In [ ]:

```
import pandas as pd
import random
import time
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [ ]:

```
# Selection sort in Python
# # time complexity O(n*n)
#sorting by finding min_index
def selectionSort(array, size):

 for ind in range(size):
  min_index = ind

  for j in range(ind + 1, size):
   # select the minimum element in every iteration
   if array[j] < array[min_index]:
    min_index = j
  # swapping the elements to sort the array
  (array[ind], array[min_index]) = (array[min_index], array[ind])
# def end
```

In [ ]:

```
# creating a data frame object
df = pd.DataFrame(columns = ["length", "time_ns","org_arr","arr"])


n=2000 # Length of the array  range will be 2 to n-1
for i in range(2,n):
    r_sample = random.sample(range(10,i+1000), i)
    temp_df = {'length':i,'org_arr':r_sample}
    df=df.append(temp_df,ignore_index=True)
```

In [ ]:

```
df
```

In [ ]:

```
index_len=len(df.index)

for d in range(index_len):

    temp_arr=(df.loc[d].at['org_arr']).copy()
    arr_size=(df.iloc[d]['length'])

    start_counter_ns = time.perf_counter_ns()
    selectionSort(temp_arr, arr_size)
    end_counter_ns = time.perf_counter_ns()
    timer_ns = end_counter_ns - start_counter_ns
    df.at[d, 'time_ns'] = timer_ns
    df.at[d,'arr']=temp_arr
```

In [ ]:

```
df.head(50)
```

In [ ]:

```
# using dictionary to convert specific columns
convert_dict = {'length': int,'time_ns': int}
```

```
df = df.astype(convert_dict)


x=df.length
y=df.time_ns
#create basic scatterplot
plt.plot(x, y, '.')

#obtain m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(x, y, 1)

#add linear regression line to scatterplot
plt.plot(x, m*x+b)

print(m,b)
```