

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.

* use of collections (dictionaries or lists).

What to look for:

* 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.?)

* how did you
get input
from the
user?

* use of loops

how many loops did you make?

why use them?

how do you control a loop.

2 Functions

Functions

A function is a block of code that can be "called" at any time.
Assume that everything you do will be using a function.

```
def my_function():
    function code goes here
    return some value
```

name
arguments (can be empty)
return statement (Sends back a value to the place where the function is called)
If there is nothing to return, we return None.

adding numbers

```
def add_two(a,b):
    my_sum = a+b
    return my_sum
```

arguments.

def main():
 my_value = add_two(4,5)
 print(my_value)

main()

functions do not have to have a return statement, but it is good to get into the habit of using them.
a return automatically terminates the function.

- 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.

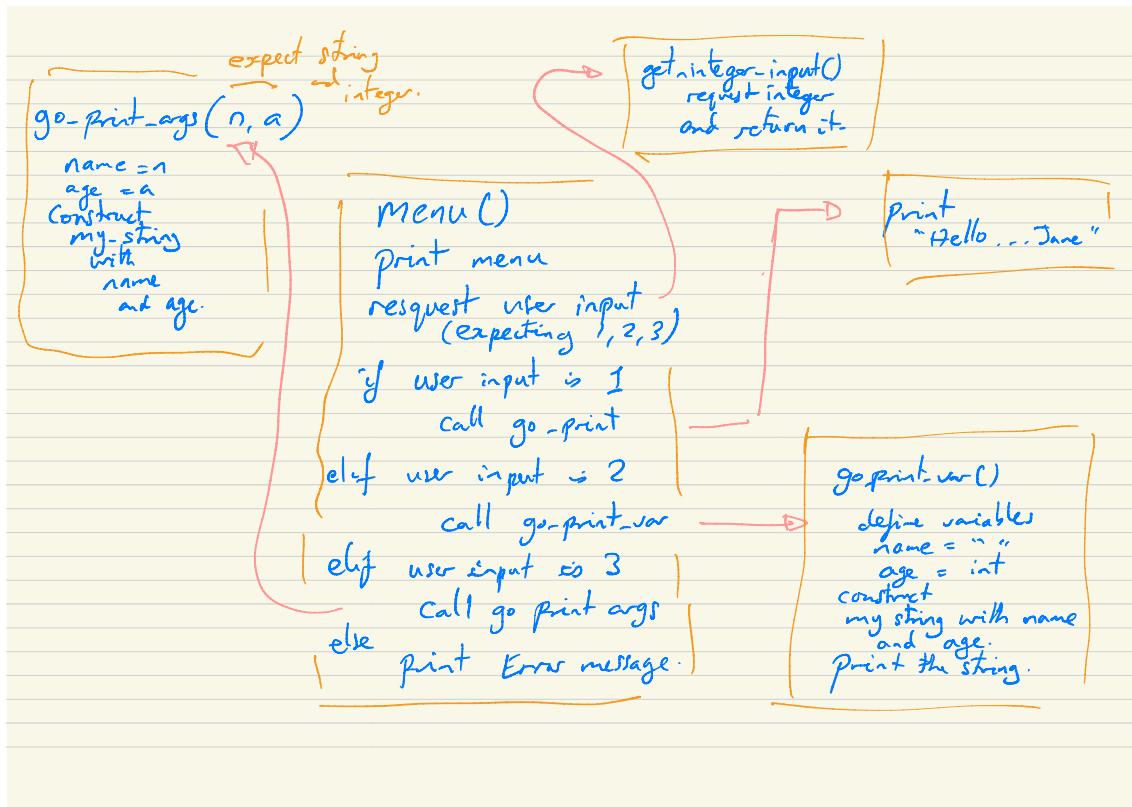
```
1
2
3 def go_print():
4     print("Hello my name is Jane")
5
6 def go_print_var():
7     name = "Bob"
8     age = 10
9     # put them together in a string
10    my_string = "My name is {} \nI am {} years old".format(name, age)
11    print(my_string)
12
13 def go_print_args(n,a):
14     name = n
15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

