I

This section consists of 6 short question tests basic python concepts. Although in practice, the approach/method is completely up to you, in this section you may encounter questions that specifically tell you how to code the solution . This is only done to see if you're knowledgeable in the concepts in question.

Question

Question 1 - Dates

Complete the function `is_date_format_correct()` below . The function accepts a string as in input and returns a Boolean indicating whether or not that string is of the format `YYYY-MM-DD`

Sample input and result:

`is_date_format_correct('2022/01/01')` ⇒ False

`is_date_format_correct('1999-01-01')` ⇒ True

`is_date_format_correct('20221129')` ⇒ False

NB: Do not remove any of the original code. Only add to it

```
def is_date_format_correct(date:str)->bool:
    """
    This function takes in a date in string format
    and returns true if the date matches the format
    YYYY-MM-DD and false if it doesn't
    """

    #your code here#
```

NB: Do not remove any of the original code. Only add to it

Question 2 - Code flow

Complete the below function such that it prints `1,2,3,4,5,7,8,9,10,`

NB: Do not remove any of the original code. Only add to it

```
for i in range(1, 11):

    if i==6:
        #your code here#

    else:
        print(i, end=',')
```

NB: Do not remove any of the original code. Only add to it

Question 3 - List comprehensions

For this problem, you will be required to use a python's *list comprehension*

Complete the function below, `compute_prev_date()`, that will take a list of dates `dates_list` in this format: `YYYY-MM-DD` and for each date returns the **previous** date in the format `DD xxx YYYY` where `xxx` is the month as locale's abbreviation (e.g Jan, Sep, Oct, Dec ctc). The function should therefore also return a list

Example:

Input ⇒ `['2022-11-01', '2022-11-02']`

Desired Output ⇒ `['31 Oct 2022', '01 Nov 2022']`

You can see that in the desired output list, each date is the corresponding previous day from the input list in the format `DD xxx YYYY`

More examples

`['1999-01-21']` should return `['20 Jan 1999']`

`['1999-01-21', '2022-12-30', '2099-12-21']` should return `['20 Jan 1999', '29 Dec 2022', '20 Dec 2099']`

NB: Do not remove any of the original code. Only add to it

```
def compute_prev_date(dates_list:list):
  """
  """


  return [ """your code here""" ]
```

NB: Do not remove any of the original code. Only add to it

Question 4 - Basic exception handling

You are working on a project with one your fellow software developer and this is what you're told about the below function `main()`:

- The functions `fetch_quantity()` and `fetch_cost()` are very error prone and sometime exceptions are thrown from inside these 2 functions causing the whole code to crash. These 2 functions are very long complex and you should not worry about what happens inside them.
- Editing the below code such that if an exception is raised from inside `fetch_cost()` (i.e if something goes wrong with that function) the program should continue running to the next step, BUT if something goes wrong with `fetch_quantiy()` or the line `cost_per_quantity = cost/qty` then it should print the exception and terminate immediately

NB: Do not remove any of the original code. Only add to it

```
def main():

    qty = None
    cost = None
```

```
def fetch_quantity():
    """
    Returns a number, any number
    """
    ...
    return ...

def fetch_cost():
    """
    Returns a number, any number
    """
    ...
    return ...


def compute_cost_per_quantity():


    qty = fetch_quantity()
    cost = fetch_cost()
    cost_per_quantity = cost/qty

    return cost_per_quantity


cost_per_quantity = compute_cost_per_quantity()
a = 1 + 2 + cost_per_quantity
b = 4 + 5
print(a+b)
```

NB: Do not remove any of the original code. Only add to it

Question 5 - Django rest framework

Here you will be tested on whether you can read basic docs.

Assume your api is running on local host port 8000 ⇒ `127.0.0.1:8000`

How would you complete the below view function, `get_params()` so that it returns the `name` and `surname` parameters in the json response from the following request url:

`http://127.0.0.1:8000/get_params?name=John&surname=Doe`

```
from rest_framework import status
from rest_framework.response import Response
from rest_framework.decorators import api_view

@api_view(['GET'])
def get_params():
    """

    """



    content = {

    }

    return Response(content, status=status.HTTP_200_OK)
```

the response

Question 6 - OOP

Below is poorly written class that doesn’t take advantage of the Object oriented principles, principles that are supposed to reduce repeating yourself (DRY) and make code refactoring and mantainance easier. Fix the below function makin use of OOP with DRY principles. None of the methods should be static methods.

```
class TestMath:


    def test_add(x,y):
        x=10
        y=10
```

```
        return x + y

    def test_subtract(x,y):
        x=10
        y=10
        return x - y

    def test_milutiply(x,y):
        x=10
        y=10
        return x * y
```

nb