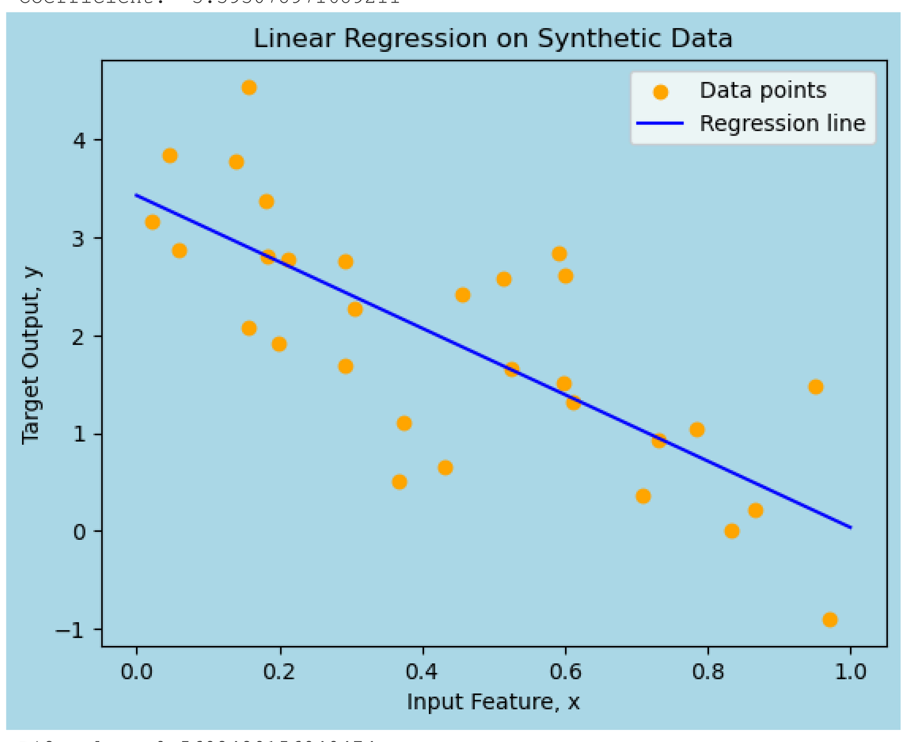
Creating a Linear Regression Model with synthetic data

In [33]:
```python
#import packages
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt


#create synthetic data randomly then define randomly uniformly for 30 points
np.random.seed(42)
x = np.random.uniform(0, 1, 30)
epsilon = np.random.normal(0, 1, 30)
y = 3 - 2*x + epsilon

# fit
x = x.reshape(-1, 1)

# create the model for lin regression
model = LinearRegression()
model.fit(x, y)

# Get the intercept and coefficient
intercept = model.intercept_
coefficient = model.coef_[0]

print(f"Intercept: {intercept}")
print(f"Coefficient: {coefficient}")

# Plot the original data points
fig, ax = plt.subplots()
plt.scatter(x, y, color='orange', label='Data points')
fig.patch.set_facecolor('lightblue')  # Set background color for the plot
ax.set_facecolor('lightblue')  # Set background color for the plot area

# Plot the regression line
x_line = np.linspace(0, 1, 100).reshape(-1, 1)
y_line = model.predict(x_line)
plt.plot(x_line, y_line, color='blue', label='Regression line')

# Labels and title
plt.xlabel('Input Feature, x')
plt.ylabel('Target Output, y')
plt.legend()
plt.title('Linear Regression on Synthetic Data')

plt.show()

r_squared = model.score(x, y)
print(f"R^2 value: {r_squared}")
```

```
Intercept: 3.4289712779012214
Coefficient: -3.393078971689211
```




```
R^2 value: 0.5692429156949474
```

Project Summary

Using historical data from 2010 to 2018 from Jeep Wrangler, a ridge regression model was developed to predict monthly sales. Features such as the year, unemployment rate, Wrangler-related Google search queries, and CPI-related indices were used to create the model. In the ridge regression model the regularization parameter λ was tuned, and then the optimized model was used to predict future sales based on new feature values.

The Python packages used were pandas, Numpy, Scikit-Learn (sklearn), matplotlib.

Overview

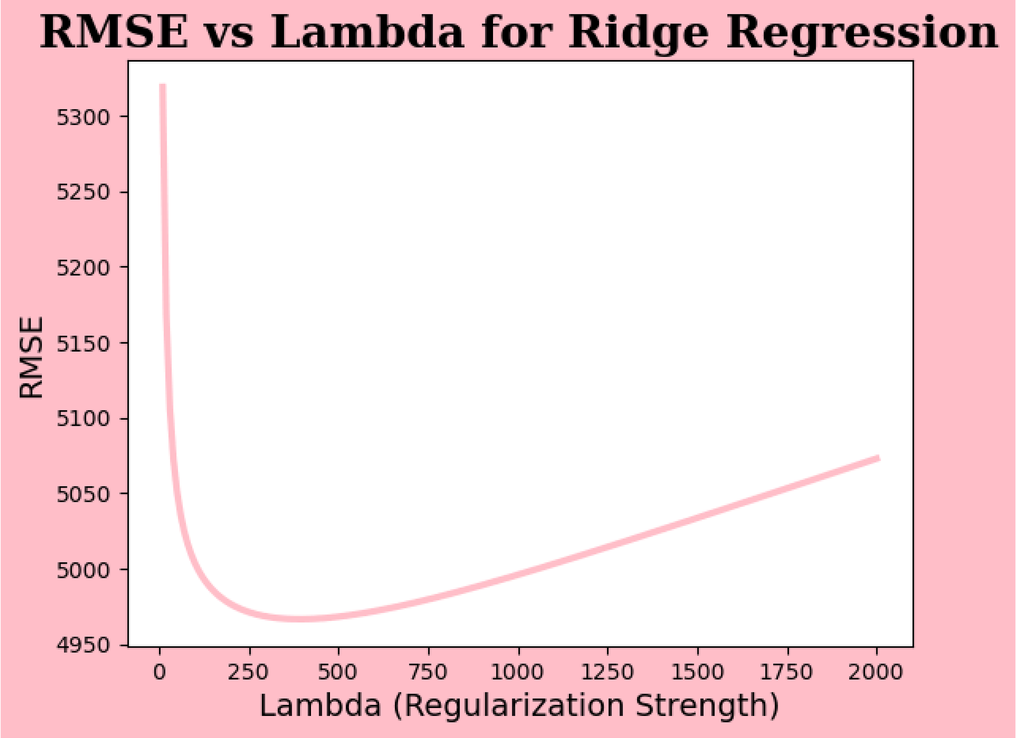
First, the data was loaded and prepared for ridge regression. The data was split into training and test sets based on specific timeframes (2010-2017 for training, 2018 for testing). Regularization prevents overfitting in linear regression and the regularization parameter λ was adjusted. The model's performance was measured using RMSE) to quantify the model's performance and λ was selected to minimize the RMSE. This demostrates how the machine learning can be utilized in predicting future sales with reasonable accuracy. This is helpful information for companies such as Jeep so they can make better decisions for resource and sales in the future.

In [12]:
```python
import pandas as pd

# Load the data from Wrangler
data = pd.read_csv('wrangler2018.csv')

# show a table of the data
print(data.head())
```

```
   Month.Numeric Month.Factor  Year  Wrangler.Sales  Elantra.Sales  \
0              1      January  2010            4888           7690   
1              2     February  2010            5967           7966   
2              3        March  2010            8410           8225   
3              4        April  2010            8327           9657   
4              5          May  2010            9634           9781   

   Unemployment.Rate  Wrangler.Queries  Elantra.Queries  CPI.All  CPI.Energy  
0                9.8                32                9  217.488     212.807  
1                9.8                35               10  217.281     209.624  
2                9.9                35               10  217.353     209.326  
3                9.9                38               10  217.403     209.219  
4                9.6                38               11  217.290     206.631  
```

In [17]:
```python
# I want to separate vectors of the features that most related to our model and also the target variable, y. In this case, it would be the sales.
X = data[['Year', 'Unemployment.Rate', 'Wrangler.Queries', 'CPI.Energy', 'CPI.All']]
y = data['Wrangler.Sales']

# Now, I want to do some splitting of data to prepare of machine learning. Creating a training (2010-2017) and test set data set (2018)
train_data = data[data['Year'] < 2018]
test_data = data[data['Year'] == 2018]

# Define the training and test features (X) and target (y)
X_train = train_data[['Year', 'Unemployment.Rate', 'Wrangler.Queries', 'CPI.Energy', 'CPI.All']]
y_train = train_data['Wrangler.Sales']

X_test = test_data[['Year', 'Unemployment.Rate', 'Wrangler.Queries', 'CPI.Energy', 'CPI.All']]
y_test = test_data['Wrangler.Sales']

# I want to see how the model performs with different lambda values
lambdas = np.arange(10, 2001, 10)
rmse_values = []

# ridge regression model for each lambda then i need to  calculate the rate mean square estimate on the test data
for lam in lambdas:
    ridge_model = Ridge(alpha=lam)
    ridge_model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = ridge_model.predict(X_test)

    # Calculate RMSE
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    rmse_values.append(rmse)

# # Plot RMSE against lambda values
# plt.plot(lambdas, rmse_values)
# plt.xlabel('Lambda (Regularization Strength)')
# plt.ylabel('RMSE')
# plt.title('RMSE vs Lambda for Ridge Regression')
# plt.show()

# Identify and print the best lambda (which minimizes RMSE)
best_lambda = lambdas[np.argmin(rmse_values)]
print(f"Best Lambda: {best_lambda}")

# If desired, print the minimum RMSE value as well
best_rmse = min(rmse_values)
print(f"Best RMSE: {best_rmse}")

# Plot the figure in pink for fun
plt.figure(facecolor='pink')
plt.plot(lambdas, rmse_values, color='pink', linewidth=3)

# Customize the title, font, and size
plt.title('RMSE vs Lambda for Ridge Regression', fontsize=20, fontweight='bold', fontname='Serif')

# Customize other elements
plt.xlabel('Lambda (Regularization Strength)', fontsize=14)
plt.ylabel('RMSE', fontsize=14)

# Show the plot with changes
plt.show()
```

```
Best Lambda: 390
Best RMSE: 4966.568479664433
```



In [18]:
```python
# Retrain the ridge regression model with the best lambda (from Part (c))
ridge_model = Ridge(alpha=best_lambda)
ridge_model.fit(X_train, y_train)

# Define the new data point for which we want to predict Wrangler.Sales
new_data = np.array([[2018, 4, 79, 220, 249]])

# Predict Wrangler Sales
```

```python
predicted_sales = ridge_model.predict(new_data)
print(f"Predicted Wrangler Sales: {predicted_sales[0]}")
```

```
Predicted Wrangler Sales: 16993.910585035002
```

```
C:\Users\jessi\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but Ridge was fitted with feature names
  warnings.warn(
```