```

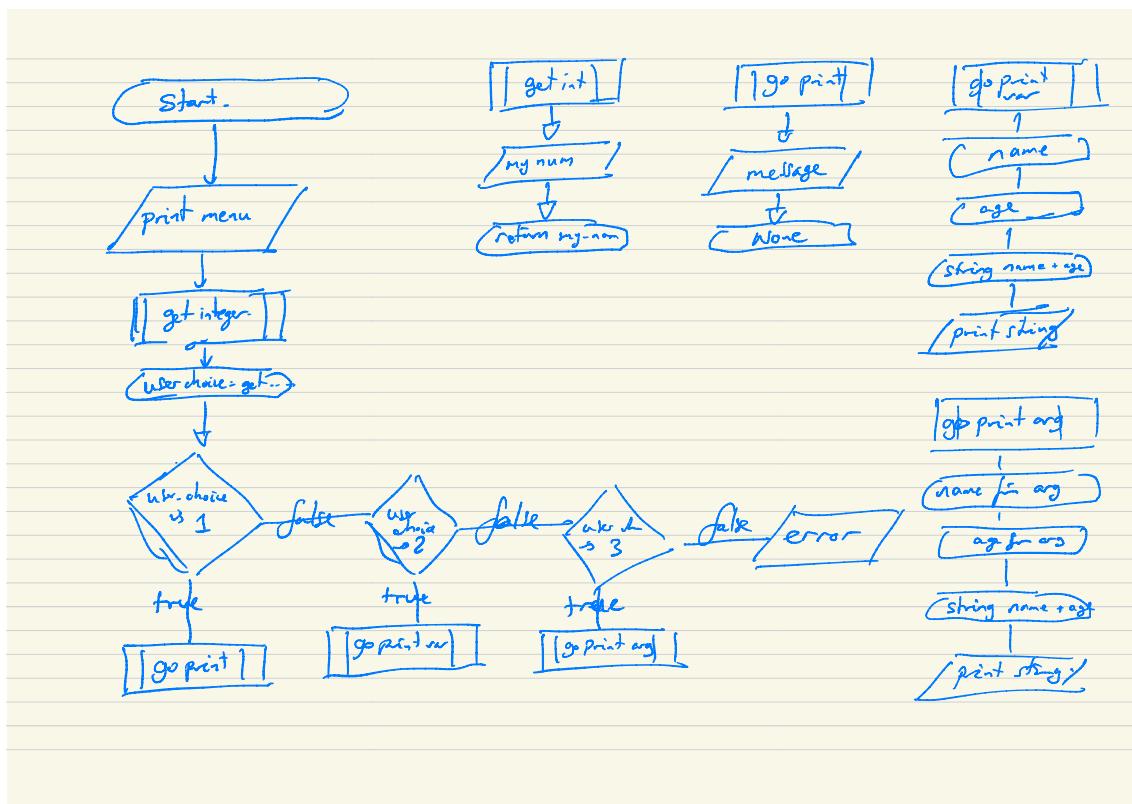
19 def get_integer_input():
20     my_number = int(input("please enter number: -> "))
21     return my_number
22
23 def menu():
24     run = True
25     while run is True:
26         my_menu = '''
27             1: go print
28             2: go print var
29             3: go print args
30             0: quit
31             ''',
32         print(my_menu)
33         choice = get_integer_input()
34         if choice == 1:
35             go_print()
36         elif choice == 2:
37             go_print_var()
38         elif choice == 3:
39             go_print_args("Jill", 0)
40         elif choice == 0:
41             run = False
42             print("Thank you")
43         else:
44             print("you have not chose a valid option")
45
46 menu()

```

Listing 1: Test program - functions and menu



Simple Plan



Simple Plan as flow diagram

Little project: create a set of test functions
that add, subtract, multiply, divide
numbers.
(and print the result) as $2+3=5$

do not create menu

Extend your program with a menu system.

Sample run.

```
please enter a number 5
please enter a number -3
1: Add
2: Subtract
3: Multiply
4: Divide
0: Quit
please choose your option 2
5 - -3 = 8
End.
```

Extend it further
so the program
runs in a loop
and new numbers
can be entered.

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

Loops

allow repeated processes to occur.

many programming involves a continuously running loop that allows the program to "stay awake" waiting for a user interaction

loops allow the program to "iterate" through lists (such as questions in a quiz).

basic loop structures.

