Question 1 – Dates

```python
from datetime import datetime

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
    """
    date_format = '%Y-%m-%d'
    try:
        datetime.strptime(date, date_format)
        return True
    except ValueError:
        return False

if __name__ == '__main__':
    boool = is_date_format_correct('2000-09-28')
    print(boool)
```

Question 2 - Code flow

```python
for i in range(1,11):
    if i==6:
        continue
    else:
        print(i,end=",")
```

Question 3 - List comprehensions

```python
from datetime import datetime, timedelta

def compute_prev_date(dates_list: list):
    prev_days_list = []
    fmt = '%Y-%m-%d'
    prev_date_format = '%d %b %Y'
    for date in dates_list:
        prev_date = datetime.strptime(date, fmt) - timedelta(1)
        formarted_prev_day = prev_date.strftime(prev_date_format)
        prev_days_list.append(formarted_prev_day)
    return prev_days_list

if __name__ == '__main__':
    list = compute_prev_date(["1999-01-21", "2012-12-30"])
    print(list)
```

Question 4 - Basic exception handling

```python
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():
        try:
            qty = fetch_quantity()
        except Exception as e:
            print(f"Exception: {e}")
            exit()

        try:
            cost = fetch_cost()
        except Exception as e:
            print(f"Exception: {e}")

        try:
            cost_per_quantity = cost / qty
        except Exception as e:
            print(f"Exception: {e}")
            exit()

        return cost_per_quantity

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

Question 5 - Django rest framework

```
from rest_framework import status
from rest_framework.response import Response
from rest_framework.decorators import api_view
@api_view(['GET'])
def get_params(request):

content = {
    'name': request.GET.get('name', ''),
    'surname': request.GET.get('surname', '')
}
return Response(content, status=status.HTTP_200_OK)
```

Question 6 – OOP

```
class TestMath:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def test_add(self):
        return self.x + self.y

    def test_subtract(self):
        return self.x - self.y

    def test_milutiply(self):
        return self.x * self.y
```