Import same libraries

```
import pickle
import numpy as np
import  os
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
directory_path = '/content/drive/My Drive/Colab Notebooks/'
```

```
OHE_sex = pickle.load(open(directory_path + 'OHE_sex.pkl', 'rb'))
OHE_embarked = pickle.load(open(directory_path + 'OHE_embarked.pkl', 'rb'))
DTC = pickle.load(open(directory_path + 'DTC.pkl', 'rb'))
```

Here I just created objects to store the files after loadiing the dumped files using load,
So we did same procedure just instead of 'dump' we did 'load' and instead of 'wb' we did 'rb'.

```
test_input = np.array([2, 'male', 31.0, 0, 0, 10.5, 'S'],dtype=object).reshape(1,7)
```

Created object test_input and loaded input as per format of our X_test_transformed in our previous file, declared data type as object and reshape our array as (1, 7)
Here, in our code:
Pclass --> 2
Sex -->male
Age -->31.0
SibSp --> 0
Parch --> 0
Fare --> 10.5 Dollars
Embarked --> S

Now direcly use models we loaded using pickle.

```
test_input
```

```
array([[2, 'male', 31.0, 0, 0, 10.5, 'S']], dtype=object)
```

```
test_input_sex = OHE_sex.transform(test_input[:, 1].reshape(1, 1))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but OneHotEncoder was fitted with feature names
  warnings.warn(
```

```
test_input_sex
```

```
array([[0., 1.]])
```

```
test_input_embarked = OHE_embarked.transform(test_input[:, -1].reshape(1, 1))
```

```
test_input_embarked
```

```
array([[0., 0., 1.]])
```

```
test_input_age = test_input[:, 2].reshape(1, 1)
```

```
test_input_age
```

```
array([[31.0]], dtype=object)
```

```
test_input_transformed = np.concatenate((test_input[:,[0, 3, 4, 5]],
                                         test_input_age,
                                         test_input_sex,
                                         test_input_embarked), axis=1)
```

```
test_input_transformed
```

```
array([[2, 0, 0, 10.5, 31.0, 0.0, 1.0, 0.0, 0.0, 1.0]], dtype=object)
```

```
test_input_transformed.shape
```

```
(1, 10)
```

```
test_input_sex --> 0, 1 means it is male
                   1, 0 means it is female
test_input_embarked --> 0, 0, 1 means 'S'
                        0, 1, 0 means 'C'
                        1, 0, 0 means 'Q'
```

Then, I created object test_input_transformed where we concatenate the results same format as X_train_transformed

np.concatenate((test_input[:, [0, 3, 4, 5]], test_input_age, test_input_sex, test_input_embarked), axis = 1)

Then predicted whether passenger will survive or not.

```
DTC.predict(test_input_transformed)
```

```
array([1])
```