Simple earthquake prediction model using Python:

1. **Data Collection**:
   - Gather earthquake data from reliable sources such as the United States Geological Survey (USGS) or the European-Mediterranean Seismological Centre (EMSC).

2. **Data Preprocessing**:
   - Clean and preprocess the data, removing any irrelevant information and dealing with missing values.

3. **Feature Engineering**:
   - Extract relevant features from the data, such as earthquake location, depth, magnitude, and time.

4. **Data Splitting**:
   - Split the data into training and testing sets. The training data will be used to train the model, while the testing data will be used to evaluate its performance.

5. **Model Selection**:
   - Choose a machine learning model. For a basic example, you can use a logistic regression model, which can classify data into two classes: earthquake (1) and non-earthquake (0).

6. **Model Training**:
   - Train your selected model using the training data. In Python, you can use libraries like scikit-learn for this purpose.

7. **Model Evaluation**:
   - Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1-score. You can also use ROC-AUC if you are working with a binary classification model.

Here's a simple Python code example for creating a basic earthquake prediction model using scikit-learn:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load earthquake data
data = pd.read_csv('database.csv')

# Preprocessing and feature engineering (remove irrelevant columns and preprocess data)
```

```
# Split the data into features (X) and the target (y)
X = data.drop('is_earthquake', axis=1)
y = data['is_earthquake']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test, y_pred))
```