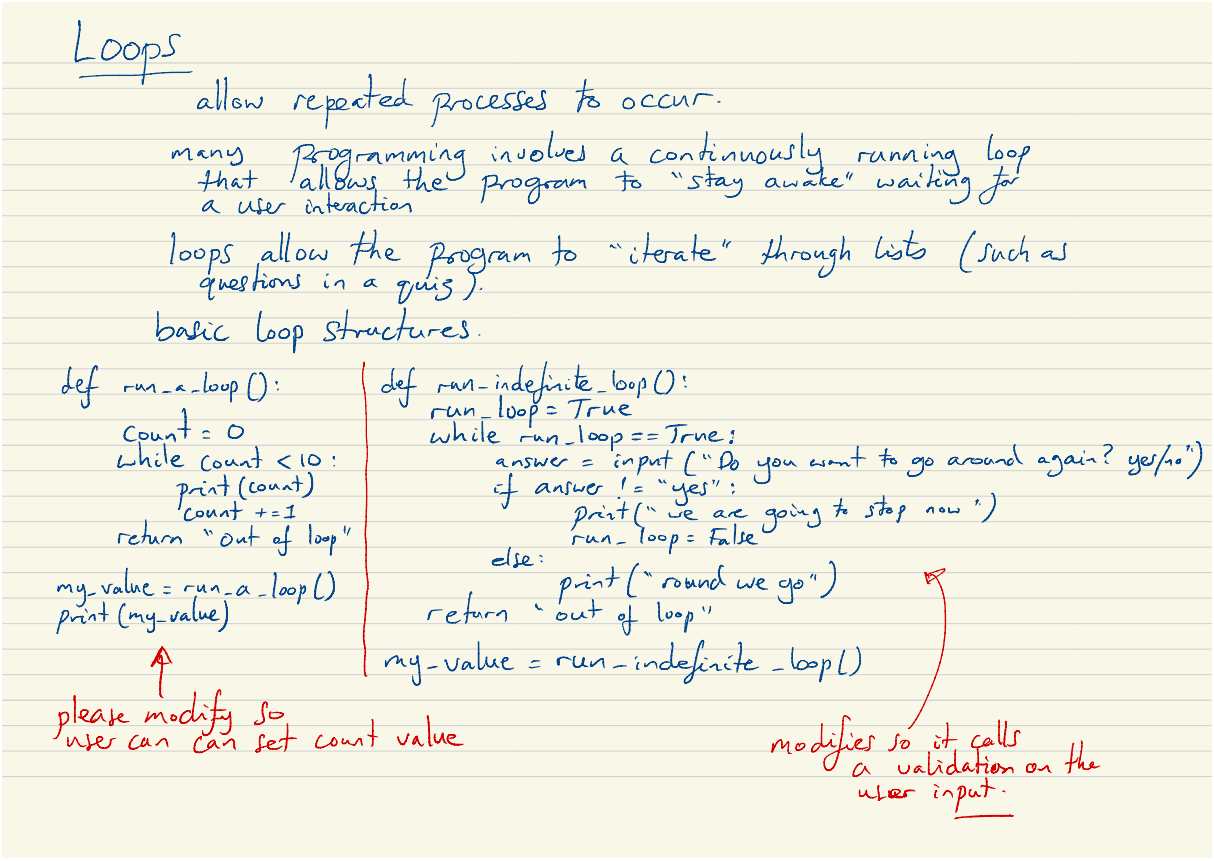
Manging strings for print output is critical:

Please review using:

https://www.w3schools.com/python/python_string_formatting.asp

This means that you actively make variations on the examples given on the page.

4 Loops



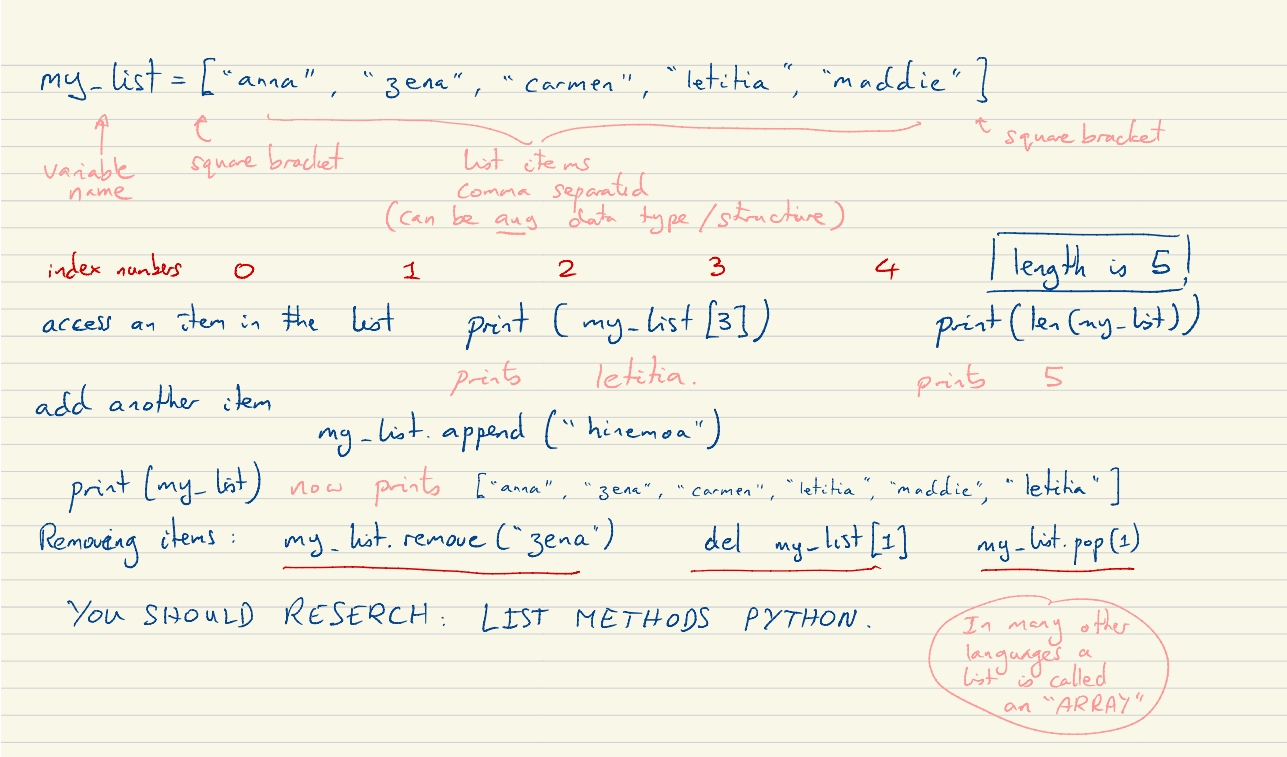

```
def while_loop_counter(count):
    # requires a counter
    c = 0
    # condition is set on the counter value
    while c < count:
        print("hello")
        # counter must be incremented
        c +=1
    print("All done")


def while_loop_quit():
    # set a condition
    run = True
    # check the condition
    while run is True:
        choice = input("Enter Q to quit or any other key to stay where you are: -> ")
        if choice == "Q":
            print("See you later")
            run = False
        else:
            print("I am still here")
```

```
23
24
25 def for_in_range_loop():
26     # this has a built in counter
27     # can be anything but we generally use i or j or k
28     # can add steps
29     for i in range(6,50, 3):
30         print(i)
31
32
33 def double_loop():
34     for i in range(0,10):
35         for j in range(0,10):
36             print(" ({:^2}, {:^2})".format(j,i), end="")
37         print()
38
39
40 def menu():
41     my_menu = '''
42                 1: while loop with counter
43                 2: while loop with quit
44                 3: for in range
45                 4: double loop
46                 0: quit
47                 '''
48     print(my_menu)
49     choice = int(input("choose your option: -> "))
50     if choice is 1:
51         amount = int(input("How many would you like: -> "))
52         while_loop_counter(amount)
53     elif choice is 2:
54         while_loop_quit()
55     elif choice is 3:
56         for_in_range_loop()
57     elif choice is 4:
58         double_loop()
59     elif choice is 5:
60         print("Thank you")
61
62 menu()
```

