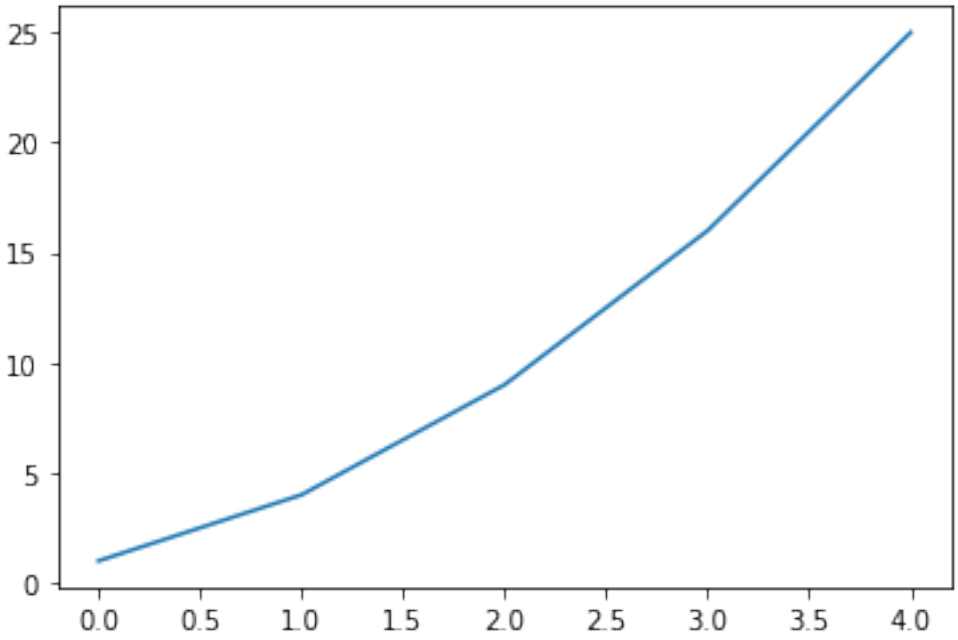
matplotlib

December 16, 2021

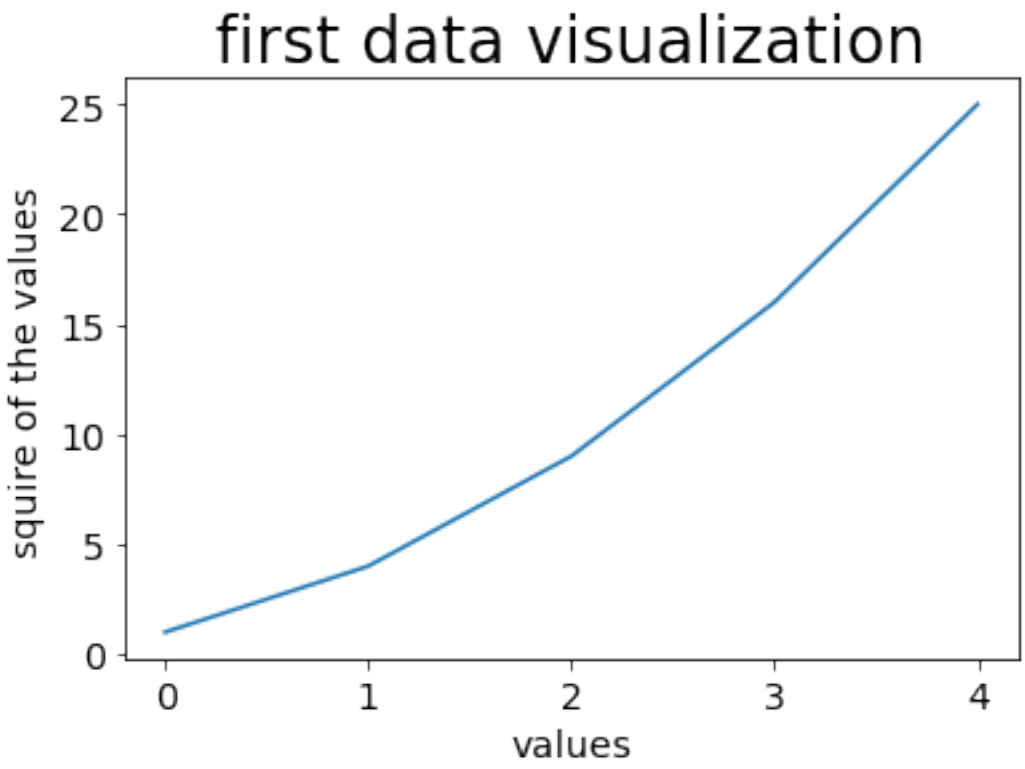
[3]:
```
#plotting a  simple line graph
import matplotlib.pyplot as plt
data = [1, 4, 9, 16, 25]
fig, ax = plt.subplots()
ax.plot(data)
plt.show()
```



[6]:
```
#changing label types and line thickness
import matplotlib.pyplot as plt
data = [1,4,9,16,25]
fig, ax = plt.subplots()
ax.plot(data)
ax.set_title('first data visualization', fontsize = 24)
ax.set_xlabel('values', fontsize = 14)
ax.set_ylabel('squire of the values', fontsize=14)
ax.tick_params(axis ='both', labelsize = 14)
```

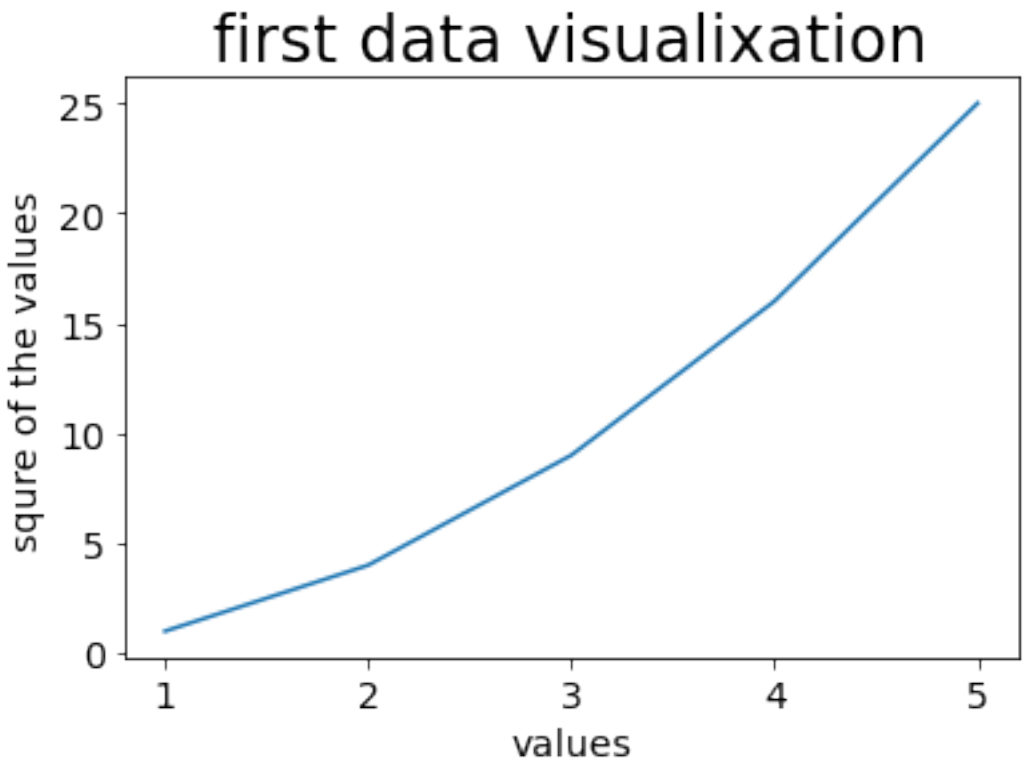
```
plt.show
```

[6]: <function matplotlib.pyplot.show(close=None, block=None)>



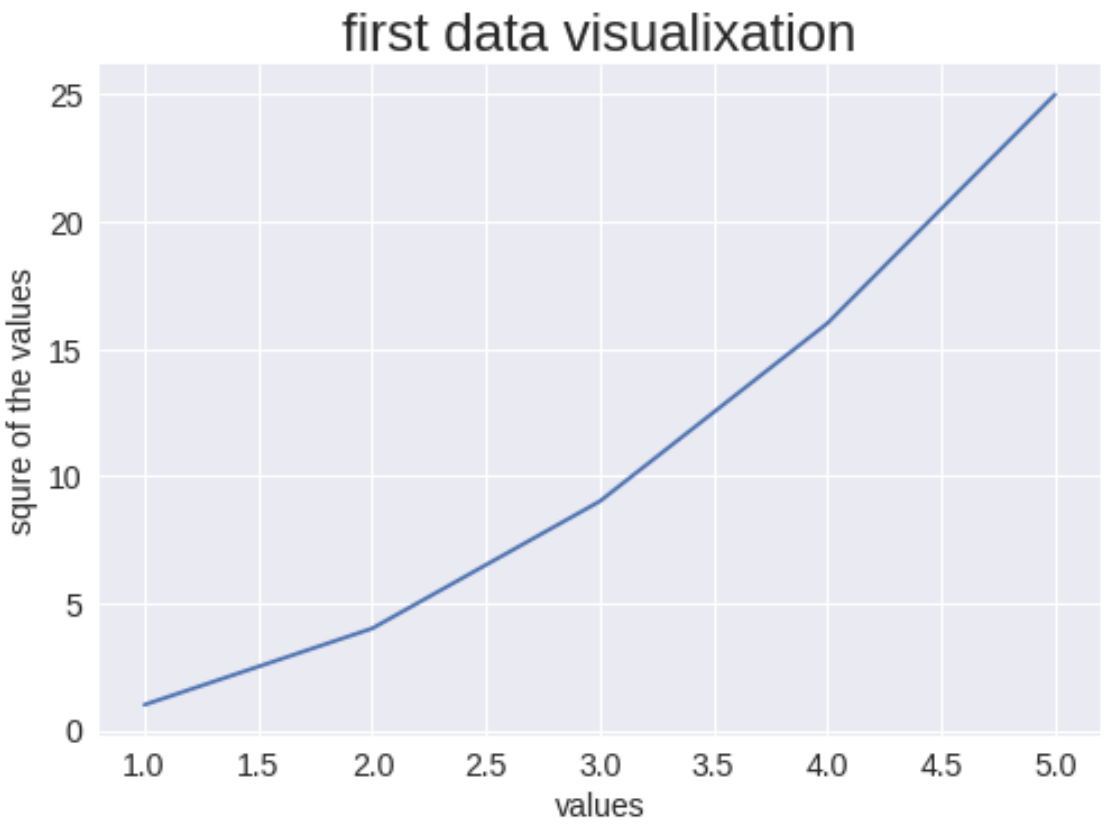
[7]:
```
#correcting the plot
import matplotlib.pyplot as plt
input_values = [1,2,3,4,5]
data = [1,4,9,16,25]
fig, ax = plt.subplots()
ax.plot(input_values, data)
ax.set_title('first data visualixation', fontsize = 24)
ax.set_xlabel('values', fontsize= 14)
ax.set_ylabel('squre of the values',fontsize = 14)
ax.tick_params(axis = 'both', labelsize = 14)
plt.show
```

[7]: <function matplotlib.pyplot.show(close=None, block=None)>



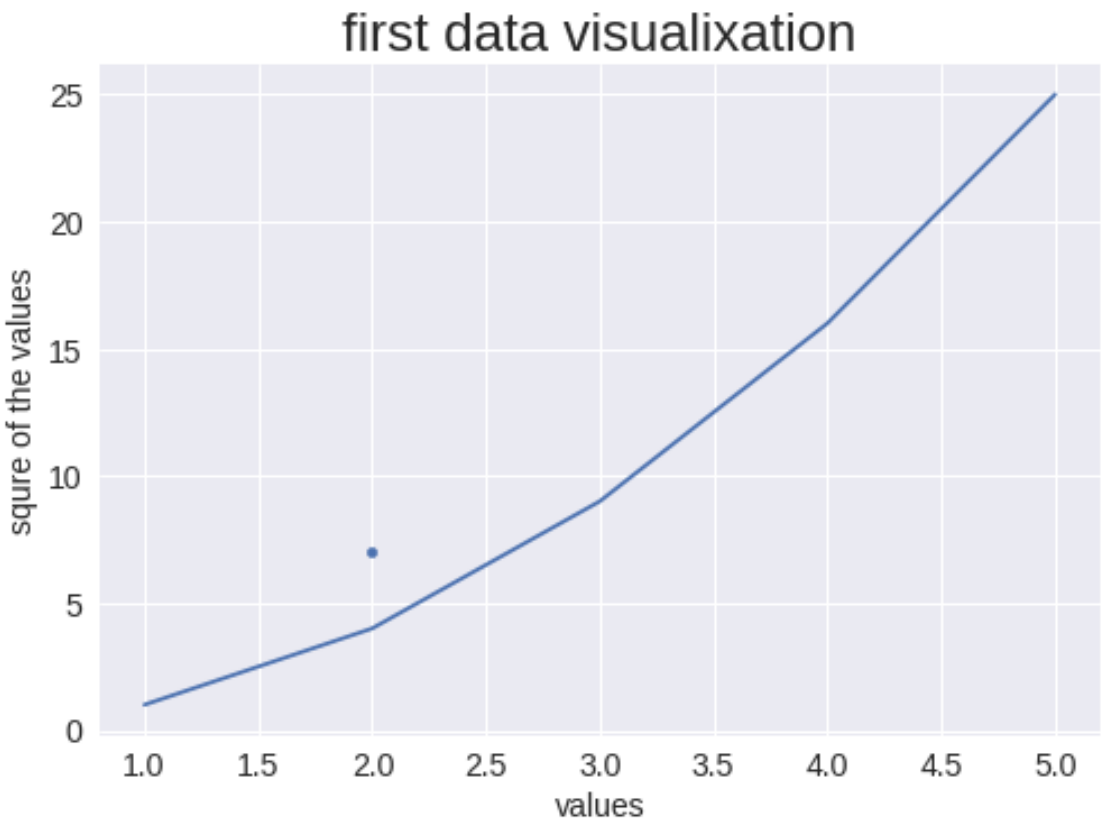
[11]:
```
#using build-in styles
plt.style.available

import matplotlib.pyplot as plt
input_values = [1,2,3,4,5]
data = [1,4,9,16,25]
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(input_values, data)
ax.set_title('first data visualixation', fontsize = 24)
ax.set_xlabel('values', fontsize= 14)
ax.set_ylabel('squre of the values',fontsize = 14)
ax.tick_params(axis = 'both', labelsize = 14)
plt.show
```

[11]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



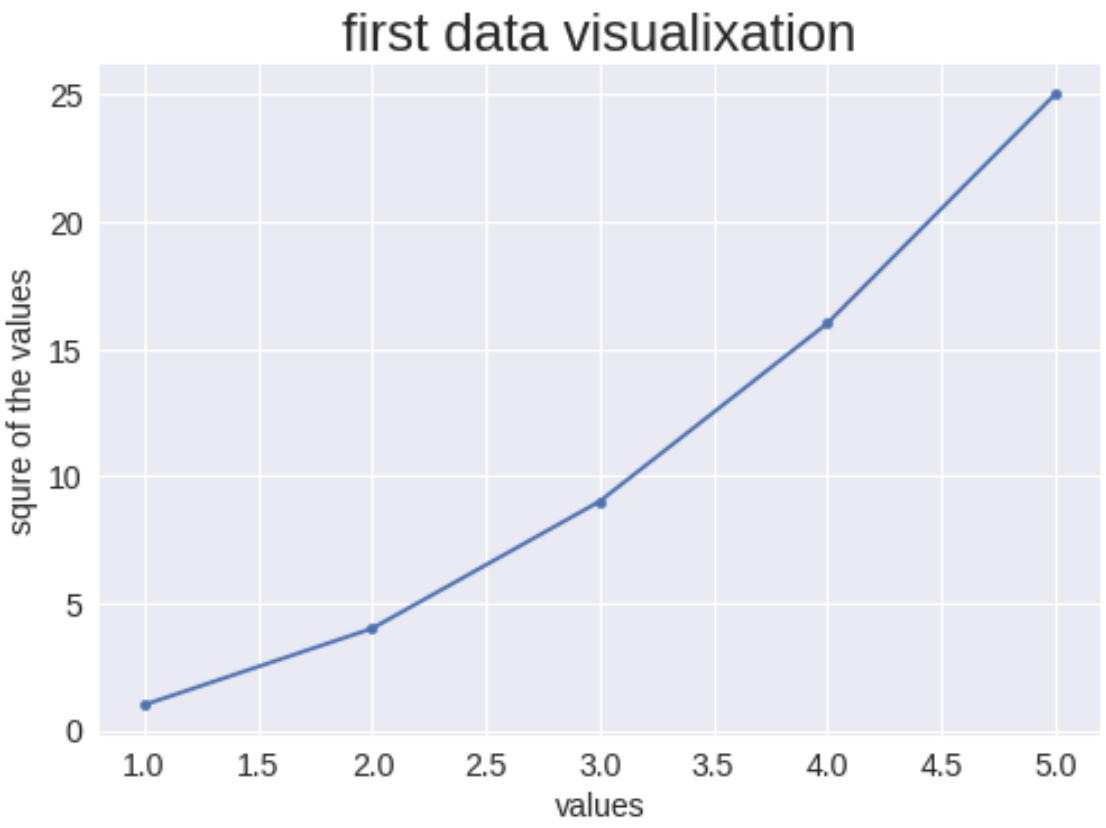
[13]:
```
#plottig and stagging a single point usinng scatter
import matplotlib.pyplot as plt
input_values = [1,2,3,4,5]
data = [1,4,9,16,25]
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(input_values, data)
ax.set_title('first data visualixation', fontsize = 24)
ax.set_xlabel('values', fontsize= 14)
ax.set_ylabel('squre of the values',fontsize = 14)
ax.tick_params(axis = 'both', labelsize = 14)
plt.scatter(2,4, s=20)
```

[13]: <matplotlib.collections.PathCollection at 0x7fd566267a30>



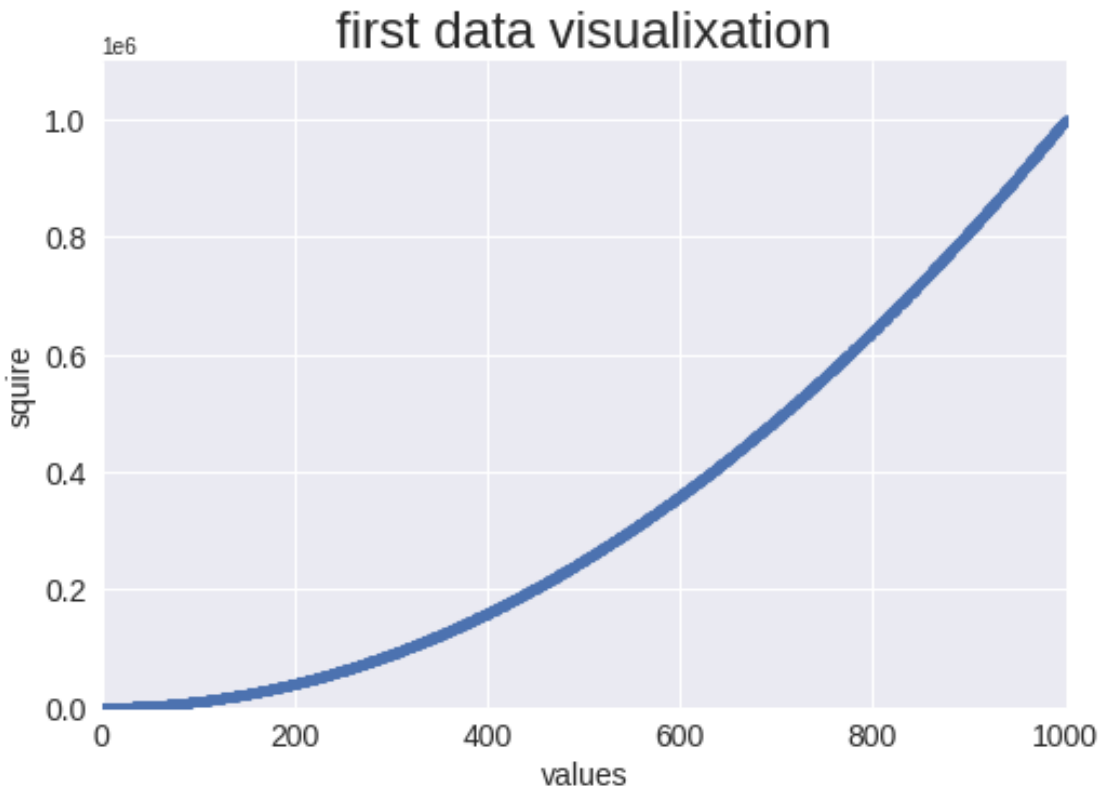
[14]:
```
#plottig and stagging a multiple points usinng scatter
import matplotlib.pyplot as plt
input_values = [1,2,3,4,5]
data = [1,4,9,16,25]
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(input_values, data)
ax.set_title('first data visualixation', fontsize = 24)
ax.set_xlabel('values', fontsize= 14)
ax.set_ylabel('squre of the values',fontsize = 14)
ax.tick_params(axis = 'both', labelsize = 14)
plt.scatter(input_values, data, s=20)
```

[14]: <matplotlib.collections.PathCollection at 0x7fd5664748b0>



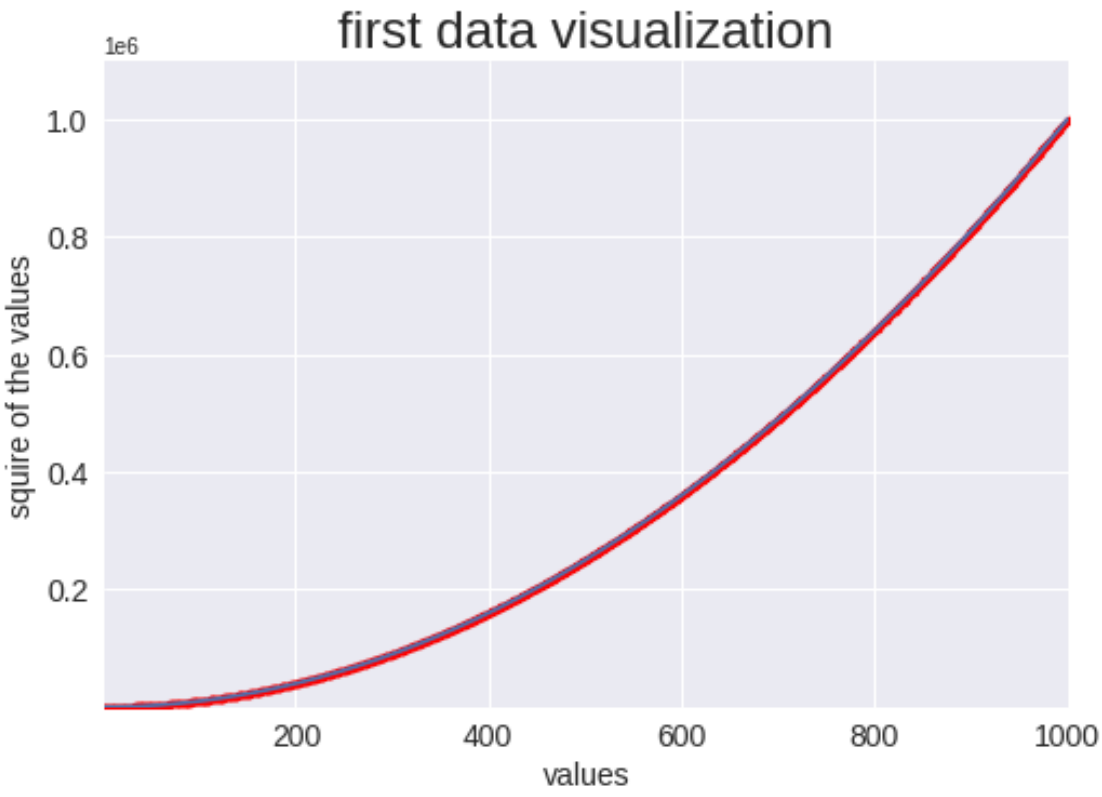
[21]:
```
#culculating the data automatically
import matplotlib.pyplot as plt
x_axis = range(1,1001)
y_axis = [x**2 for x in x_axis ]
plt.style.use('seaborn')
fig,ax = plt.subplots()
ax.plot(x_axis, y_axis)
ax.set_title('first data visualixation', fontsize = 24)
ax.set_xlabel('values', fontsize = 14)
ax.set_ylabel('squire', fontsize = 14)
ax.tick_params(axis = 'both', labelsize = 14)
plt.scatter(x_axis, y_axis, s = 20)
ax.axis([0,1001, 0, 1100000])
plt.show
```

[21]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

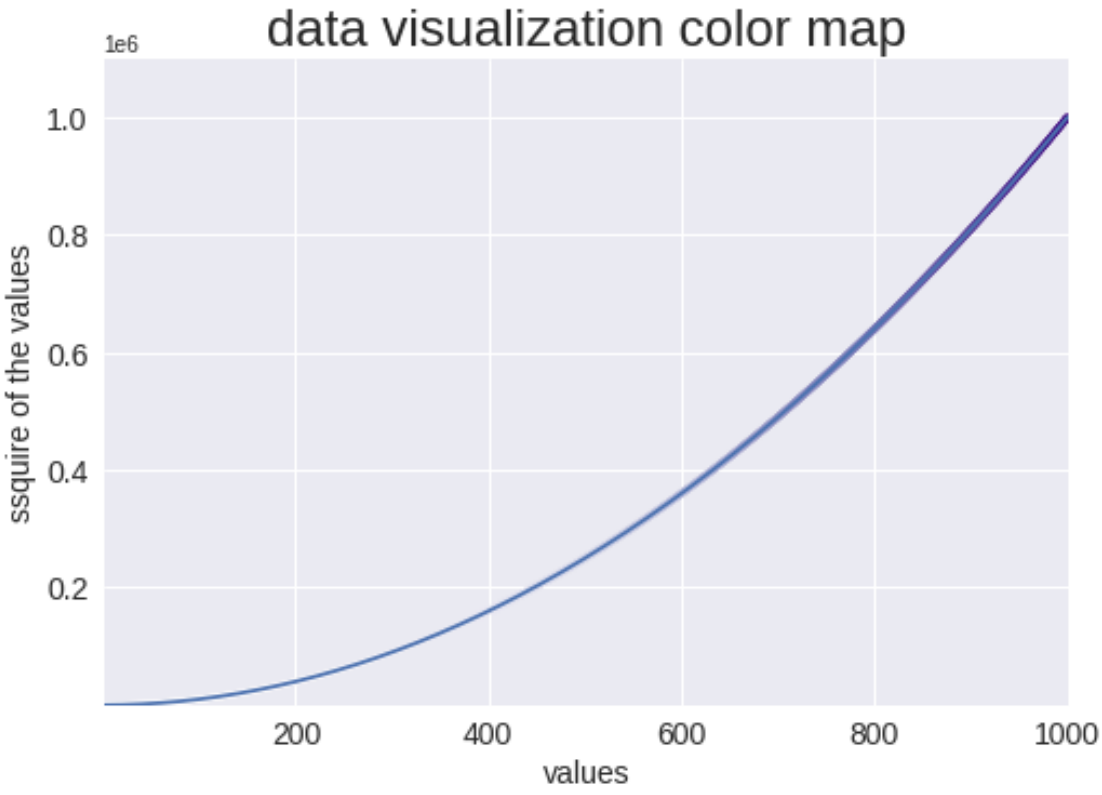


[7]:
```
#difining custom colors
import matplotlib.pyplot as plt

x_axis = range(1 ,1001)
y_axis  = [x ** 2 for x in x_axis]
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(x_axis, y_axis)
ax.set_title('first data visualization', fontsize=24)
ax.set_xlabel('values', fontsize=14)
ax.set_ylabel('squire of the values', fontsize=14)
ax.tick_params(axis= 'both', labelsize=14)
ax.scatter(x_axis, y_axis, c = 'red', s = 10)
ax.axis([1, 1001, 1, 1100000])
```

