

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, use 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.

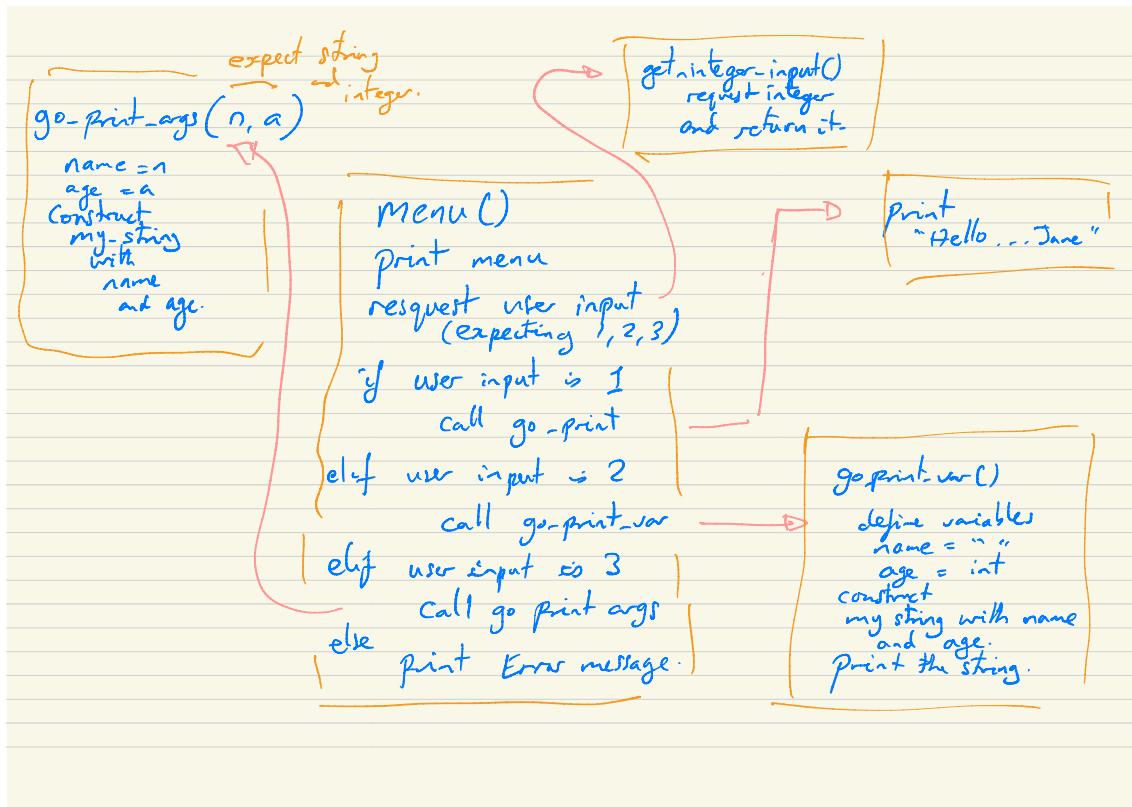
```
1
2
3 def go_print():
4     print("Hello my name is Jane")
5
6 def go_print_var():
7     name = "Bob"
8     age = 10
9     # put them together in a string
10    my_string = "My name is {} \nI am {} years old".format(name, age)
11    print(my_string)
12
13 def go_print_args(n,a):
14     name = n
15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

```

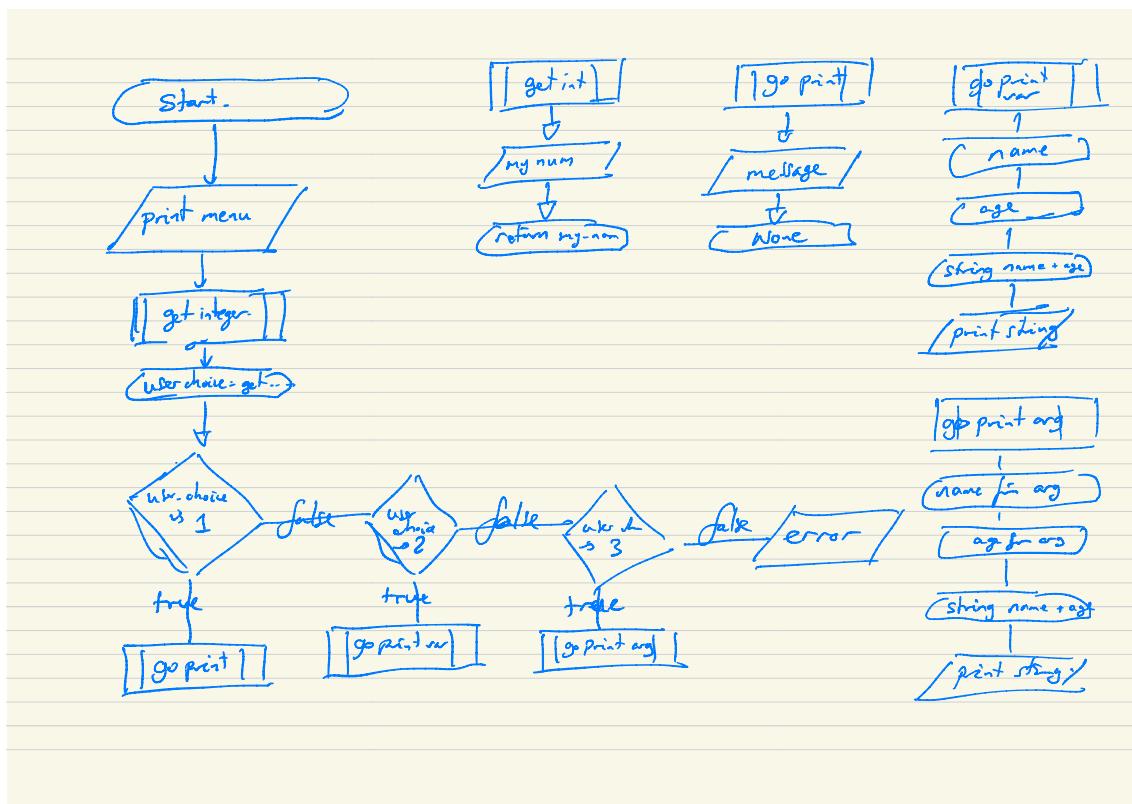
19 def get_integer_input():
20     my_number = int(input("please enter number: -> "))
21     return my_number
22
23 def menu():
24     run = True
25     while run is True:
26         my_menu = '''
27             1: go print
28             2: go print var
29             3: go print args
30             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

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)
```

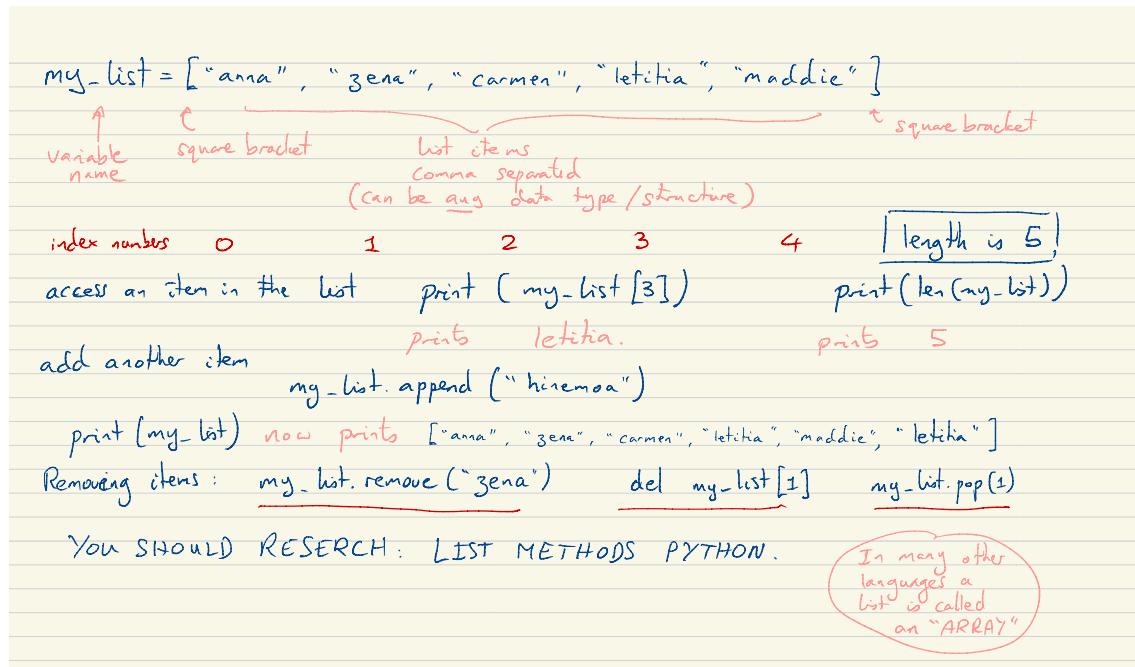
↑
please modify so
user can set count value

```
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.

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])
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"

This links to a number of code examples

5.1 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.

5.2 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

```

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 2: 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 3: Lists

```

1 names = ["Alice", "Belinda", "Chansing", "Debbie", "Eloise", "Floss"]
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
15 def remove_item(L):
16     print_list(L)
17     choice = input("Who would you like to remove? ")
18     if choice in L:
19         L.remove(choice)
20         print("{} has been removed".format(choice))
21     else:
22         print("Your choice is not in the list")
23     print_list(L)
24
25
26 def add_to_list(L):
27     print_list(L)
28     choice = input("Who would you like to add? ")
29     L.append(choice)
30     print_list(L)
31
32 def remove_at_index(L):
33     print_with_indexes(L)
34     choice = int(input("Enter index number to remove: "))
35     if 0 <= choice < len(L):
36         del L[choice]
37         print_with_indexes(L)
38     else:
39         print("This is not a valid index number")
40
41 def sort_list(L):
42     L.sort()
43     print_with_indexes(L)
44
45
46
47 #print_list(names)
48 remove_item(names)
49 #print_with_indexes(names)

```

```
50 #remove_at_index(names)
51 #add_to_list(names)
52 #sort_list(names)
```

Listing 4: Lists

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

```

43     return average
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 5: Menu and Functions