```
def run_a_loop():
    count = 0
    while count < 10:
        print(count)
        count += 1
    return "out of loop"
my_value = run_a_loop()
print(my_value)
```

↑
please modify so user can set count value

```
def run_indefinite_loop():
    run_loop = True
    while run_loop == True:
        answer = input("Do you want to go around again? yes/no")
        if answer != "yes":
            print("we are going to stop now")
            run_loop = False
        else:
            print("round we go")
    return "out of loop"
my_value = run_indefinite_loop()
```

↑
modifies so it calls a validation on the user input.

```
1 def while_loop_counter(count):
2     # requires a counter
3     c = 0
4     # condition is set on the counter value
5     while c < count:
6         print("hello")
7         # counter must be incremented
8         c += 1
9     print("All done")
10
11
12 def while_loop_quit():
13     # set a condition
14     run = True
15     # check the condition
16     while run is True:
17         choice = input("Enter Q to quit or any other key to stay where you
18 are: -> ")
19         if choice == "Q":
20             print("See you later")
21             run = False
22         else:
23             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
49     print(my_menu)
50     choice = int(input("choose your option: -> "))
51     if choice is 1:
52         amount = int(input("How many would you like: -> "))
53         while_loop_counter(amount)
54     elif choice is 2:
55         while_loop_quit()
56     elif choice is 3:
57         for_in_range_loop()
58     elif choice is 4:
59         double_loop()
60     elif choice is 5:
61         print("Thank you")
62 menu()

```

Listing 2: Test program - functions and menu

5 Lists

Lists are a collection of data items

my_list = ["anna", "zena", "carmen", "letitia", "maddie"]

variable name square bracket list items
index numbers 0 1 2 3 4 length is 5
access an item in the list print(my_list[3]) prints letitia.
add another item my_list.append("hinehoa")
print(my_list) now prints ["anna", "zena", "carmen", "letitia", "maddie", "letitia"]
Removing items: my_list.remove("zena") del my_list[1] my_list.pop(1)

You SHOULD RESEARCH: LIST METHODS PYTHON.

In many other languages a list is called an "ARRAY"

```
1 import random
2
3 def print_list(L):
4     my_string = ""
5     for x in L:
6         my_string = my_string + x + ", "
7     print(my_string)
8
9 def print_with_indexes(L):
10    for i in range(0, len(L)):
11        output = "{} : {}".format(i, L[i])
12        print(output)
13
14 def remove_item(L):
15    print_list(L)
16    choice = input("Who would you like to remove? ")
17    if choice in L:
18        L.remove(choice)
19        print("{} has been removed".format(choice))
20    else:
21        print("Your choice is not in the list")
22    print_list(L)
23
24
```

```

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,
60             L.index(value))
61     else:
62         my_string = "{} is not in the list".format(value)
63     print(my_string)
64
65 def main():
66     names = ["Alice", "Belinda", "Chansing", "Debbie", "Eloise", "Floss"]
67     run = True
68     while run:
69         print("-"*50)
70         menu = {

```

```

71         "A" : "Print a list horizontally",
72         "B" : "Print a list vertically with indexes",
73         "C" : "Remove and item from the list",
74         "D" : "Add an item",
75         "E" : "Remove an item at a specified index",
76         "F" : "Insert an item at a given index",
77         "G" : "Sort the list",
78         "H" : "Find an item in the list",
79         "I" : "Shuffle List",
80         "Q" : "Quit"
81     }
82     i = 0
83     menu_string = ""
84     for x, y in menu.items():
85         menu_string += "| {:2} - {:50} ".format(x,y)
86         i += 1
87         if i != 0 and i%2 == 0:
88             menu_string += "\n"
89     print(menu_string)
90
91     choice = input("Please enter your choice: -> ")
92     print("-" * 50)
93     if choice == "A":
94         print_list(names)
95     elif choice == "B":
96         print_with_indexes(names)
97     elif choice == "C":
98         remove_item(names)
99     elif choice == "D":
100        add_to_list(names)
101    elif choice == "E":
102        remove_at_index(names)
103    elif choice == "F":
104        insert_at_index(names)
105    elif choice == "G":
106        sort_list(names)
107    elif choice == "H":
108        find_in_list(names)
109    elif choice == "I":
110        shuffle_list(names)
111    elif choice == "Q":
112        print("Thanks , you have quit")
113        run = False
114    else:
115        print("Your entry has not been recognised")
116
117

```

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,
10                  " "*5)
11         else:
12             my_string = "{:2}: {:5} , {:5} , {:^5}".format(c, x[0], x[1], x
13 [2])
14         print(my_string)
15         c+=1
16
17 def update_hair(L):
18     loop_on_lists(L)
19     my_index = int(input("Choose option number for hair to update: -> "))
20     new_colour = input("Enter new hair colour: -> ")
21     L[my_index][1] = new_colour
22     my_string = "The hair colour for {} has been updated to {}".format(L[
23     my_index][0], L[my_index][1])
24     print(my_string)
25     return None
26
27 def update_age(L):
28     loop_on_lists(L)
29     my_index = int(input("Choose option number for age to update: -> "))
30     new_age = int(input("Enter new age: -> "))
31     L[my_index][2] = new_age
32     my_string = "The age for {} has been updated to {}".format(L[my_index]
33     ][0], L[my_index][2])
34     print(my_string)
35     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[
58             item_index],row, item_index) )
59             row += 1
60
61
62 def create_new_entry(L):
63     name = input("Please enter name: -> ")
64     hair_colour = input("Please enter hair colour: -> ")
65     age = int(input("Please enter hair colour: -> "))
66     L.append([name, hair_colour, age])
67     loop_on_lists(L)
68
69
70 def sort_on_index(L):
71     loop_on_lists(L)
72     for i in range(0, len(L[0])):
73         print(" {} : {}".format(i, L[0][i]))
74     sort_index = int(input(" Please choose index to sort on: -> "))
75     M = L.pop(0)
76     L=sorted(L, key = lambda x: x[sort_index])
77     L.insert(0,M)
78     loop_on_lists(L)
79     return L
80
81
82 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

```

1 names = ["Alice", "Belinda", "Chansing", "Debbie", "Eloise", "Floss"]
2 # view the list
3 # view the list with a counter
4 # find how many elements are in the list
5 # remove an elements from a list
6 # access an element in a list
7
8
9 # loop through list
10 for x in names:
11     print(x, end="")
12 print()
13 # loop using a counter
14 for i in range(0, len(names)):
15     print(i)
16 names.remove("Alice")
17 print(names)
18 names.append("Alice")
19 print(names)
20 names.sort()
21 print(names)
22 del names[3]
23 print(names)
24 print(names.pop(0))
25 print(names)

```

Listing 5: Lists

```

1 /usr/local/bin/python3.7 /Users/Paul/Documents/Python_projects/
    FruitBowlGitHub/lists.py
2 AliceBelindaChansingDebbieEloiseFloss
3 0
4 1
5 2
6 3
7 4
8 5
9 ['Belinda', 'Chansing', 'Debbie', 'Eloise', 'Floss']
10 ['Belinda', 'Chansing', 'Debbie', 'Eloise', 'Floss', 'Alice']
11 ['Alice', 'Belinda', 'Chansing', 'Debbie', 'Eloise', 'Floss']
12 ['Alice', 'Belinda', 'Chansing', 'Eloise', 'Floss']
13 Alice
14 ['Belinda', 'Chansing', 'Eloise', 'Floss']

```

Listing 6: Lists

```

1 def get_numbers():
2     num_list = []

```

```

3    counter = 0
4
5    while counter < 3:
6        num = int(input("Please enter a number "))
7        num_list.append(num)
8        counter += 1
9
10   num_list.sort()
11   print_list(num_list)
12
13 def print_list(l):
14     my_string = ""
15     counter = 0
16
17     for x in l:
18         if counter == len(l) - 1:
19             my_string = my_string + str(x)
20         else:
21             my_string = my_string + str(x) + " , "
22
23     counter += 1
24
25
26 def add_nums(l):
27     _sum = 0
28
29     for x in l:
30         _sum += x
31
32
33 def product_nums(l):
34     product = 1
35
36     for x in l:
37         product = product * x
38
39
40 def average_nums(l):
41     _sum = add_nums(l)
42     average = _sum / len(l)
43
44
45
46 def print_output(m, v):
47     print(30 * "-")
48     print("Your {} is : {}".format(m, v))
49     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