[7]:
```
(1.0, 1001.0, 1.0, 1100000.0)
```

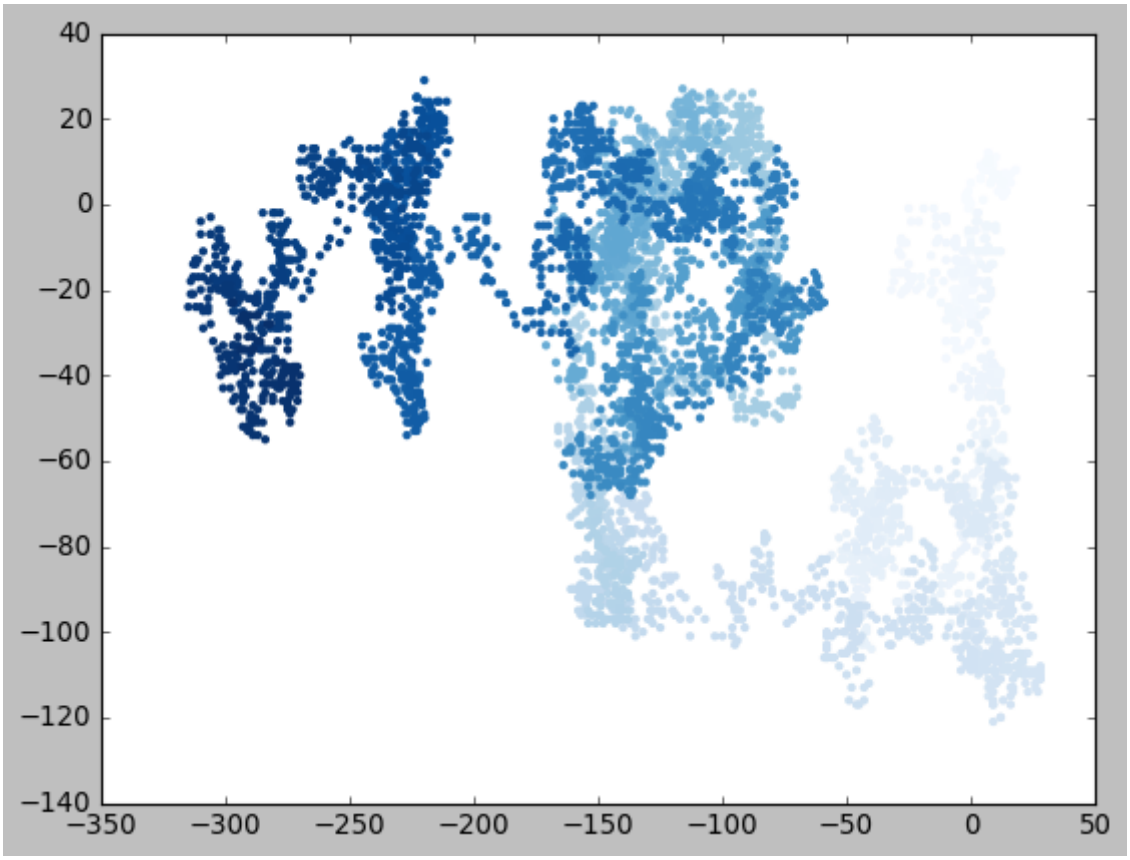


[13]:
```
#ussing color map
import matplotlib.pyplot as plt
x_axis = range(1,1001)
y_axis = [x**2 for x in x_axis]
plt.style.use('seaborn')
fig ,ax = plt.subplots()
ax.plot(x_axis, y_axis)
ax.set_title('data visualization color map', fontsize = 24)
ax.set_xlabel('values', fontsize=14)
ax.set_ylabel('ssquire of the values', fontsize=14)
ax.tick_params(axis='both', labelsize=14)
ax.scatter(x_axis, y_axis, c= y_axis,cmap = plt.cm.Purples , s=10)
ax.axis([1,1001, 1,1100000])
plt.show()
```

