7 June 2024

TOPICS: List and List Methods, Tuple and Tuple Methods

1.colors = ['red', 'blue', 'green', 'yellow']

Using the colors list defined above, print the:

First element, Second element, Last element, Second-to-last element, Second and third elements, Element at index 4.

```
#print first element of the list
color=['red','blue','green','yellow']
num=color[:1]
print(num)
```

```
['red']
```

```
#print second element of the list
num=color[1:2]
print(num)
```

```
['blue']
```

```
#print last element of the list
num=color[-1:]
print(num)
```

```
['yellow']
```

```
#print second-to-last element of the list
num=color[1:4]
print(num)
```

```
['blue', 'green', 'yellow']
```

```
#print second and third element of the list
num=color[1:3]
print(num)
```

```
['blue', 'green']
```

```
#print element at index 4
num=color[4:]
print(num)
```

```
[]
```

2. Below is a list with seven integer values representing the daily water level (in cm) in an imaginary lake. However, there is a mistake in the data. The third day's water level should be 693. Correct the mistake and print the changed list.

water_level = [730, 709, 682, 712, 733, 751, 740]

```
#On third day there should 693 print update value
water_level=[730,709,682,712,733,751,740]
water_level.pop(2)
water_level.insert(2,693)
water_level
```

```
[730, 709, 693, 712, 733, 751, 740]
```

3. Add the data for the eighth day to the list from above. The water level was 772 cm on that day.

```
#On eighth day there shoould be 772
water_level.append(772)
water_level
```

```
[730, 709, 693, 712, 733, 751, 740, 772]
```

```
#delete first element from the list using index number
water_level=[730,709,682,712,733,751,740]
water_level.pop(0)
print(water_level)
```

```
[709, 682, 712, 733, 751, 740]
```

```
#delete first element using its value
water_level=[730,709,682,712,733,751,740]
water_level.remove(730)
print(water_level)
```

```
[709, 682, 712, 733, 751, 740]
```

6.You are managing the inventory for a small bookstore. Create a list of book titles available in the store. Add new titles to the list as they arrive. If a book is sold out, remove it from the list. Write a function to check if a specific book is in stock.

```
#check either the book is present or not in the stock
inventory=["MOBY DICK","THE GREAT INDIAN'S","ON FRIENDSHIP DAY"]
def check_stock(book_title,inventory):
  if book_title in inventory:
    return ("present")
  else:
    return ("not present")
print(check_stock("MOBY DICK",inventory))
print(check_stock("THE MARSCHANT",inventory))
```

```
present
not present
```

7.You have a list of grades for a class of students. Write a function to add a new grade to the list. Another function should remove the lowest grade. Write a third function to calculate the average grade.

```
grades=[83,73,71,67,52]
def add_new(grades,new_grade):
  grades.extend(new_grade)
  return grades
new_grade=[96,79]
update=add_new(grades,new_grade)
print(update)


def remove_lowest(grades):
  grades.pop(3)
  grades.pop(4)
  return grades
update=remove_lowest(grades)
print(update)

grades=[83,73,71,67,52]
def avg_grade(grades):
  if not grades:
   return None
   return sum(grades)/len(grades)

calculate=avg_grade(grades)
print(calculate)
```

```
[83, 73, 71, 67, 52, 96, 79]
[83, 73, 71, 52, 79]
None
```

8.Implement a simple to-do list application. Create a list to store tasks. Write functions to add a task, remove a task by its name, and display all tasks. Ensure that the tasks are displayed in the order they were added.

```
tasks=["task1","task2","task3"]
def add_task(tasks,new_task):
  tasks.extend(new_task)
  return tasks
new_task=["task4"]
add=add_task(tasks,new_task)
print(add)

def remove(tasks):
  tasks.pop(3)
  return tasks
  update=remove(tasks)
  print(update)

def update(tasks):
 print(add)
```

```
['task1', 'task2', 'task3', 'task4']
```

9.- You are tracking daily temperatures for a month. Create a list to store these temperatures. Write functions to find the highest and lowest temperatures, and to calculate the average temperature for the month.

```
temperatures=[22,23,24,25,26,27,28,29,25,26,22,24,25,23,22,26,22,24,25,23,22,26,22,24,25,23,22,26,22,24,25,27]
def temperature_status(tempertaure):
   highest_temperature=max(temperatures)
   lowest_temperature=min(temperatures)
   avg_temperature=sum(temperatures)/len(temperatures)
   return highest_temperature,lowest_temperature,avg_temperature
   highest,lowest,avg=temperature_status(temperatures)

   print(f"highest_temperature:{highest}degree celsius")
   print(f"lowest_temperature:{lowest}degree celsius")
   print(f"avg_temperature:{avg:.2f}degree celsius")
```

10.You are managing a library system. Create a list of books currently borrowed by patrons. Write functions to add a borrowed book, return a book (remove it from the list), and check if a specific book is currently borrowed

```
borrowed_books = []

def add_borrowed_book(book_title):
    borrowed_books.append(book_title)

def return_book(book_title):
    if book_title in borrowed_books:
        borrowed_books.remove(book_title)
        return True
    else:
        return False

def is_book_borrowed(book_title):
    return book_title in borrowed_books

add_borrowed_book("Harry Potter and the Sorcerer's Stone")
add_borrowed_book("The Great Gatsby")
add_borrowed_book("To Kill a Mockingbird")

print("List of borrowed books:", borrowed_books)

print("\nReturning 'The Great Gatsby'")
return_book("The Great Gatsby")

print("Is 'To Kill a Mockingbird' borrowed", is_book_borrowed("To Kill a Mockingbird"))
print("List of borrowed books after return:", borrowed_books)
```

```
List of borrowed books: ["Harry Potter and the Sorcerer's Stone", 'The Great Gatsby', 'To Kill a Mockingbird']

Returning 'The Great Gatsby'
Is 'To Kill a Mockingbird' borrowed True
List of borrowed books after return: ["Harry Potter and the Sorcerer's Stone", 'To Kill a Mockingbird']
```