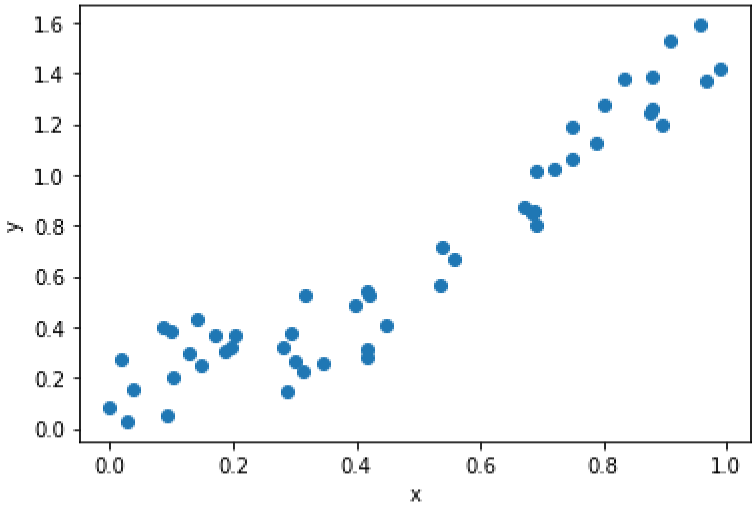
In [2]:

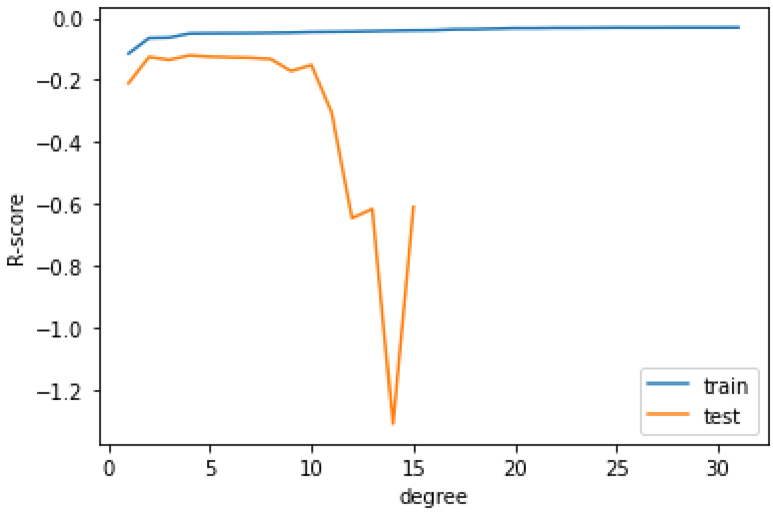
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(1)
X = np.random.rand(50,1)
y = 0.7*(X**5) - \
    2.1*(X**4) + \
    2.3*(X**3) + \
    0.2*(X**2) + \
    0.3* X + \
    0.4*np.random.rand(50,1) # no data in world is perfect
fig = plt.figure()
plt.scatter(X, y)
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```



In [8]:

```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10)

from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline

degrees = 32# number of data-points
train_scores = []
test_scores = []

scaler = StandardScaler()

for degree in range(1, degrees):
    fold_train_scores = []
    fold_test_scores = []
    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        polyreg_scaled = make_pipeline(PolynomialFeatures(degree), scaler, LinearReg
        polyreg_scaled.fit(X_train, y_train)
        train_score = polyreg_scaled.score(X_train, y_train)
        test_score = polyreg_scaled.score(X_test, y_test)
        fold_train_scores.append(train_score)
        fold_test_scores.append(test_score)
    train_score = np.mean(fold_train_scores)
    test_score = np.mean(fold_test_scores)
    train_scores.append(train_score)
    test_scores.append(test_score)

plt.figure()
plt.plot(list(range(1, 32)), np.log(train_scores), label="train")
plt.plot(list(range(1, 32)), np.log(test_scores), label="test")
plt.legend(loc='lower right')
plt.xlabel("degree")
plt.ylabel("R-score")
plt.show()
```

```
<ipython-input-8-b30fd2528fac>:34: RuntimeWarning: invalid value encou
ntered in log
  plt.plot(list(range(1, 32)), np.log(test_scores), label="test")
```



In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [10]:

```
# from sklearn.linear_model import ElasticNet, Ridge, Lasso, LinearRegression
```

In [ ]:

```

```

In [11]:

```
def sigmoid(x):
    return 1/(1 + np.exp(-x))
```

In [12]:

```
sigmoid(0)
```

Out[12]:

```
0.5
```

In [13]:

```
sigmoid(10)
```

Out[13]:

```
0.9999546021312976
```

In [14]:

```
sigmoid(-15)
```

Out[14]:

```
3.059022269256247e-07
```

In [19]:

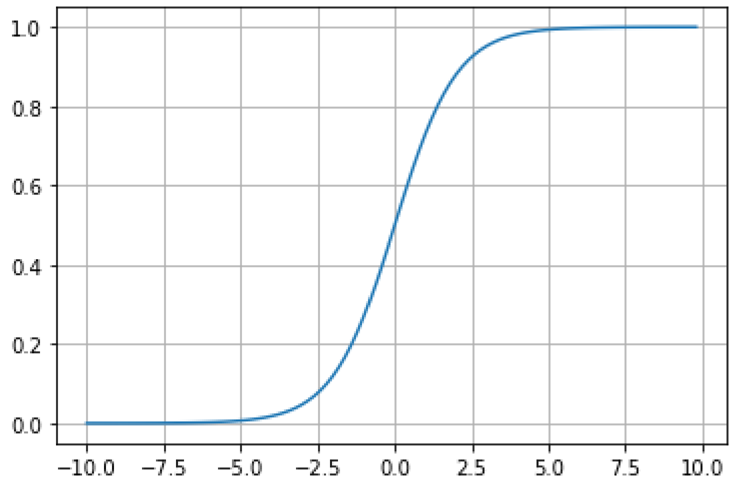
```
x = np.arange(-10, 10, 0.2)
```

In [21]:

```
y = sigmoid(x)
```

In [24]:

```
plt.plot(x, y)
plt.grid('on')
```



In [25]:

```
sigmoid(5)
```

Out[25]:

```
0.9933071490757153
```

In [27]:

```
sigmoid(7)
```

Out[27]:

```
0.9990889488055994
```

In [ ]: