Adam Suttle Year 12

Using Lists (Arrays) Notes - 27.11.16

Array/List = a data structure that allows for multiplied items or elements to be stored using just one variable name.

Each element in a list is accessed using an index position. It is convention that lists begin at 0 and go up to the specified domain.

Creating and displaying a list in python

Empty lists to be filled later:

```
List_nums = [int] * 8
```

 # the multiplication will set the list length (create in this case 8 cells/element positions) and integer shows that the list will store integer data types only.

```
List_names = [string] * 8      # 8 entities of strings like first names

List_nums = [float] * 8        #8 entities of real numbers
```

To fill the entities in a list we can use such things as for loops:

```
For x in range (0 , len(list_names)):                    #len() gives length of list
        list_names[x] = string(input("Enter your first name: "))
```

To display all the results of a list we don't require index positions i.e. **print(list_names)**

….unless we want to specifically output an entity i.e. **print(list_names[3])**

To create a random list of numbers we do the following:

```
Import random
For i in range (0, len(my_list)):
        my_list[i] = random.randint(1,100)            # random integer between 1 and 100
```

Declaring a list where we know the pre-set elements

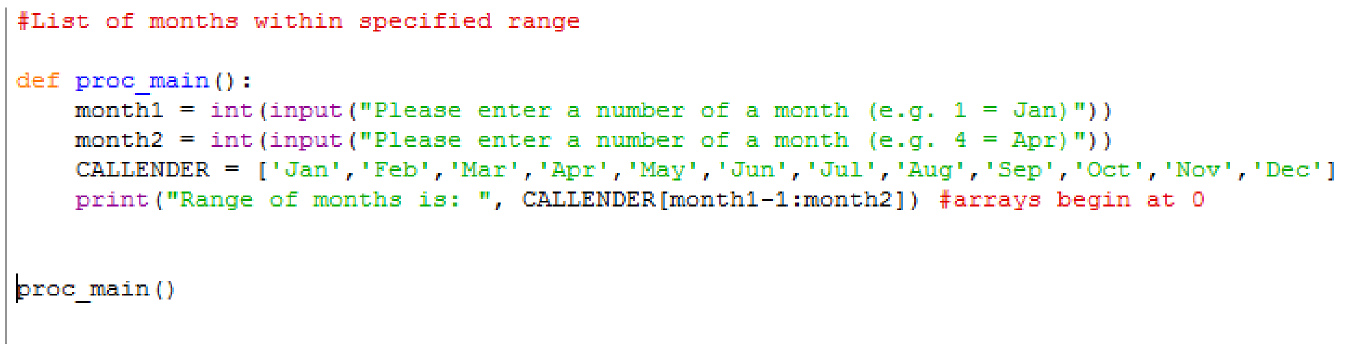
```
shopping = [`milk', `bread', `eggs', `cheese', `cereal']

        print(shopping)

        print(shopping[2])
```

Slicing a list

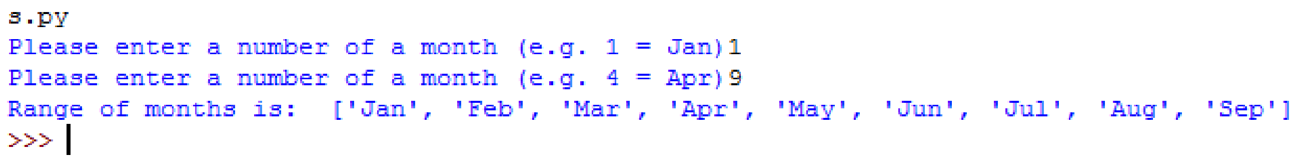
```
shopping = [`milk', `bread', `eggs', `cheese', `cereal']
```

- **print(shopping[:3])**
  This will output the first 3 items - i.e. elements: 0, 1 & 2 or “milk”, “bread”, “eggs”
- **print(shopping[3:])**
  This will output the items from the 3rd element to the end, i.e. 3 & 4 or “cheese”, “cereal”.
- **print(shopping[1:4])**
  This will output the items from the 1st element up to but not including the 4th i.e. 1,2, 3 or “bread”, “eggs”, “cheese”.
- **print(shopping[:-1])**
  This will output all the items except the last 1 elements i.e. 0,1,2, 3 or “milk”, “bread”, “eggs”, “cheese”.
- **print(shopping[-2:])**
  This will only output the last 2 items i.e. 3,4 or “cheese”, “cereal”

My arrays months program



```
#List of months within specified range

def proc_main():
    month1 = int(input("Please enter a number of a month (e.g. 1 = Jan)"))
    month2 = int(input("Please enter a number of a month (e.g. 4 = Apr)"))
    CALLENDER = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
    print("Range of months is: ", CALLENDER[month1-1:month2]) #arrays begin at 0


proc_main()
```

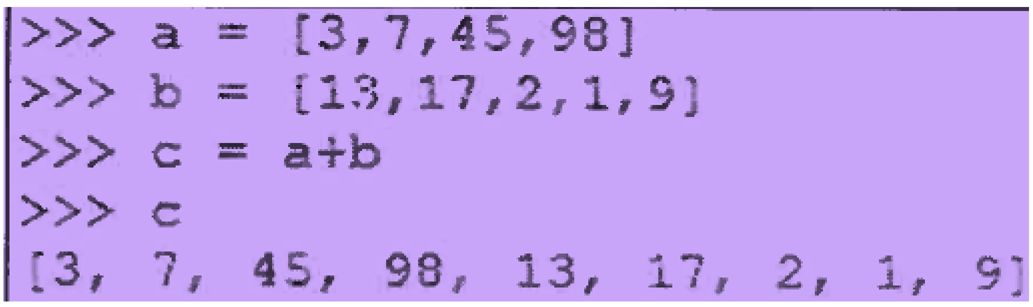


```
s.py
Please enter a number of a month (e.g. 1 = Jan)1
Please enter a number of a month (e.g. 4 = Apr)9
Range of months is:  ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep']
>>> 
```

Manipulating lists

- Adding two lists (this means join, the process is called Concatenation)

e.g.



```
>>> a = [3,7,45,98]
>>> b = [13,17,2,1,9]
>>> c = a+b
>>> c
[3, 7, 45, 98, 13, 17, 2, 1, 9]
```

- Sorting lists numerically or alphabetically

  List_nums = [12,8,3,22,0]
  List_nums = List_nums.sort()
  Print(list_nums)
  >> 0 3 8 12 22

The `.sort() function' will sort the items in the list into numerical order (lowest to highest) by rearranging the items' index positions.

The same .sort() function can be used to sort numerically floating point lists. Moreover it can be sued to order a list of strings into alphabetical order. E.g....

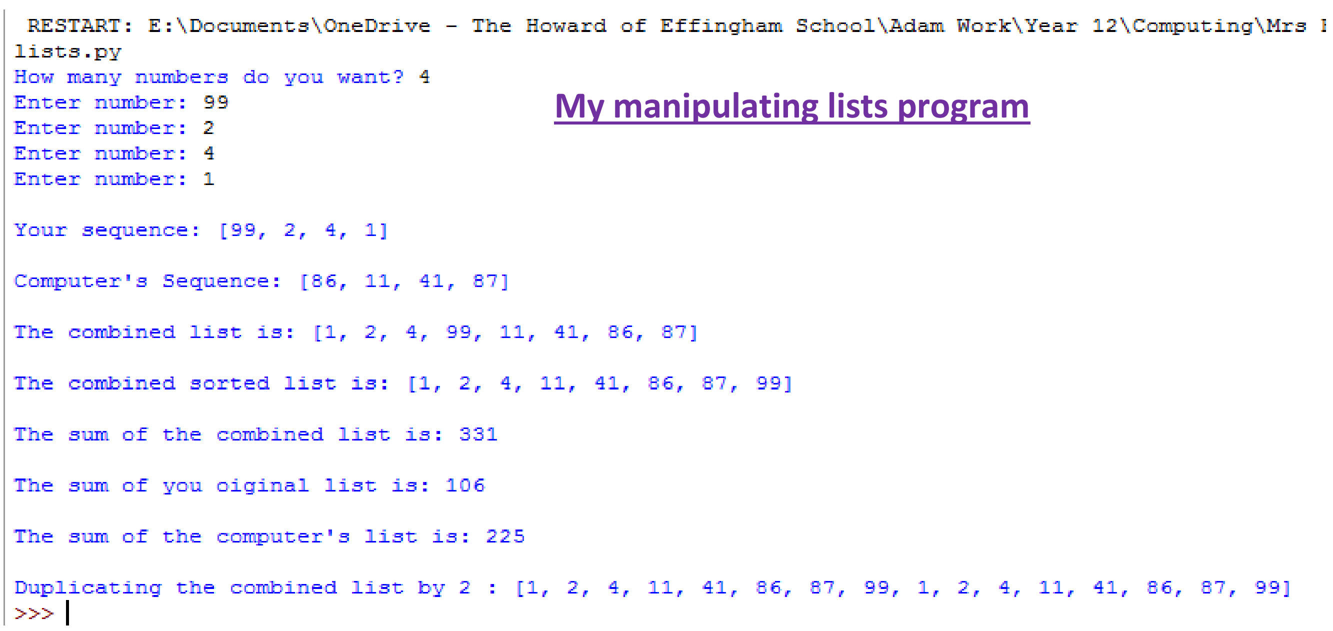
  List_names = [Joe, Craig, Sam, Bob, Aaron]
  List_names = List_names.sort()
  Print(list_names)
  >> Aaron Bob Craig Joe Sam

- Multiplying lists. This will cause reputation of elements.

  List_names = [Joe, Craig, Sam]
  List_names = List_names * 3
  Print(list_names)
  >> Aaron Joe Craig Sam Joe Craig Sam Joe Craig Sam

It will duplicate the list by the number which it is multiplies by. We can only ever multiply by whole integers not floats (we must use techniques of slicing if we say 1½ of the list).
The same thing occurs with a list of integers/floats

  List_nums = [2.5, 11 , 6.6]
  List_nums = List_nums * 3
  Print(list_nums)
  >> 2.5, 11 , 6.6, 2.5, 11 , 6.6, 2.5, 11 , 6.6



My manipulating lists program

```
 RESTART: E:\Documents\OneDrive - The Howard of Effingham School\Adam Work\Year 12\Computing\Mrs
lists.py
How many numbers do you want? 4
Enter number: 99
Enter number: 2
Enter number: 4
Enter number: 1

Your sequence: [99, 2, 4, 1]

Computer's Sequence: [86, 11, 41, 87]

The combined list is: [1, 2, 4, 99, 11, 41, 86, 87]

The combined sorted list is: [1, 2, 4, 11, 41, 86, 87, 99]

The sum of the combined list is: 331

The sum of you oiginal list is: 106

The sum of the computer's list is: 225

Duplicating the combined list by 2 : [1, 2, 4, 11, 41, 86, 87, 99, 1, 2, 4, 11, 41, 86, 87, 99]
>>> 
```

Adam Suttle Year 12

Source Code

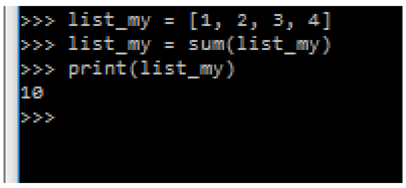
```python
def main_proc2(): #My preferred method       ###
    title_list = ['This', 'manipulates', 'lists', 'in different', 'ways!']
    print(" ".join(title_list)) # the join function joins strings in a list
                                # and the " " will put a space between each item


    length = int(input("How many numbers do you want? "))
    list_nums = [int] * length #sets the list length and declares all entities as integers

    for x in range(0,length):
        list_nums[x] = int(input("Enter number: " ))

    print("\nYour sequence: " + str(list_nums))

    computer_nums = [int] * length
    import random #RANDOM function

    for i in range(0,len(list_nums)):                #Length function is very efficient!
        computer_nums[i] = random.randint(1,90)      #Creates RANDOM list of integers

    print("\nComputer's Sequence: " + str(computer_nums))

    combined_list = (combine_func(computer_nums,list_nums))
    summation_proc(combined_list,list_nums)


def combine_func(computer_nums,list_nums): ####
    combined_list = [int] * (len(list_nums) * 2)     #double the length of 1 list

    #.sort() MUST BE USED BEFORE THE OUPUT TO PREVENT "none" ERROR
    computer_nums.sort()        #.sort() will sort the list lowest to highest if numerical
    list_nums.sort()            # And sort alphabetically if characters are used
    combined_list = list_nums + computer_nums
    print("\nThe combined list is: " + str(combined_list))
    combined_list.sort()
    print("\nThe combined sorted list is: " + str(combined_list))
    return combined_list


def summation_proc(combined_list,list_nums): ####
    print("\nThe sum of the combined list is: " + str(sum(combined_list)))      #The sum of both computer and use list
    print("\nThe sum of you oiginal list is: " + str(sum(list_nums)))           #The sum of the user list
    summation_computer = str(sum(combined_list) - sum(list_nums))
    print("\nThe sum of the computer's list is: " + summation_computer)         #The sum of the computer
    print("\nDuplicating the combined list by 2 : " + str(combined_list * 2))  #The combined list duplicated

main_proc2()
```

Adam Suttle Year 12

Adding the elements within a list (sum function)

The sum() function will add all the items of the lists and give the output as one result.

e.g list_of_numbers = [1,2,3,4]
list_of_numbers = sum(list_of_numbers)
print(list_of_numbers)   >>> 10



```
>>> list_my = [1, 2, 3, 4]
>>> list_my = sum(list_my)
>>> print(list_my)
10
>>>
```

Joining elements of a list

The “”.join function can be used if we want to join strings in a list and the item within the speech icons will be what is placed between each of the elements in the list.

e.g.

```
title_list = ['This', 'manipulates', 'lists', 'in different', 'ways!']
print(" ".join(title_list)) # the join function joins strings in a list
                            # and the " " will put a space between each item
```

```
>>>
        RESTART: E:\Documents\OneDrive - The Howa
       lists.py
       This manipulates lists in different ways!
```

Inserting into a list

The .insert(,) function will insert the element specified after the comma into the index position specified by the number on the left side of the comma.

e.g. **my_list.insert(2,”hello”)** will insert “Hello” into the list at index position 2. The previous element of index position 2 is not deleted as all the elements are shifted along by 1 (list length is increased).

```
def main_proc():
    string_list = ['Caecilius','Metella', 'Marcus', 'Aurellius', 'Julius']
    integer_list = [5, 2, 66, 1, 7]
    print("Original: " + str(string_list))
    print("Original: " + str(integer_list) + "\n")

    def sub_proc_inserting():
        string_list.insert(2, 'Felix')  #Inserts item "Felix" in index position 2
        integer_list.insert(3, 9)       #Inserts item 9 in index position 3

        print(string_list)
        print(integer_list)

    sub_proc_inserting()

|
main_proc()
```

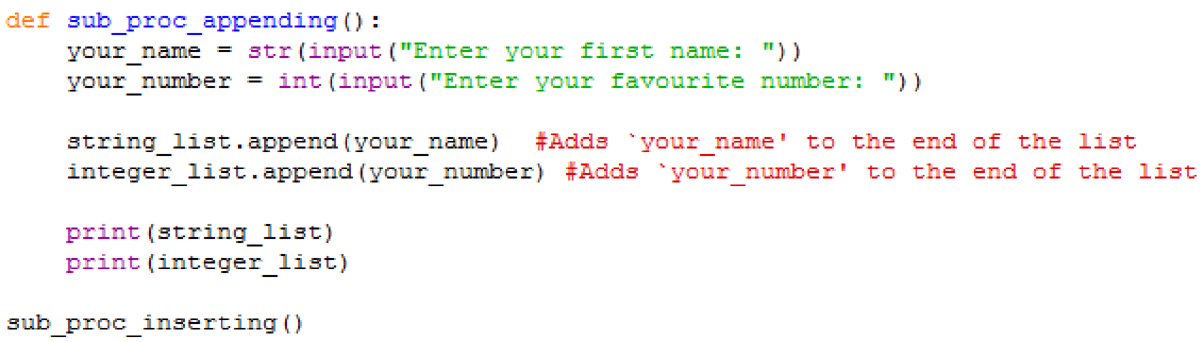
```
Original: ['Caecilius', 'Metella', 'Marcus', 'Aurellius', 'Julius']
Original: [5, 2, 66, 1, 7]

['Caecilius', 'Metella', 'Felix', 'Marcus', 'Aurellius', 'Julius']
[5, 2, 66, 9, 1, 7]
```

Adam Suttle Year 12

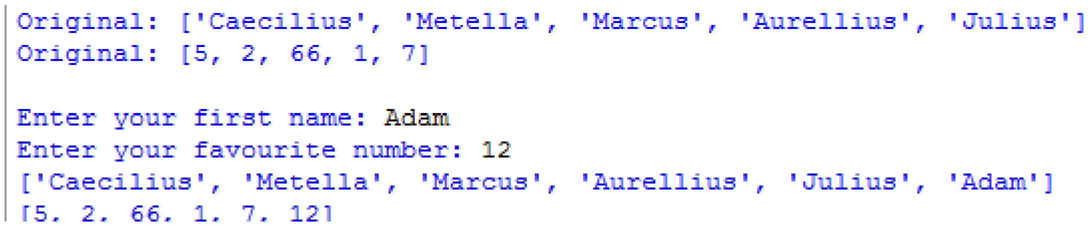
Adding an item to the end of a list (Append)

The .append() function will add the element specified in the bracket onto the end of the list.

e.g. **my_list = my_list.append("Hello")   >>> item1, item2, item3, Hello**



```
def sub_proc_appending():
    your_name = str(input("Enter your first name: "))
    your_number = int(input("Enter your favourite number: "))

    string_list.append(your_name)  #Adds `your_name' to the end of the list
    integer_list.append(your_number) #Adds `your_number' to the end of the list

    print(string_list)
    print(integer_list)

sub_proc_inserting()
```



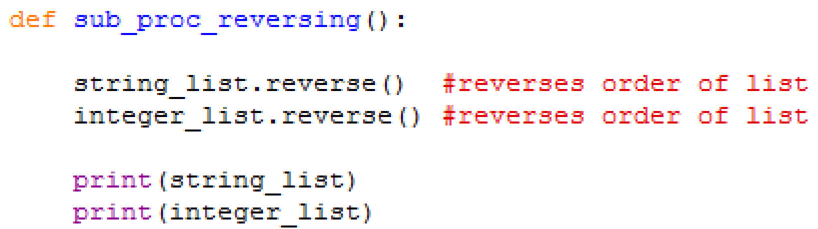
```
Original: ['Caecilius', 'Metella', 'Marcus', 'Aurellius', 'Julius']
Original: [5, 2, 66, 1, 7]

Enter your first name: Adam
Enter your favourite number: 12
['Caecilius', 'Metella', 'Marcus', 'Aurellius', 'Julius', 'Adam']
[5, 2, 66, 1, 7, 12]
```

Finding the length of a list (len() function)

len(my_list) >>> returns the length of the list (number of elements)

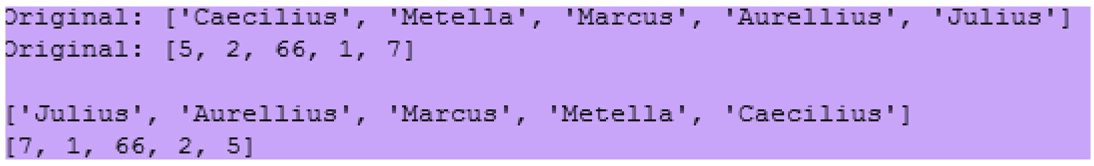
Reversing a list

The **.reverse()** function is used when we want to reverse the list so all the index positions are swapped to other end.



```
def sub_proc_reversing():

    string_list.reverse()  #reverses order of list
    integer_list.reverse() #reverses order of list

    print(string_list)
    print(integer_list)
```

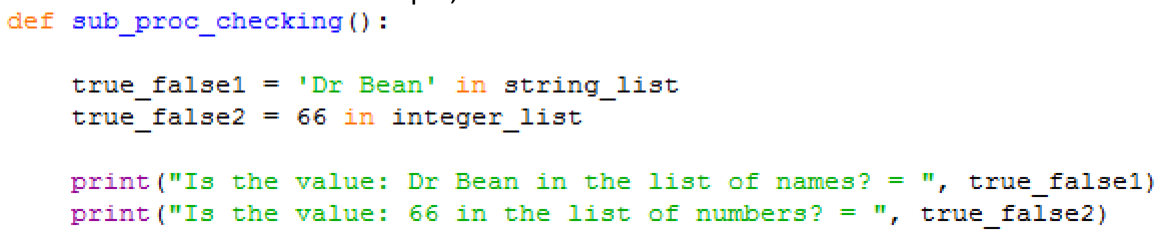
>>>



```
Original: ['Caecilius', 'Metella', 'Marcus', 'Aurellius', 'Julius']
Original: [5, 2, 66, 1, 7]

['Julius', 'Aurellius', 'Marcus', 'Metella', 'Caecilius']
[7, 1, 66, 2, 5]
```

Checking whether an item is in the list

The in operator is used to check if the value given on the left side is present any of the elements in the list.   For example,

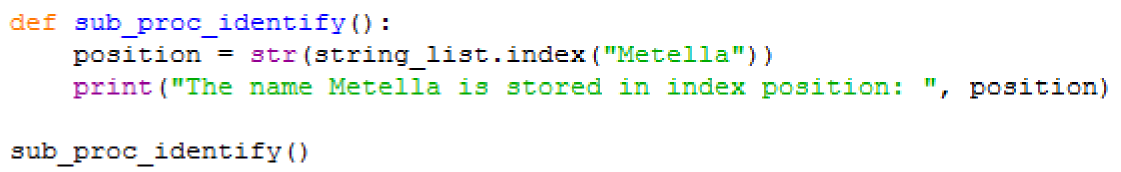


```
def sub_proc_checking():

    true_false1 = 'Dr Bean' in string_list
    true_false2 = 66 in integer_list

    print("Is the value: Dr Bean in the list of names? = ", true_false1)
    print("Is the value: 66 in the list of numbers? = ", true_false2)
```

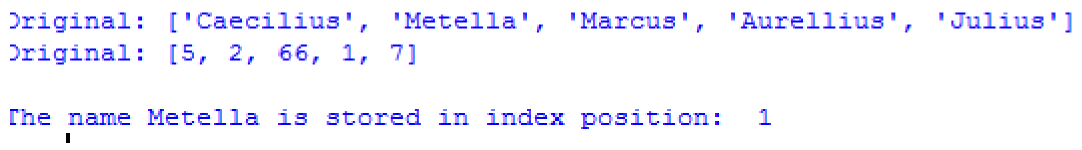
Adam Suttle Year 12

Checking the position of a single item in a list

We use the **.index()** function to find the index position of an item in a list. The brackets is filled with the item we want to locate.



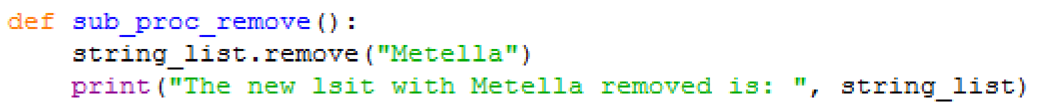
```
def sub_proc_identify():
    position = str(string_list.index("Metella"))
    print("The name Metella is stored in index position: ", position)

sub_proc_identify()
```



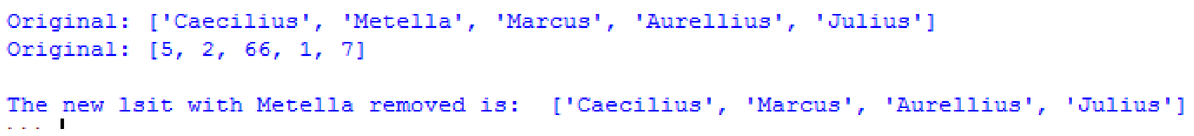
```
Original: ['Caecilius', 'Metella', 'Marcus', 'Aurellius', 'Julius']
Original: [5, 2, 66, 1, 7]

The name Metella is stored in index position:  1
```

Deleting an item from the list

The **.remove()** function will delete the first occurrence of the item specified from your list and so decrease list length.



```
def sub_proc_remove():
    string_list.remove("Metella")
    print("The new lsit with Metella removed is: ", string_list)
```



```
Original: ['Caecilius', 'Metella', 'Marcus', 'Aurellius', 'Julius']
Original: [5, 2, 66, 1, 7]

The new lsit with Metella removed is:  ['Caecilius', 'Marcus', 'Aurellius', 'Julius']
```

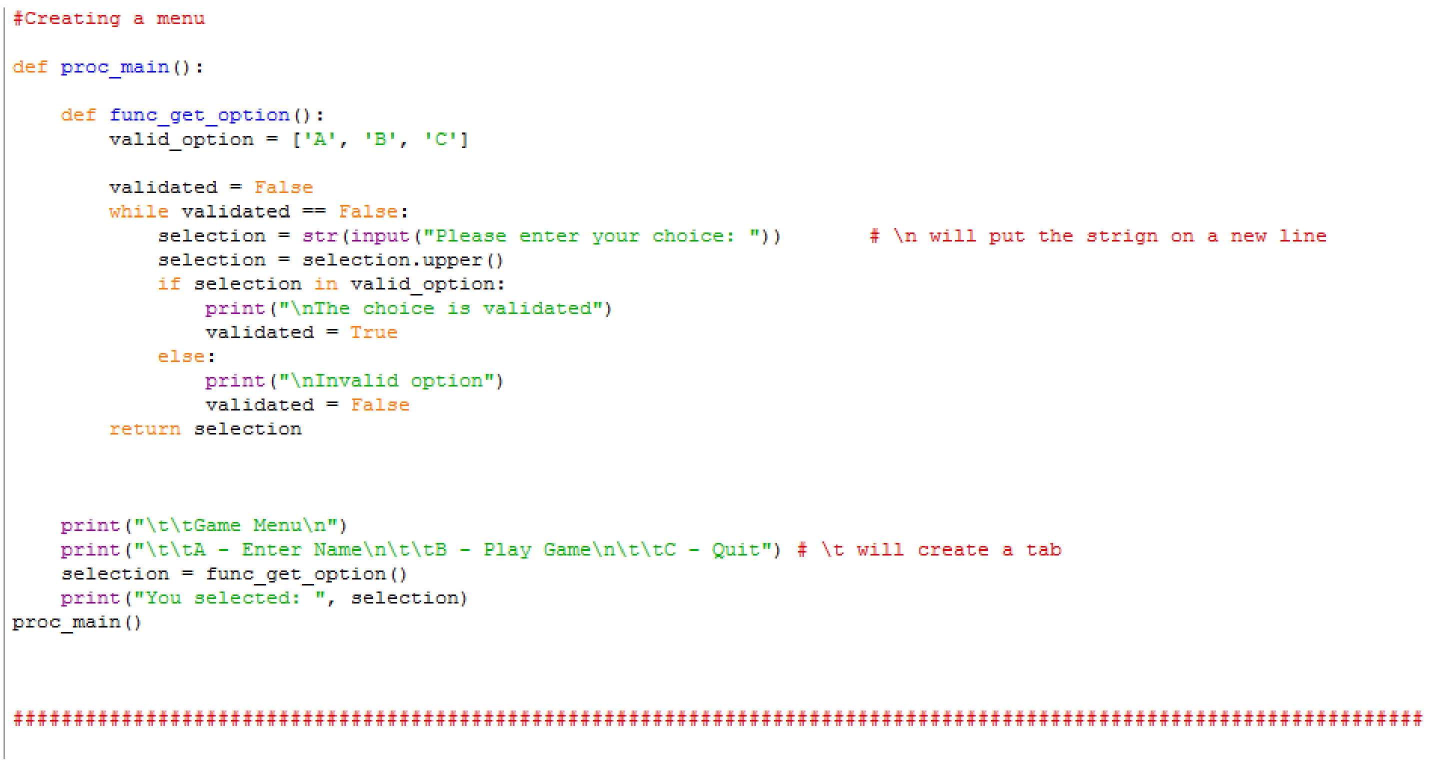
```python
#Identifying list items, inserting into lists, adding items, finding length, deleting items, reverse a list

def main_proc():
    string_list = ['Caecilius','Metella', 'Marcus', 'Aurellius', 'Julius']
    integer_list = [5, 2, 66, 1, 7]
    print("Original: " + str(string_list))
    print("Original: " + str(integer_list) + "\n")

    def sub_proc_inserting():
        string_list.insert(2, 'Felix')  #Inserts item "Felix" in index position 2
        integer_list.insert(3, 9)       #Inserts item 9 in index position 3

        print(string_list)
        print(integer_list)

    def sub_proc_appending():
        your_name = str(input("Enter your first name: "))
        your_number = int(input("Enter your favourite number: "))

        string_list.append(your_name)  #Adds `your_name' to the end of the list
        integer_list.append(your_number) #Adds `your_number' to the end of the list

        print(string_list)
        print(integer_list)

    def sub_proc_reversing():

        string_list.reverse()  #reverses order of list
        integer_list.reverse() #reverses order of list

        print(string_list)
        print(integer_list)

    def sub_proc_checking():

        true_false1 = 'Dr Bean' in string_list
        true_false2 = 66 in integer_list
        print("Is the value: Dr Bean in the list of names? = ", true_false1)
        print("Is the value: 66 in the list of numbers? = ", true_false2)

    def sub_proc_identify():
        position = str(string_list.index("Metella"))
        print("The name Metella is stored in index position: ", position)

    def sub_proc_remove():
        string_list.remove("Metella")
        print("The new lsit with Metella removed is: ", string_list)


    error = True  #option selector
    while error == True:
        user_option = str(input("Select an option 1, 2, 3, 4, 5, 6 for (removing item, identifying location, checking item, reversing list, inserting or appending: "))
        if user_option == "1":
            sub_proc_remove()
            error = False
        elif user_option == "2":
            sub_proc_identify()
            error = False
        elif user_option == "3":
            sub_proc_checking()
            error = False
        elif user_option == "4":
            sub_proc_reversing()
            error = False
        elif user_option == "5":
            sub_proc_inserting()
            error = False
        elif user_option == "6":
            sub_proc_appending()
            error = False
        else:
            print("Invalid option")

continue1 == True  #continue loop
while continue1 == True:
    main_proc()
    continue_option = str(input("Would you like to continue y/n? "))
    continue_option = continue_option.lower()

    if continue_option == "y":
        print("\n")
        main_proc()
    else:
        print("End")
        continue1 = False
```

Escape Sequences

Escape	What it does.
`\\`	Backslash (\)
`\'`	Single-quote (')
`\"`	Double-quote (")
`\a`	ASCII bell (BEL)
`\b`	ASCII backspace (BS)
`\f`	ASCII formfeed (FF)
`\n`	ASCII linefeed (LF)
`\N{name}`	Character named name in the Unicode database (Unicode only)
`\r`	Carriage Return (CR)
`\t`	Horizontal Tab (TAB)
`\uxxxx`	Character with 16-bit hex value xxxx (u'' string only)
`\Uxxxxxxxx`	Character with 32-bit hex value xxxxxxxx (u'' string only)
`\v`	ASCII vertical tab (VT)
`\ooo`	Character with octal value ooo
`\xhh`	Character with hex value hh

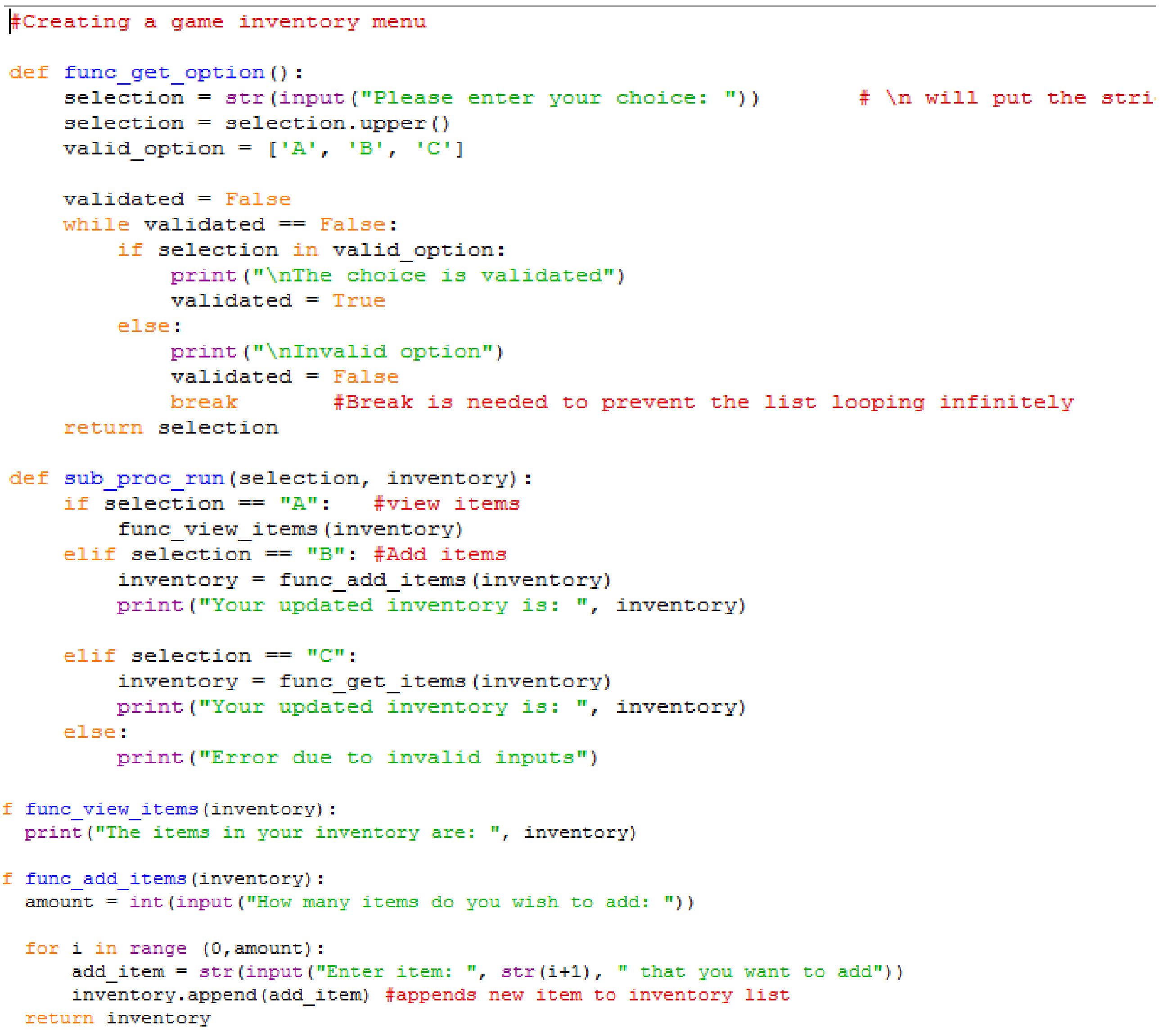
Adam Suttle Year 12

Creating a menu



```
#Creating a menu

def proc_main():

    def func_get_option():
        valid_option = ['A', 'B', 'C']

        validated = False
        while validated == False:
            selection = str(input("Please enter your choice: "))       # \n will put the strign on a new line
            selection = selection.upper()
            if selection in valid_option:
                print("\nThe choice is validated")
                validated = True
            else:
                print("\nInvalid option")
                validated = False
        return selection



    print("\t\tGame Menu\n")
    print("\t\tA - Enter Name\n\t\tB - Play Game\n\t\tC - Quit") # \t will create a tab
    selection = func_get_option()
    print("You selected: ", selection)
proc_main()


##################################################################################################################
```

Creating a game inventory (needs amending due to errors)



```
#Creating a game inventory menu

def func_get_option():
    selection = str(input("Please enter your choice: "))       # \n will put the stri
    selection = selection.upper()
    valid_option = ['A', 'B', 'C']

    validated = False
    while validated == False:
        if selection in valid_option:
            print("\nThe choice is validated")
            validated = True
        else:
            print("\nInvalid option")
            validated = False
            break       #Break is needed to prevent the list looping infinitely
    return selection

def sub_proc_run(selection, inventory):
    if selection == "A":   #view items
        func_view_items(inventory)
    elif selection == "B": #Add items
        inventory = func_add_items(inventory)
        print("Your updated inventory is: ", inventory)

    elif selection == "C":
        inventory = func_get_items(inventory)
        print("Your updated inventory is: ", inventory)
    else:
        print("Error due to invalid inputs")

def func_view_items(inventory):
    print("The items in your inventory are: ", inventory)

def func_add_items(inventory):
    amount = int(input("How many items do you wish to add: "))

    for i in range (0,amount):
        add_item = str(input("Enter item: ", str(i+1), " that you want to add"))
        inventory.append(add_item) #appends new item to inventory list
    return inventory
```

Adam Suttle Year 12

```
def func_get_items(inventory):
    print("The items in your inventory are: ", inventory)

    validated = False
    while validated == False:

        get_item = str(input("Enter the item name that you want: "))
        if get_item in inventory:
            validated = True
            break #breaks from this loop
        else:
            print("Invalid option")
            validated = False

    inventory.remove(get_item) #Removes the specified item from the list
    func_return_get_item(get_item) #since a function can only return one value we send the otehr to another function
    return inventory

def func_return_get_item(get_item):
    print("You are now holding the: ", get_item)
    return get_item

def main_inventory_proc():
    inventory = ['Axe', 'Saw', 'Spade', 'Matches']
    continue_op = True

    while continue_op == True:

        print("\t\tGame Menu - 2\n")
        print("\t\tA - View items\n\t\tB - Add items\n\t\tC - Get item") # \t will create a tab
        selection = func_get_option()

        print("\n")

        sub_proc_run(selection, inventory)
        func_view_items(inventory)

main_inventory_proc()
```