

# 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():
|     |     |     name
|     |     |     arguments (can be empty)
|     |     |     function
|     |     |     code goes here
|     |     return some value    ← return statement (sends back a value
|     |     |     to the place where the function
|     |     |     is called)
|     |     my_value = my_function()    If there is nothing to return, use
|     |     |     holds the returned value.    return None.
```

adding numbers.

```
def add_two(a,b)    ← arguments.
    my_sum = a+b
    return my_sum
```

```
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.

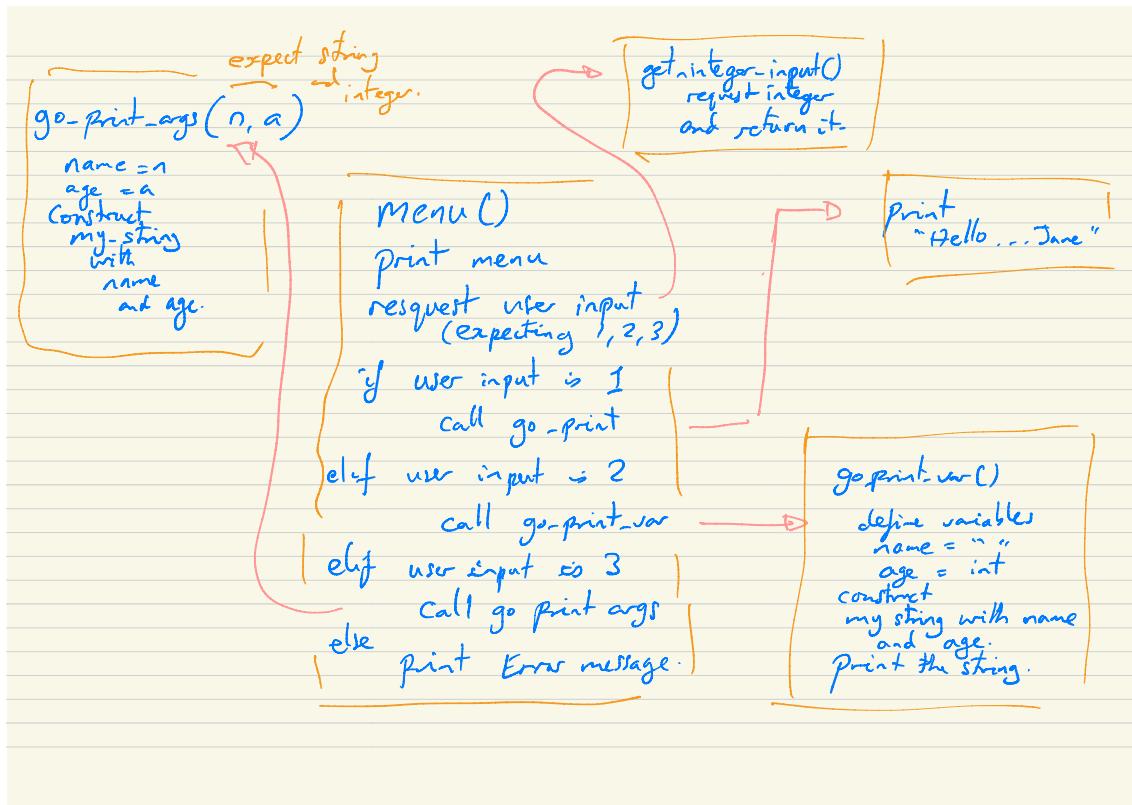
```
1
2
3 def go_print():
4     print("Hello my name is Jane")
5
6 def go_print_var():
7     name = "Bob"
8     age = 10
9     # put them together in a string
10    my_string = "My name is {} \nI am {} years old".format(name, age)
11    print(my_string)
12
13 def go_print_args(n,a):
14     name = n
15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

```

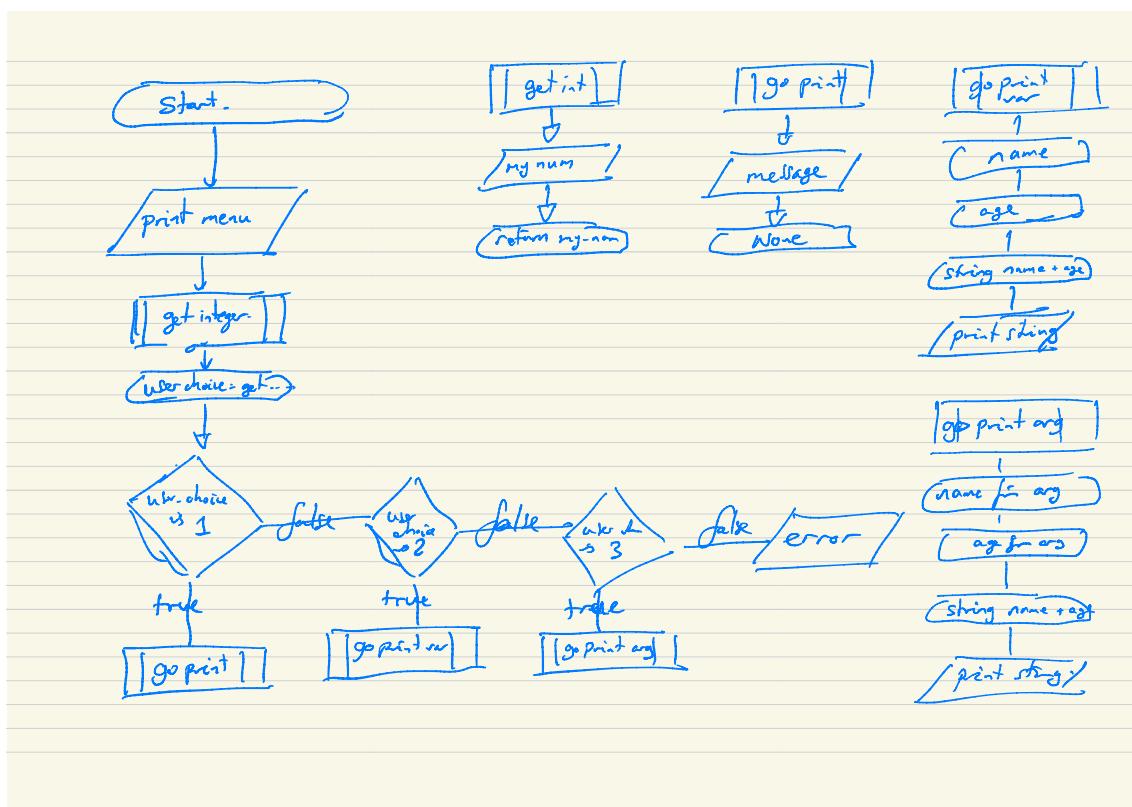
18
19 def get_integer_input():
20     my_number = int(input("please enter number: -> "))
21     return my_number
22
23 def menu():
24     run = True
25     while run is True:
26         my_menu = '''
27             1: go print
28             2: go print var
29             3: go print args
30             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](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_indefinite():
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
```

```

21     else:
22         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_indefinite()
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
63 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
4 def get_string(m):
5     my_string = input(m)
6     return my_string
7
8
9 def get_integer(m):
10    my_integer = int(input(m))
11    return my_integer
12
13
14 def print_list(L):
15    my_string = ""
16    for x in L:
17        my_string = my_string + x + " , "
18    print(my_string)
19
20
21 def print_with_indexes(L):
22    for i in range(0, len(L)):
23        output = "{} : {}".format(i, L[i])
24        print(output)
```

```

25
26
27 def change_value(L):
28     print_with_indexes(L)
29     index_num = get_integer("Please choose the index value you want to
30     change: -> ")
31     new_value = get_string("Please enter the new name: -> ")
32     old_value = L[index_num]
33     L[index_num] = new_value
34     output = "The old value of {} has now been changed to {}".format(
35         old_value, L[index_num])
36     print(output)

37 def remove_item(L):
38     print_list(L)
39     choice = input("Who would you like to remove? ")
40     if choice in L:
41         L.remove(choice)
42         print("{} has been removed".format(choice))
43     else:
44         print("Your choice is not in the list")

45
46
47 def add_to_list(L):
48     print_list(L)
49     choice = input("Who would you like to add? ")
50     L.append(choice)
51     print("{} has been added".format(choice))

52
53
54 def remove_at_index(L):
55     print_with_indexes(L)
56     choice = int(input("Enter index number to remove: "))
57     if 0 <= choice < len(L):
58         del L[choice]
59     else:
60         print("This is not a valid index number")

61
62 def insert_at_index(L):
63     choice = int(input("Enter index number to insert at: -> "))
64     value = input("Enter value to insert: -> ")
65     L.insert(choice, value)
66     print_with_indexes(L)
67     return L
68
69

```

```

70 def shuffle_list(L):
71     random.shuffle(L)
72
73
74 def sort_list(L):
75     L.sort()
76
77
78 def find_in_list(L):
79     value = input("Who would you like find? -> ")
80     if value in L:
81         my_string = "{} is in the list and it is at index {}".format(value,
82                           L.index(value))
83     else:
84         my_string = "{} is not in the list".format(value)
85     print(my_string)
86
87 def main():
88     names = ["Alice", "Belinda", "Chansing", "Debbie", "Eloise", "Floss"]
89     run = True
90     while run:
91         print("-"*100)
92         menu = {
93             "A" : "Print a list horizontally",
94             "B" : "Print a list vertically with indexes",
95             "C" : "Remove an item from the list",
96             "D" : "Add an item",
97             "E" : "Remove an item at a specified index",
98             "F" : "Insert an item at a given index",
99             "G" : "Sort the list",
100            "H" : "Find an item in the list",
101            "I" : "Shuffle List",
102            "J" : "Change a value",
103            "Q" : "Quit"
104        }
105        i = 0
106        menu_string = ""
107        for x, y in menu.items():
108            menu_string += "| {:2} - {:50} ".format(x,y)
109            i += 1
110            if i != 0 and i%2 == 0:
111                menu_string += "\n"
112        print(menu_string)
113
114        choice = input("Please enter your choice: -> ")
115        print("-" * 100)

```

```

116     if choice == "A":
117         print_list(names)
118     elif choice == "B":
119         print_with_indexes(names)
120     elif choice == "C":
121         remove_item(names)
122     elif choice == "D":
123         add_to_list(names)
124     elif choice == "E":
125         remove_at_index(names)
126     elif choice == "F":
127         insert_at_index(names)
128     elif choice == "G":
129         sort_list(names)
130     elif choice == "H":
131         find_in_list(names)
132     elif choice == "I":
133         shuffle_list(names)
134     elif choice == "J":
135         change_value(names)
136     elif choice == "Q":
137         print("Thanks , you have quit")
138         run = False
139     else:
140         print("Your entry has not been recognised")
141
142
143 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 def get_integer(message):
2     my_integer = int(input(message))
3     return my_integer
4
5
6 def get_string(m):
7     my_string = input(m)
8     return my_string
9
10
11 def loop_on_lists(L):
12     print("-" * 30)
13     c = 0
14     for x in L:
```

```

15     my_string = "{:2}: {:5} , {:5} , {:^5}" .format(c, x[0], x[1], x[2])
16     print(my_string)
17     c += 1
18     print("-" * 30)
19     return None
20
21
22 def update_name(L):
23     loop_on_lists(L)
24     my_index = get_integer("Choose option number for name to update: -> ")
25     new_name = get_string("Enter new name: -> ")
26     old_name = L[my_index][0]
27     L[my_index][0] = new_name
28     my_string = "The name for {} has been updated to {}".format(old_name,
29     new_name)
30     print(my_string)
31     return None
32
33 def update_hair(L):
34     loop_on_lists(L)
35     my_index = get_integer("Choose option number for hair to update: -> ")
36     new_colour = get_string("Enter new hair colour: -> ")
37     L[my_index][1] = new_colour
38     my_string = "The hair colour for {} has been updated to {}".format(L[
39     my_index][0], L[my_index][1])
40     print(my_string)
41     return None
42
43 def update_age(L):
44     loop_on_lists(L)
45     my_index = get_integer("Choose option number for age to update: -> ")
46     new_age = get_integer("Enter new age: -> ")
47     L[my_index][2] = new_age
48     my_string = "The age for {} has been updated to {}".format(L[my_index
49     ][0], L[my_index][2])
50     print(my_string)
51     return None
52
53 def update(main_list):
54     update_menu = [
55         ("N", "Name"),
56         ("H", "Hair"),
57         ("A", "Age"),
58         ("B", "Back")

```

```

59     ]
60
61     for x in update_menu:
62         print("{} : {}".format(x[0], x[1]))
63
64     choice = get_string("Please enter your option letter: -> ")
65
66     if choice is "N":
67         update_name(main_list)
68
69     elif choice is "H":
70         update_hair(main_list)
71
72     elif choice is "A":
73         update_age(main_list)
74
75     elif choice is "B":
76         print("Returning to main menu")
77
78     return None
79
80 else:
81     print("Unrecognised Entry")
82
83 return None
84
85
86
87 def search_lists(L):
88
89     search_item = get_string("Please enter your search item: -> ")
90
91     try:
92
93         s = int(search_item)
94
95     except ValueError:
96
97         s = search_item
98
99     row = 0
100
101    for y in L:
102
103        if s in y:
104
105            item_index = y.index(s)
106
107            print("{} has been found at Row {} index {}".format( y[item_index],row, item_index ) )
108
109        row += 1
110
111    return None
112
113
114
115 def create_new_entry(L):
116
117     name = get_string("Please enter name: -> ")
118
119     hair_colour = get_string("Please enter hair colour: -> ")
120
121     age = get_integer("Please enter age: -> ")
122
123     L.append([name, hair_colour, age])
124
125     return None
126
127
128
129 def sort_on_index(L, H):
130
131     for i in range(0, len(H)):
132
133         print("{} : {}".format(i, H[i]))
134
135     sort_index = get_integer(" Please choose index to sort on: -> ")
136
137     L = sorted(L, key=lambda x: x[sort_index])

```

```

105     return L
106
107
108 def menu():
109     group_of_lists_header = ["Name", "Hair", "Age"]
110     group_of_lists = [
111         ["Alice", "Blond", 12],
112         ["Cathy", "Blond", 15],
113         ["Dan", "Black", 10],
114         ["Bob", "Brown", 13],
115         ["Joe", "Green", 13]
116     ]
117
118     my_menu = [
119         ("R", "Review"),
120         ("U", "Update"),
121         ("S", "Search"),
122         ("A", "Add new name"),
123         ("O", "Order"),
124         ("Q", "Quit")
125     ]
126
127     run = True
128     while run is True:
129         for x in my_menu:
130             print("{} : {}".format(x[0], x[1]))
131         print("-"*30)
132         choice = get_string("Please enter your option letter: -> ")
133         if choice is "R":
134             loop_on_lists(group_of_lists)
135         elif choice is "U":
136             update(group_of_lists)
137         elif choice is "S":
138             search_lists(group_of_lists)
139         elif choice is "A":
140             create_new_entry(group_of_lists)
141         elif choice is "O":
142             group_of_lists = sort_on_index(group_of_lists,
group_of_lists_header)
143         elif choice is "Q":
144             print("Thank you")
145             run = False
146         else:
147             print("Unrecognised Entry")
148     return None
149
150

```

```
151 if __name__ == "__main__":
152     menu()
```

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