In [7]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [8]:
```
df=pd.read_csv('canada_per_capita_income.csv')
```

In [9]:
```
df.head()
```

Out[9]:

	year	per capita income (US$)
0	1970	3399.299037
1	1971	3768.297935
2	1972	4251.175484
3	1973	4804.463248
4	1974	5576.514583

In [10]:
```
df.shape
```

Out[10]:
```
(47, 2)
```

In [11]:
```
missing_values=df.isnull().sum()
print(missing_values)
```

```
year                        0
per capita income (US$)     0
dtype: int64
```

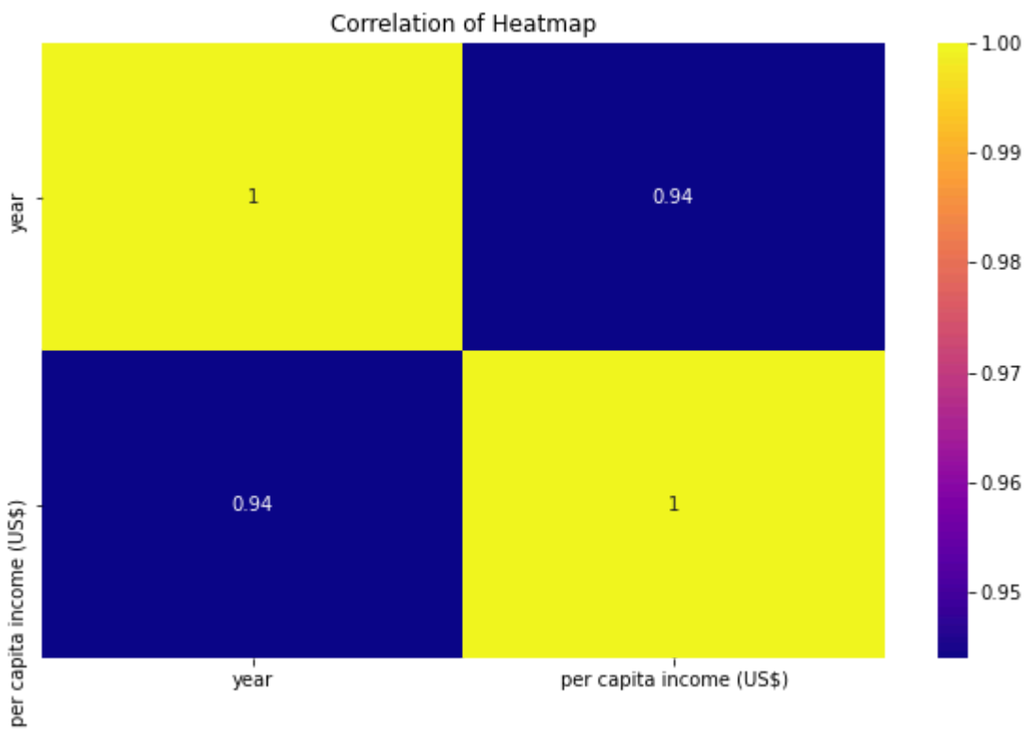
In [12]:
```
df.describe()
```

Out[12]:

	year	per capita income (US$)
count	47.000000	47.000000
mean	1993.000000	18920.137063
std	13.711309	12034.679438
min	1970.000000	3399.299037
25%	1981.500000	9526.914515
50%	1993.000000	16426.725480
75%	2004.500000	27458.601420
max	2016.000000	42676.468370

In [13]:
```
df_final=df
```

In [14]:
```
plt.figure(figsize=(10,6))
sns.heatmap(df_final.corr(),annot=True,cmap="plasma")
plt.title('Correlation of Heatmap' )
plt.show()
```



In [15]:
```
x=df_final.drop('per capita income (US$)',axis=1)
y=df_final['per capita income (US$)']
```

In [16]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=42)
```

In [17]:
```
regression=LinearRegression()
```

In [18]:
```
regression.fit(x_train,y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
regression.score(x_test,y_test)
```

Out[19]:
```
0.8751771396846304
```

In [20]:
```
regression.coef_
```

Out[20]:
```
array([815.14251301])
```

In [21]:
```
regression.intercept_
```

Out[21]:
```
-1605560.1987964248
```

In [22]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [23]:
```
reg_pred=regression.predict(x_test)
```

In [24]:
```
import numpy as np
```

In [25]:
```
print("Mean_squared_error:", mean_squared_error(y_test,reg_pred))
print("Mean_absolute_error:", mean_absolute_error(y_test,reg_pred))
print("Squared_error:",(np.sqrt(mean_squared_error(y_test,reg_pred))))
```

```
Mean_squared_error: 15147815.5477862
Mean_absolute_error: 3240.91399747583
Squared_error: 3892.0194690913613
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: