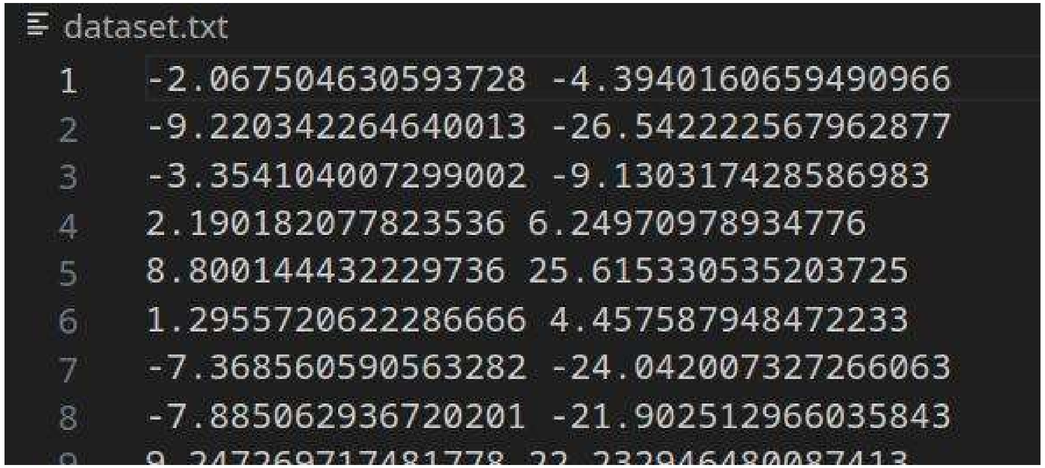
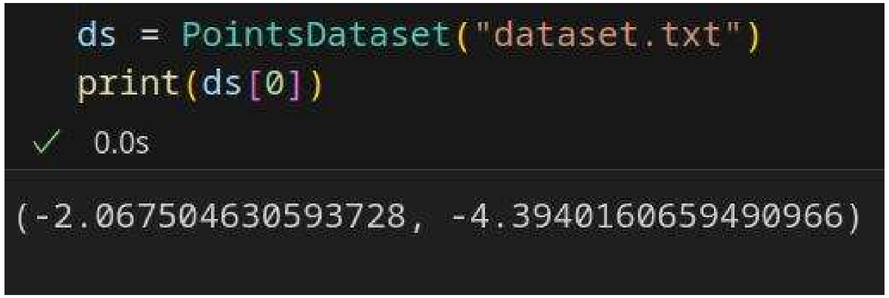
Exercise 1: (1)

Write the **PointsDataset** class that implements the **torch.utils.data.Dataset** interface:

- Reads a txt file in which each line represents a bidimensional data point, with each dimension separated by a space.
- Saves the content of the file in a data structure of your choice
- The **__len__(self)** method should return the number of data points
- The **__getitem__(self, i)** method should return the i-th data point as a tuple.


dataset.txt

```
-2.067504630593728 -4.3940160659490966
-9.220342264640013 -26.542222567962877
-3.354104007299002 -9.130317428586983
2.190182077823536 6.24970978934776
8.800144432229736 25.615330535203725
1.2955720622286666 4.457587948472233
-7.368560590563282 -24.042007327266063
-7.885062936720201 -21.902512966035843
9.247269717481778 22.232946480087413
```



```
ds = PointsDataset("dataset.txt")
print(ds[0])
```

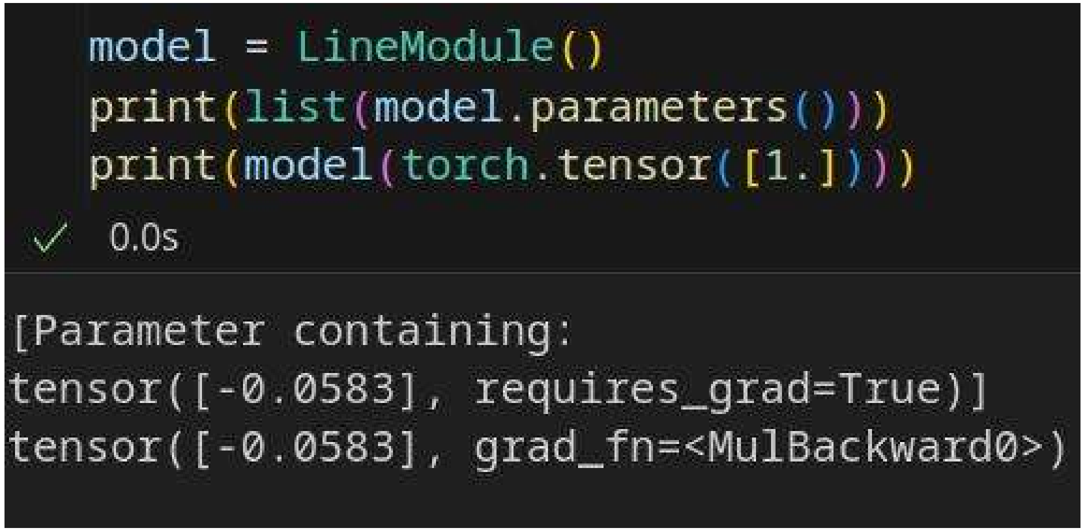
✓ 0.0s

```
(-2.067504630593728, -4.3940160659490966)
```

Exercise 1: (2)

Write the **LineModule** class that implements the torch.nn.Module interface implementing the function $f(x) = wx$.

- LineModule has 1 parameter w
- The **forward(self, x)** method should return wx



```
model = LineModule()
print(list(model.parameters()))
print(model(torch.tensor([1.])))
```

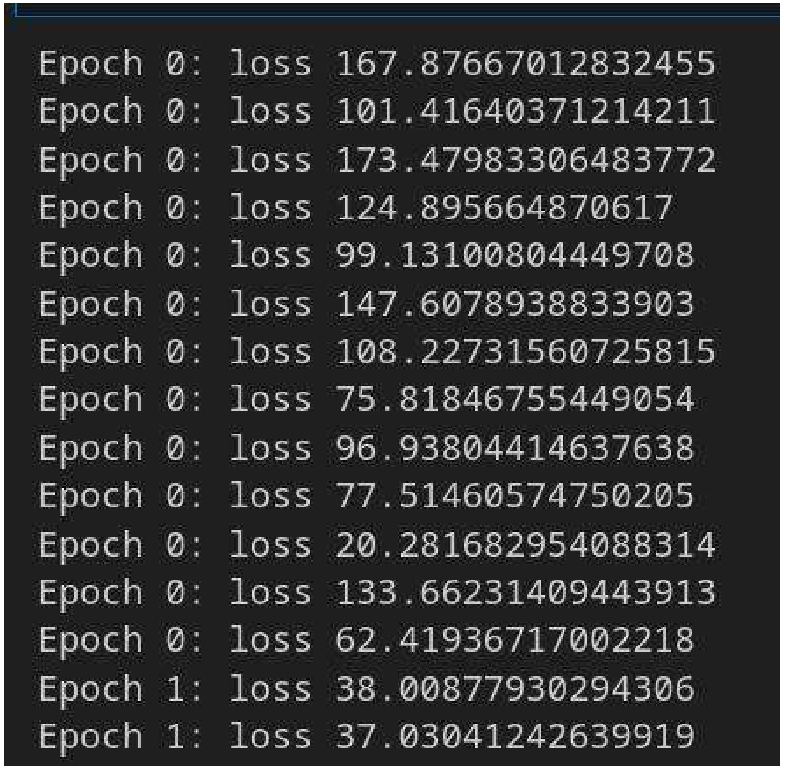
✓ 0.0s

```
[Parameter containing:
tensor([-0.0583], requires_grad=True)]
tensor([-0.0583], grad_fn=<MulBackward0>)
```

Exercise 1: (3)

Write a complete python script that trains **LineModule** to approximate the data in **dataset1.txt**

- Use the **SGD** optimizer from **torch.optim**
- Use the **MSELoss** from **torch.nn**
- Use a batch_size of 8
- Use a learning rate of 0.001
- Train for 1000 epochs



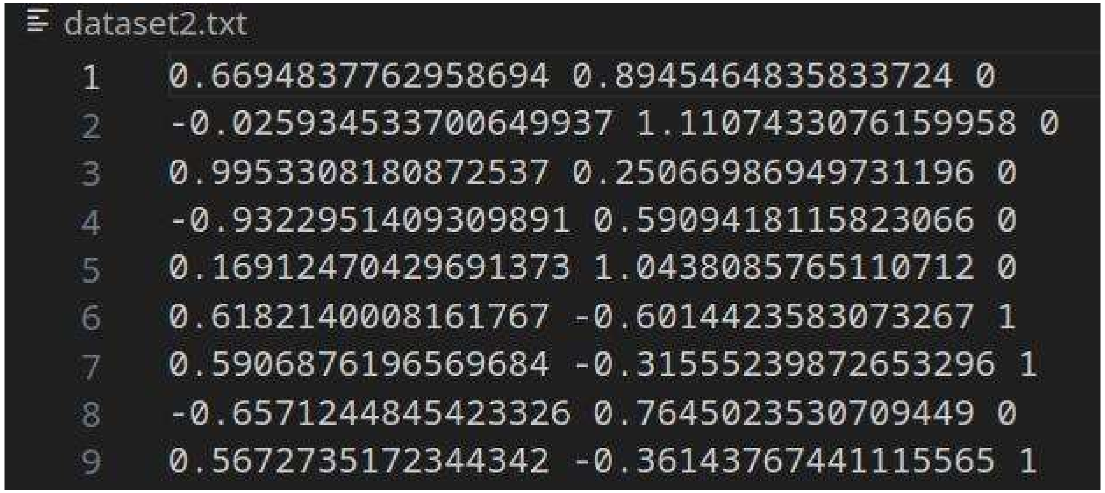
```
Epoch 0: loss 167.87667012832455
Epoch 0: loss 101.41640371214211
Epoch 0: loss 173.47983306483772
Epoch 0: loss 124.895664870617
Epoch 0: loss 99.13100804449708
Epoch 0: loss 147.6078938833903
Epoch 0: loss 108.22731560725815
Epoch 0: loss 75.81846755449054
Epoch 0: loss 96.93804414637638
Epoch 0: loss 77.51460574750205
Epoch 0: loss 20.281682954088314
Epoch 0: loss 133.66231409443913
Epoch 0: loss 62.41936717002218
Epoch 1: loss 38.00877930294306
Epoch 1: loss 37.03041242639919
```

Exercise 2 (hard)

Train a polynomial model in this form:

$$ax^4 + bx^3 + cx^2 + dx + e$$

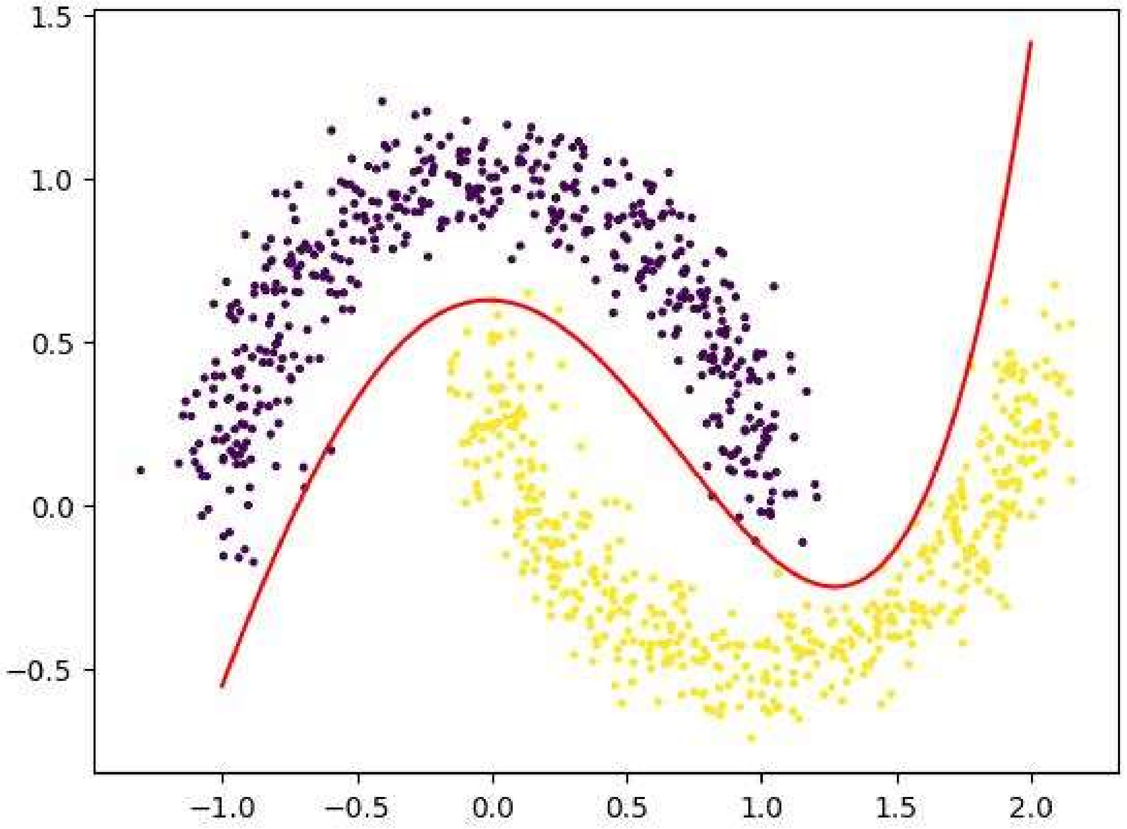
To divide the points of **dataset2.txt** (in the form “x y class” each line)



```
0.6694837762958694 0.8945464835833724 0
-0.025934533700649937 1.1107433076159958 0
0.9953308180872537 0.2506698694973196 0
-0.9322951409309891 0.5909418115823066 0
0.16912470429691373 1.0438085765110712 0
0.6182140008161767 -0.6014423583073267 1
0.5906876196569684 -0.31555239872653296 1
-0.6571244845423326 0.7645023530709449 0
0.5672735172344342 -0.36143767441115565 1
```

Exercise 2 (hard)

- The polynomial model takes x as input and gives $\overline{y}$ as output
- Given an (x, y) pair from the dataset you can compute $\overline{y}$ = model(x)
- If y is **above** the line (so y > $\overline{y}$) and **it should be** (class 0) than everything is **ok**
- if y is **above** the line and **it should be below it** (class 1) than you should compute an **error**
- And so on…


1.5
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0