Listing 2: Test program - functions and menu

5 Lists

Lists are a collection of data items




```
import random

def print_list(L):
    my_string = ""
    for x in L:
        my_string =  my_string + x + " , "
    print(my_string)

def print_with_indexes(L):
    for i in range(0,len(L)):
        output = "{} : {}".format(i, L[i])
        print(output)

def remove_item(L):
    print_list(L)
    choice = input("Who would you like to remove?  ")
    if choice in L:
        L.remove(choice)
        print("{} has been removed".format(choice))
    else:
        print("You choice is not in the list")
    print_list(L)


```

```
25 def add_to_list(L):
26     print_list(L)
27     choice = input("Who would you like to add?  ")
28     L.append(choice)
29     print_list(L)
30 
31 def remove_at_index(L):
32     print_with_indexes(L)
33     choice = int(input("Enter index number to remove:   "))
34     if 0 <= choice < len(L):
35         del L[choice]
36         print_with_indexes(L)
37     else:
38         print("This is not a valid index number")
39 
40 def insert_at_index(L):
41     choice = int(input("Enter index number to insert at:  ->  "))
42     value = input("Enter value to insert: ->  ")
43     L.insert(choice , value)
44     print_with_indexes(L)
45     return L
46 
47 def shuffle_list(L):
48     random.shuffle(L)
49     print_with_indexes(L)
50 
51 
52 def sort_list(L):
53     L.sort()
54     print_with_indexes(L)
55 
56 def find_in_list(L):
57     value = input("Who would you like find? ->  ")
58     if value in L:
59         my_string = "{} is in the list and it is at index #{}".format(value,
    L.index(value))
60     else:
61         my_string = "{} is not in the list".format(value)
62     print(my_string)
63 
64 
65 def main():
66     names = ["Alice", "Belinda", "Chansing", "Debbie", "Eloise", "Floss"]
67     run = True
68     while run:
69         print("-"*50)
70         menu = {
```

```
            "A" : "Print a list horizontally",
            "B" : "Print a list vertically with indexes",
            "C" : "Remove and item from the list",
            "D" : "Add an item",
            "E" : "Remove an item at a specified index",
            "F" : "Insert an item at a given index",
            "G" : "Sort the list",
            "H" : "Find an item in the list",
            "I" : "Shuffle List",
            "Q" : "Quit"
        }
        i = 0
        menu_string = ""
        for x, y in menu.items():
            menu_string += "| {:2} - {:50} ".format(x,y)
            i += 1
            if i != 0 and i%2 == 0:
                menu_string += "\n"
        print(menu_string)

        choice = input("Please enter your choice: -> ")
        print("-" * 50)
        if choice == "A":
            print_list(names)
        elif choice == "B":
            print_with_indexes(names)
        elif choice ==  "C":
            remove_item(names)
        elif choice == "D":
            add_to_list(names)
        elif choice == "E":
            remove_at_index(names)
        elif choice == "F":
            insert_at_index(names)
        elif choice == "G":
            sort_list(names)
        elif choice == "H":
            find_in_list(names)
        elif choice == "I":
            shuffle_list(names)
        elif choice =="Q":
            print("Thanks , you have quit")
            run = False
        else:
            print("Your entry has not been recognised")


```

```
118 main()
```

Listing 3: Lists

5.1 Issues with lists

- The standard error with lists is: **IndexError: list index out of range**

5.2 What we should be able to do with lists:

- Create one and access items in the list using index notation.
- Find how many items are in the list
- Loop through the list and print out items (have more than one way of doing)
- Understand that there are basic **methods** associated with a list. These mean that we can add, remove, update elements in a list.

