MSCS & MSDS OOP WITH PYTHON ASSIGNMENT 2, ADVENT 2025

Instruction: Submit a GitHub repository link and a well-documented Jupyter Notebook (.ipynb) file and/or .py file(s) on Canvas or Moodle. Ensure the notebook runs without errors and includes clear explanations, code, and outputs.

Q1: Electricity Billing Optimization

You are analyzing monthly household electricity consumption data for 10 customers in Kampala. Each record contains: customer_name, units_used, connection_type (e.g., 'Bosco', 245, 'Domestic').

Attempt the following tasks:

1.1. Use zip() and map() to pair customer names with total bills computed by a lambda:
- Domestic rate = 820 UGX/unit
- Commercial rate = 1000 UGX/unit

1.2. Use filter() to separate high-consumption customers (units > 500).

1.3. Using *args, write a function average_bill(*bills) that computes the average.

1.4. Display results using list comprehension in the format: ['Bosco: UGX 200,900', 'Jane: UGX 400,000', ...]

Q2: Market Basket Price Aggregator

Each vendor reports different fruit prices daily:

```
mango_prices = [2500, 2700, 2600, 2800]
orange_prices = [3000, 3200, 3100, 3050]
apple_prices = [4500, 4600, 4550, 4700]
```

Attempt the following tasks:

2.1 Use zip_longest() to handle missing prices.
2.2 Define a lambda function avg = lambda lst: sum(lst)/len(lst) to compute average per fruit.
2.3 Create a generator expression that yields all fruits with average price above 3000.
2.4 Print the generator’s results neatly as: Fruits Above Average Price: Mango, Orange, Apple

Q3: District Temperature Tracker

Temperature data (°C):

```
kampala = [28, 30, 29, 27, 26, 29, 30, 31, 32, 30, 29, 28]
gulu = [25, 26, 27, 27, 28, 29, 30, 30, 29, 27, 26, 25]
mbarara = [22, 23, 23, 24, 25, 25, 26, 27, 27, 26, 24, 23]
```

Attempt the following tasks:

3.1. Use map() with lambda to convert all temperatures to Fahrenheit.
3.2. Use zip() to pair months (Jan–Dec) with Kampala temps.
3.3. Use filter() to identify months with temperature > 30°C.
3.4. Create a generator to yield month names and temperatures above threshold.
3.5. Print as: Hot months in Kampala: July - 31°C, August - 32°C

Q4: School Fees Payment Analyzer

Installment data:

```
students = ['Alex', 'Grace', 'Sarah', 'Brian']
installments = [
  [150000, 200000, 250000],
  [500000, 0, 200000],
  [300000, 300000, 300000],
  [400000, 100000, 0]
]
```

Attempt the following tasks:

4.1. Write a lambda function that sums valid payments and ignores zeros.
4.2. Use map() to compute total paid by each student.
4.3. Combine student names and totals using zip().
4.4. Use filter() to find students who have cleared full fees (≥600,000).
4.5. Create payment_summary(**kwargs) that receives named args like Alex=600000 and prints summaries.

Q5: Agricultural Yield Estimator

Yield data:

```
districts = ['Bushenyi', 'Mityana', 'Kasese', 'Mbale']
yield_data = [1200, 1500, 900, 1300]
```

Attempt the following tasks:

5.1. Use a list comprehension with a lambda to convert yields from kg to tons.
5.2. Write a generator that yields only districts with yield > 1 ton.
5.3. Use *args to compute average yield across all districts.
5.4. Use **kwargs to simulate price changes per district and compute revenue (tons × price).
5.5. Print formatted output like: Kasese produced 0.9 tons — Revenue: UGX 4,500,000.

Q6: Web Data Aggregation

Given that sites = ['https://ucu.ac.ug', 'https://harba.ug', 'https://www.bou.or.ug']

Attempt the following tasks:

6.1. Write a function that takes *urls and uses requests.get() to return response codes.
6.2. Use a list comprehension to print only URLs with status code 200.
6.3. Store results in a dictionary using a dictionary comprehension.
6.4. Create a generator expression that yields 'Active Site: <url>' for reachable domains.