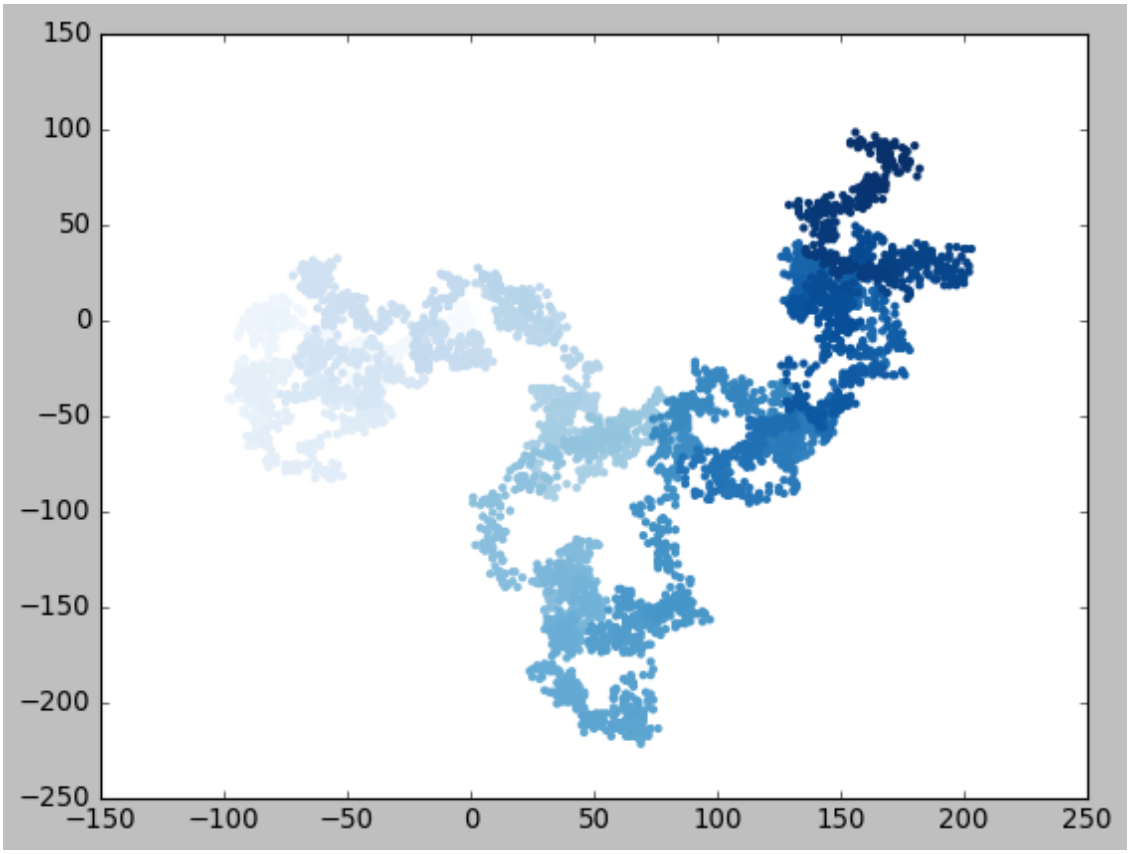


[1]:
```python
#random walk project
from random import choice
import matplotlib.pyplot as plt

class random_walk:
    def __init__(self, steps_num = 5000):
        self.steps_num = steps_num
        self.x_values = [0]
        self.y_values = [0]

    def walk(self):
        while len(self.x_values)< self.steps_num:
            x_direction = choice([1,-1])
            x_distance = choice([0,1,2,3,4])
            x_step = x_direction * x_distance

            y_direction = choice([1,-1])
            y_distance = choice([0,1,2,3,4])
            y_step = y_direction * y_distance

            if self.x_values and self.y_values == 0:
```

```
                continue

            x = self.x_values[-1] + x_step
            y =self.y_values[-1] + y_step

            self.x_values.append(x)
            self.y_values.append(y)
while True:
    rw = random_walk()
    rw.walk()
    plt.style.use('classic')
    fig, ax = plt.subplots()
    ax.plot()
    point_number = range(rw.steps_num)
    ax.scatter(rw.x_values, rw.y_values, c = point_number, cmap = plt.cm.Blues,␣
 ↪edgecolors= 'none', s = 15)
    plt.show()
    keep_running = input('would you like to make another walk? (N/Y)')
    if keep_running == 'N':
        break
```



```
would you like to make another walk? (N/Y)n
```




```
would you like to make another walk? (N/Y)N
```

[9]:
```python
#rolling dice
from random import randint
class die:
    def __init__(self, sides = 6):
        self.sides = sides

    def roll(self):
        return randint(1, self.sides)


die_class = die()

results = []
for roll_num in range(100):
    result = die_class.roll()
    results.append(result)
```

```
frequencies = []
for values in range(1, die_class.sides+1):
    frequency = results.count(values)
    frequencies.append(frequency)
print(frequencies)
```

```
[19, 15, 15, 17, 19, 15]
```

[ ]: