Double-click (or enter) to edit

Lab2 bmi.py

```
def calculate_bmi(weight, height):
    weight = float(weight)
    height = float(height)

    bmi = weight/ (height ** 2)
    return_value = print_result(weight, height, bmi)
    return return_value


def analyse_bmi(bmi):
    if bmi < 18.5:
        condition = 'under weight'
        val = -1
    elif bmi >= 18.5 and bmi <= 25.0:
        condition = 'normal weight'
        val = 0
    elif bmi > 25.0:
        condition = 'over weight'
        val = 1
    print(f"You are {condition.title()}.")
    return val


def print_result(weight, height, bmi):
    print(f"Your Height is {height} m.")
    print(f"Your Weight is {weight} kg.")
    #print(f"Your Bmi is {bmi}.")
    print(f"Your Bmi is {round(bmi, 2)}.")
    #print(f"Your Bmi is {bmi:.2f}.")
    #print(f"Your Bmi is {"%.2f" % bmi}.")

    return_value = analyse_bmi(bmi)

    return return_value


active = True
def main():
    global active
    while active:
        print("Enter 'q' anytime to quit the program.")
        height = input("Enter your height in m: ")
        if height == 'q':
            break
        weight = input("Enter your weight in kg: ")
        if weight == 'q':
            break
        calculate_bmi(weight=weight, height=height)
        again = input("Do you want to calculate again? (Yes/No): ")
        if again.lower() == 'no':
            active = False

if __name__ == "__main__":
    main()
```

Lab2.py

```
#print("ET0735 (DevOps for AToT) - Lab2 - Introduction to Python")
def display_main_menu():
    print('Enter some numbers seperated by commas (e.g. 5, 67, 32)')


def get_user_input():
    input_string = input()
    input_list = input_string.split(',')
    final_input_list = []
    for value in input_list:
        final_input_list.append(float(value.strip()))
    return final_input_list
```

```
def calc_averge(list):
    average = round(sum(list)/len(list), 2)
    return average


def find_min_max(list):
    min_temp = int(min(list))
    max_temp = int(max(list))
    min_max = [min_temp, max_temp]
    return min_max

def sort_temperature(list):
    ascending_temps = sorted(list)
    return ascending_temps

def calc_median_temperature(list):
    temp_num = len(list)
    half_temp_num = int(temp_num / 2)
    if temp_num % 2 == 0:
        median = (list[half_temp_num] + list[half_temp_num - 1]) / 2
    elif temp_num % 2 == 1:
        median = list[half_temp_num]
    return median
```

Lab2_main.py

```
import Lab2 as l2

def main():
    l2.display_main_menu()
    user_inputs = l2.get_user_input()
    average = l2.calc_averge(user_inputs)
    min_max = l2.find_min_max(user_inputs)
    ascending_temps = l2.sort_temperature(user_inputs)
    median = l2.calc_median_temperature(ascending_temps)

    print(f"Average Temperature is {average}.")
    print(f"Minimum Temperature - {min_max[0]}\nMaximum Temperature - {min_max[1]}")
    print(f"Median Temperature is {median}.")


if __name__ == "__main__":
    main()
```

test_Lab2.py

```
import Lab2 as l2

def test_find_min_max():
    expected_result = [1,7]
    input = [1,2,3,4,5,6,7]
    result = l2.find_min_max(input)

    assert (result == expected_result)

def test_calc_average():
    expected_result = 3.5
    input = [1,2,3,4,5,6]
    result = l2.calc_averge(input)

    assert(result == expected_result)

def test_calc_median_temperature_():
    expected_result = 4
    input = [1,2,3,4,5,6,7]
    result = l2.calc_median_temperature(input)

    assert(result == expected_result)
```

Employee_info.py

```
# Define a dictionary to store employee information
employee_data = [
    {"name": "John", "age": 30, "department": "Sales", "salary": 50000},
    {"name": "Jane", "age": 25, "department": "Marketing", "salary": 60000},
    {"name": "Mary", "age": 23, "department": "Marketing", "salary": 56000},
```

```
    {"name": "Chloe",  "age": 35, "department": "Engineering", "salary": 70000},
    {"name": "Mike", "age": 32, "department": "Engineering", "salary": 65000},
    {"name": "Peter", "age": 40, "department": "Sales", "salary": 60000}
]

def get_employees_by_age_range(age_lower_limit, age_upper_limit):
    result = []

    # check for age limits and append the item to result
    for item in employee_data:
        if int(item["age"]) > int(age_lower_limit) and int(item["age"]) < int(age_upper_limit):
            result.append(item)

    return result

def calculate_average_salary():
    total = 0
    average = 0

    #add your implementation to calculate here
    for employee in employee_data:
        total += employee["salary"]
    average = round(total / len(employee_data), 5)

    return average

def get_employees_by_dept(department):
    result = []

    # Add your implementation from here
    for employee in employee_data:
        if employee['department'] == department:
            result.append(employee)

    return result

def display_all_records():
    print(("Name" + "\t" +"Age" +"\t" +"Department" +"\t" +"Salary" ).expandtabs(15))
    for item in employee_data:
        print((item["name"] + "\t" + str(item["age"]) + "\t" + item["department"] + "\t" + str(item["salary"])).expandtabs(15))


def display_records(employee_info):
    print(("Name" + "\t" +"Age" +"\t" +"Department" +"\t" +"Salary" ).expandtabs(15))
    for item in employee_info:
        print((item["name"] + "\t" + str(item["age"]) + "\t" + item["department"] + "\t" + str(item["salary"])).expandtabs(15))

def display_main_menu():

    print("\n----- Employee information Tracker -----")

    print("Select option\n")

    print("1 - Display all records")
    print("2 - Display average salary")
    print("3 - Display employee within age range")
    print("4 - Display employee in a department")


    print("Q - Quit")

    option = input("Enter selection =>")

    if option == '1':
        display_all_records()

    elif option == '2':
        average_salary = calculate_average_salary()
        print("Average salary = " + str(average_salary))

    elif option == '3':
        age_lower_limit = input("age (Lower Limit) = ")
        age_upper_limit = input("age (Uper Limit) = ")
        employee_info = get_employees_by_age_range(age_lower_limit, age_upper_limit)
        display_records(employee_info)


    elif option == '4':
        department = input("Name of Department = ")
        employee_info = get_employees_by_dept(department)
        display_records(employee_info)
```

```
    elif option == 'Q':
        quit()

def main():

    while (True):
        display_main_menu()


if __name__ == "__main__":
    main()
```

test_employee_info.py

```
import employee_info as ei

def test_get_employees_by_age_range():
    expected_result = [
        {"name": "Jane", "age": 25, "department": "Marketing", "salary": 60000},
        {"name": "Mary", "age": 23, "department": "Marketing", "salary": 56000},
        ]
    result = ei.get_employees_by_age_range(20,30)
    assert (result == expected_result)

def test_calculate_average_salary():
    expected_result = 60166.66667
    result = ei.calculate_average_salary()
    assert (result == expected_result)

def test_get_employees_by_dept():
    expected_result = [
        {"name": "Jane", "age": 25, "department": "Marketing", "salary": 60000},
        {"name": "Mary", "age": 23, "department": "Marketing", "salary": 56000}
    ]
    result = ei.get_employees_by_dept("Marketing")
    assert (result == expected_result)
```

Lab3.py

```
print("Lab 3 - Software Unit Testing with PyTest")

SORT_ASCENDING = 0
SORT_DESCENDING = 1


def bubble_sort(arr, sorting_order):

    # Copy input list to results list
    arr_result = arr.copy()

    # Get number of elements in the list
    n = len(arr_result)
    if n > 0:
        for value in arr_result:
            if type(value) is not int:
                arr_result = 2
            else:
                if n > 0 and n < 10:
                    # Traverse through all array elements
                    for i in range(n - 1):
                        # range(n) also work but outer loop will
                        # repeat one time more than needed.

                        # Last i elements are already in place
                        for j in range(0, n - i - 1):

                            if sorting_order == SORT_ASCENDING:
                                if arr_result[j] > arr_result[j + 1]:
                                    arr_result[j], arr_result[j + 1] = arr_result[j + 1], arr_result[j]


                            elif sorting_order == SORT_DESCENDING:
                                if arr_result[j] < arr_result[j + 1]:
                                    arr_result[j], arr_result[j + 1] = arr_result[j + 1], arr_result[j]

                            else:
                                # Return an empty array
                                arr_result = []
                elif n >= 10:
```

```
                    arr_result = 1
    elif n == 0:
        arr_result = 0

    return arr_result

def main():
    # Driver code to test above
    arr = [64, 34, 25, 12, 22, 11, 90]

    # Sort in ascending order
    result = bubble_sort(arr, SORT_ASCENDING)
    print("\nSorted array in ascending order: ")
    print(result)

    # Sort in descending order
    print("Sorted array in descending order: ")
    result = bubble_sort(arr, SORT_DESCENDING)
    print(result)

if __name__ == "__main__":
    main()
```

test_Lab3.py

```
import Lab3

print("Test_Lab3")


def test_bubble_sort_ascending():
    result = []
    input_arr = [64, 34, 25, 12, 22, 11, 90]
    test_arr = [11, 12, 22, 25, 34, 64, 90]

    result = Lab3.bubble_sort(input_arr, Lab3.SORT_ASCENDING)

    assert (result == test_arr)

def test_bubble_sort_descending():
    result = []
    input_arr = [64, 34, 25, 12, 22, 11, 90]
    test_arr = [90, 64, 34, 25, 22, 12, 11]

    result = Lab3.bubble_sort(input_arr, Lab3.SORT_DESCENDING)

    assert (result == test_arr)

def test_bubble_sort_input_greater_or_equal_10():
    input_arr = [64, 34, 25, 12, 22, 11, 90, 11, 12, 13, 14, 15]
    result = Lab3.bubble_sort(input_arr, Lab3.SORT_DESCENDING)

    assert (result == 1)

def test_bubble_sort_no_input():
    input_arr = []
    result = Lab3.bubble_sort(input_arr, Lab3.SORT_DESCENDING)

    assert (result == 0)

def test_bubble_sort_not_integer():
    input_arr = [12, 13, 14, 1.5, 15]
    result = Lab3.bubble_sort(input_arr, Lab3.SORT_DESCENDING)

    assert (result == 2)

def test_bubble_sort_invalid():
    result = []
    input_arr = [64, 34, 25, 12, 22, 11, 90]

    result = Lab3.bubble_sort(input_arr, 3)

    assert (result == [])
```

price_info.py

```
price_list={'apple' : 1.20, 'orange':1.40, 'watermelon': 6.50, 'pineapple': 2.70, 'pear' : 0.90, 'papaya': 2.95, 'pomegranate': 4.95 }

quantity_list= {'apple': 5, 'orange':5, 'watermelon': 1, 'pineapple': 2, 'pear' : 10, 'papaya': 1, 'pomegranate': 2}


def total_cost_shopping():
    total_cost = 0
    for key in price_list.keys():
        if key in quantity_list:
            # complete the implementation below:
            total_cost += price_list[key] * quantity_list[key]

    print("total cost = ", total_cost)
    return total_cost


def cost_of_fruits(fruit, quantity):
    for key in price_list.keys():
        if key == fruit:
            cost = quantity*price_list[key]
            break

    print("cost of ", quantity, fruit, "=", cost)
    return cost


def main():

    cost_of_fruits('apple', 10)
    total_cost_shopping()


if __name__ == "__main__":
    main()
```

test_price_info.py

```
import price_info as pi

def test_total_cost_shopping():
    expected_result = 46.75
    result = pi.total_cost_shopping()
    assert (result == expected_result)

def test_cost_of_fruit():
    expected_result = 4.5
    result = pi.cost_of_fruits('pear', 5)
    assert (result == expected_result)
```

Readme.py

```
# ET0735 - Lab2 (Introduction to Python)
**Khin Myat ~~Min~~ *Myat* Min use `git init`**
```python
print(f"2 + 3 is {2+3}")
```

1. Wake up

    - hello

        a.

[path to bmi.py](/Lab2_practice/bmi.py)

![alt](/images/Screenshot%202024-11-14%20003953.png)
```