Nexaquanta Senior Computer Vision Engineer Evaluation

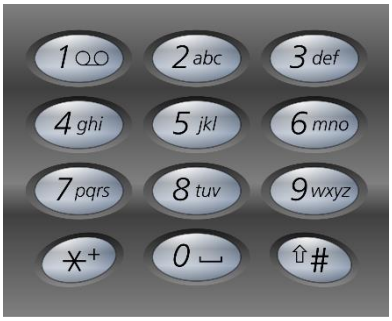
Instructions

- Complete all the tasks presented in the exam.
- Prior to submission, thoroughly test your code to ensure functionality.
- Non-functional code will not be awarded any credit.
- Adhere strictly to the instructions provided for each problem.
- Documentation is a significant component, carrying 30% of the overall weightage.
- Account for corner cases, as the code will undergo testing with various scenarios to evaluate performance metrics such as execution time and memory usage.

Problem1

Develop a Python function that, given a string composed of digits ranging from 2 to 9 inclusively, generates and returns all the feasible letter combinations that the input number could represent. Return the answer in any order. Use the function prototype defined at the end of the problem.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



Example 1:

Input: `digits = "23"`

Output: `["ad","ae","af","bd","be","bf","cd","ce","cf"]`

Example 2:

Input: `digits = ""`

Output: `[]`

Example 3:

Input: `digits = "2"`

Output: `["a","b","c"]`

Constraints:

`0 <= digits.length <= 4`

`digits[i]` is a digit in the range `['2', '9']`

Function Prototpye

```
def letter_combinations(digits: str) -> List[str]:
      """
      :type digits: str
      :rtype: List[str]
      """
```

Problem 2

Table: Customer

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| customer_id | int     |
| product_key | int     |
+-------------+---------+
```

This table may contain duplicates rows. `customer_id` is not NULL. `product_key` is a foreign key (reference column) to Product table.

Table: Product

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| product_key | int     |
+-------------+---------+
```

`product_key` is the primary key (column with unique values) for this table.

Problem Statement

Write a solution to report the customer ids from the Customer table that bought all the products in the Product table. Return the result table in any order.

The result format is in the following example.

Example 1:

Customer table:

```
+-------------+-------------+
| customer_id | product_key |
+-------------+-------------+
| 1           | 5           |
| 2           | 6           |
| 3           | 5           |
| 3           | 6           |
| 1           | 6           |
+-------------+-------------+
```

Product table:

```
+-------------+
| product_key |
+-------------+
| 5           |
| 6           |
+-------------+
```

Output:

```
+-------------+
| customer_id |
+-------------+
| 1           |
| 3           |
+-------------+
```

Explanation:

The customers who bought all the products (5 and 6) are customers with IDs 1 and 3.

Problem 3

Utilize only bash scripting or Linux commands to address the following task: Extract the content of the 10th line from a text file named `file.txt` and display it.

Example:

Assume that `file.txt` has the following content:

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
```

Your script should output the tenth line, which is:

```
Line 10
```

Problem4

Below is code for translating a given word into three different locales. It is assumed that only three specific words need to be translated into different locales.

```
class FrenchLocalizer:
     """ it simply returns the french version """
     def __init__(self):
         self.translations = {"car": "voiture", "bike": "bicyclette",
                              "cycle":"cyclette"}
     def localize(self, msg):
         """change the message using translations"""
        return self.translations.get(msg, msg)

class SpanishLocalizer:
    """it simply returns the spanish version"""
     def __init__(self):
         self.translations = {"car": "coche", "bike": "bicicleta",
                              "cycle":"ciclo"}
     def localize(self, msg):
         """change the message using translations"""
        return self.translations.get(msg, msg)

class EnglishLocalizer:
    """Simply return the same message"""
     def localize(self, msg):
        return msg
```

You are tasked with translating words into different locales using two implementations that utilize the provided classes. Both implementations yield identical outputs and demonstrate similar performance. Your objective is to compare these implementations and provide insights on which one is superior, along with reasoning for your choice.

Implementation 1

```
if __name__ == "__main__":
    # main method to call others
    f = FrenchLocalizer()
    e = EnglishLocalizer()
    s = SpanishLocalizer()

    for msg in ["car", "bike", "cycle"]:
        print(f.localize(msg))
        print(e.localize(msg))
        print(s.localize(msg))
```

Implementation 2

```
def Localizer (language ="English"):
    """Localizer Method"""
    localizers = {
        "French": FrenchLocalizer,
        "English": EnglishLocalizer,
        "Spanish": SpanishLocalizer,
    }
    return localizers[language]()

if __name__ == "__main__":

    f = Localizer("French")
    e = Localizer("English")
    s = Localizer("Spanish")

    for msg in ["car", "bike", "cycle"]:
        print(f.localize(msg))
        print(e.localize(msg))
        print(s.localize(msg))
```

Problem 5

Consider the following dataset:

https://www.kaggle.com/datasets/andrewmvd/road-sign-detection

Your job is to design a solution for an object detection software system in python for this dataset. You are free to consult other notebooks and solutions available on Kaggle or other sources.

The final solution you need to prepare your own notebook, or code and documents.

It will be good if you try two different deep learning models for object detection. It's up to you to choose which models. Present the case, as to which model should be chosen as the final solution.

You are free to choose the metrics, testing methodology, analysis types, supporting results etc., to support your solution. You may train your models if you wish, but you may choose just the pretrained weights.

This is an opportunity for you to showcase your skills in a practical but brief way. We are not looking for an exhaustive approach, where you have tried every model and every technique.