5.3 Activities

- Create a function that takes a list as an argument, prints out the list (horizontally), and returns the length of the list
- Create a program that asks for the user to enter 5 numbers, it then prints out the sum, average and product of the numbers (each part is a function).
- Create a program that asks for the user to enter 5 numbers, it then prints a menu asking giving users and option to get the sum, product, average of the numbers

6 Validating User Input

- Create a function to get input of a single letter from a user.
- Create a function to get an integer from a user
- Create a function to get a string between 2 and 5 letters
- Create a function to get an integer from a user with boundaries that might change

7 Looping through lists

8 Lists with more than one dimension

Sometimes we wish to store a "package" of information.

For example , instead of Alice, Bob, Cathy, we might want Alice , Brown, 31 ; Bob Blond 19; Cathy, Black, 21

So we have person, hair colour and age.

```
1
2  def loop_on_lists(L):
3      print("-" * 30)
4      c = 0
5      for x in L:
6          if c == 0:
7              my_string = "{:2}  {:5} , {:5} , {:^5}\n" \
8                          "{:2}  {:5}  {:5}  {:5}" \
9                          " ".format("", x[0], x[1], x[2],"",  "-"*5 ,  "-"*5,
       "-"*5)
10         else:
11             my_string = "{:2}: {:5} , {:5} , {:^5}".format(c, x[0], x[1], x
     [2])
12         print(my_string)
13         c+=1
14
15 def update_hair(L):
16     loop_on_lists(L)
17     my_index = int(input("Choose option number for hair to update: -> "))
18     new_colour = input("Enter new hair colour: -> ")
19     L[my_index][1]= new_colour
20     my_string = "The hair colour for {} has been updated to {}".format(L[
     my_index][0],L[my_index][1])
21     print(my_string)
22     return None
23
24
25 def update_age(L):
26     loop_on_lists(L)
27     my_index = int(input("Choose option number for age to update: -> "))
28     new_age = int(input("Enter new age: -> "))
29     L[my_index][2]= new_age
30     my_string = "The age for {} has been updated to {}".format(L[my_index
     ][0],L[my_index][2])
31     print(my_string)
32     return None
```

```
33
34 def update(L):
35     M = L[0]
36     for i in range(1, len(M)):
37         print("{} : {}".format(i, M[i]))
38     choice = int(input("Please enter your option letter: -> "))
39     if choice is 1:
40         update_hair(L)
41     elif choice is 2:
42         update_age(L)
43     else:
44         print("Unrecognised Entry")
45
46
47 def search_lists(L):
48     search_item = input("Please enter your search item: -> ")
49     try:
50         s= int(search_item)
51     except:
52         s = search_item
53     row = 0
54     for y in L:
55         if s in y:
56             item_index = y.index(s)
57             print("{} has been found at Row {} index {}".format( y[
    item_index],row,  item_index ) )
58         row += 1
59
60 def create_new_entry(L):
61     name = input("Please enter name: -> ")
62     hair_colour = input("Please enter hair colour: -> ")
63     age = int(input("Please enter hair colour: -> "))
64     L.append( [name, hair_colour , age])
65     loop_on_lists(L)
66
67 def sort_on_index(L):
68     loop_on_lists(L)
69     for i in range(0, len(L[0])):
70         print( "{} : {}".format(i, L[0][i]))
71     sort_index = int(input(" Please choose index to sort on: -> "))
72     M = L.pop(0)
73     L=sorted(L, key = lambda x: x[sort_index])
74     L.insert(0,M)
75     loop_on_lists(L)
76     return L
77
78 def menu(group_of_lists):
```

```
79
80      my_menu = [
81          ("R", "Review"),
82          ("U", "Update"),
83          ("S", "Search"),
84          ("A", "Add new name"),
85          ("O", "Order"),
86          ("Q", "Quit")
87      ]
88      print("-" * 30)
89      for x in my_menu:
90          print( "{} : {}".format(x[0],x[1]))
91      print("-"*30)
92      choice = input("Please enter your option letter: -> ")
93      if choice is "R":
94          loop_on_lists(group_of_lists)
95      elif choice is "U":
96          update(group_of_lists)
97      elif choice is "S":
98          search_lists(group_of_lists)
99      elif choice is "A":
100         create_new_entry(group_of_lists)
101     elif choice is "O":
102         group_of_lists = sort_on_index(group_of_lists)
103     elif choice is "Q":
104         print("Thank you")
105         return "Q"
106     else:
107         print("Unrecognised Entry")
108     return group_of_lists
109
110
111 def main():
112     group_of_lists_header = ["Name", "Hair", "Age"]
113     group_of_lists =[
114         group_of_lists_header,
115         ["Alice", "Blond", 12],
116         ["Cathy", "Blond", 15],
117         ["Dan", "Black", 10],
118         ["Bob", "Brown", 13]
119     ]
120     run = True
121     while run:
122         status = menu(group_of_lists)
123         if status == "Q":
124             run = False
125         else:
```

```
126             group_of_lists = status
127 main()
```

Listing 4: Lists

This links to a number of code examples

```
names = ["Alice","Belinda", "Chansing", "Debbie", "Eloise", "Floss"]
# view the list
# view the list with a counter
# find how many elements are in the list
# remove an elements from a list
# access an element in a list


# loop through list
for x in names:
    print(x, end="")
print()
# loop using a counter
for i in range(0, len(names)):
    print(i)
names.remove("Alice")
print(names)
names.append("Alice")
print(names)
names.sort()
print(names)
del names[3]
print(names)
print(names.pop(0))
print(names)
```

Listing 5: Lists

```
/usr/local/bin/python3.7 /Users/Paul/Documents/Python_projects/
   FruitBowlGitHub/lists.py
AliceBelindaChansingDebbieEloiseFloss
0
1
2
3
4
5
['Belinda', 'Chansing', 'Debbie', 'Eloise', 'Floss']
['Belinda', 'Chansing', 'Debbie', 'Eloise', 'Floss', 'Alice']
['Alice', 'Belinda', 'Chansing', 'Debbie', 'Eloise', 'Floss']
['Alice', 'Belinda', 'Chansing', 'Eloise', 'Floss']
Alice
['Belinda', 'Chansing', 'Eloise', 'Floss']
```

Listing 6: Lists

```
def get_numbers():
    num_list = []
```

```
    counter = 0
    while counter < 3:
        num = int(input("Please enter a number "))
        num_list.append(num)
        counter += 1
    num_list.sort()
    print_list(num_list)
    return num_list


def print_list(l):
    my_string = ""
    counter = 0
    for x in l:
        if counter == len(l) - 1:
            my_string = my_string + str(x)
        else:
            my_string = my_string + str(x) + " , "

        counter += 1
    print(my_string)


def add_nums(l):
    _sum = 0
    for x in l:
        _sum += x
    return _sum


def product_nums(l):
    product = 1
    for x in l:
        product = product * x
    return product


def average_nums(l):
    _sum = add_nums(l)
    average = _sum / len(l)
    return average


def print_output(m, v):
    print(30 * "-")
    print("Your {} is :  {}".format(m, v))
    print(30 * "-")
```

```
50
51
52  def run_menu():
53      menu = True
54      menu_list = ["Enter set", "Get average", "Get sum", "Get product"]
55      set = []
56      while menu is True:
57          for i in range(0, len(menu_list)):
58              print("{:<5}{:^10} {:20}".format(i, "--", menu_list[i]))
59          user_choice = input("Please choose your option:  ")
60          if user_choice == "q":
61              return
62          else:
63              user_choice = int(user_choice)
64          if user_choice != 0 and len(set) == 0:
65              print("You have an empty set")
66              continue
67          if user_choice == 0:
68              set = get_numbers()
69          elif user_choice == 1:
70              average = average_nums(set)
71              print_output("average", average)
72          elif user_choice == 2:
73              sum = add_nums(set)
74              print_output("sum", sum)
75          elif user_choice == 3:
76              product = product_nums(set)
77              print_output("product", product)
78          else:
79              print("Unrecognised entry")
80
81
82  if __name__ == "__main__":
83      run_menu()
```

Listing 7: Menu and Functions