Question 1

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv('data.csv')
df.head()
```

Out[3]:

	order_id	shop_id	user_id	order_amount	total_items	payment_method	created_at
0	1	53	746	224	2	cash	2017-03-13 12:36:56
1	2	92	925	90	1	cash	2017-03-03 17:38:52
2	3	44	861	144	1	cash	2017-03-14 4:23:56
3	4	18	935	156	1	credit_card	2017-03-26 12:43:37
4	5	18	883	156	1	credit_card	2017-03-01 4:35:11

In [4]:
```
df.describe()
```

Out[4]:

	order_id	shop_id	user_id	order_amount	total_items
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000
mean	2500.500000	50.078800	849.092400	3145.128000	8.78720
std	1443.520003	29.006118	87.798982	41282.539349	116.32032
min	1.000000	1.000000	607.000000	90.000000	1.00000
25%	1250.750000	24.000000	775.000000	163.000000	1.00000
50%	2500.500000	50.000000	849.000000	284.000000	2.00000
75%	3750.250000	75.000000	925.000000	390.000000	3.00000
max	5000.000000	100.000000	999.000000	704000.000000	2000.00000

a)

In [5]:
```
n_rows = df.shape[0]
AOV_bad = np.sum(df['order_amount'])/n_rows
AOV_bad
```

Out[5]:
```
3145.128
```

As seen in the dataframe description and in cell 7, the AOV is presented, and as said in the question, is a very high value. Looking at each order, a majority of them have total_items under 10. However, there are outliers that have much larger item amounts, such as 2,000, which significantly sway the AOV.

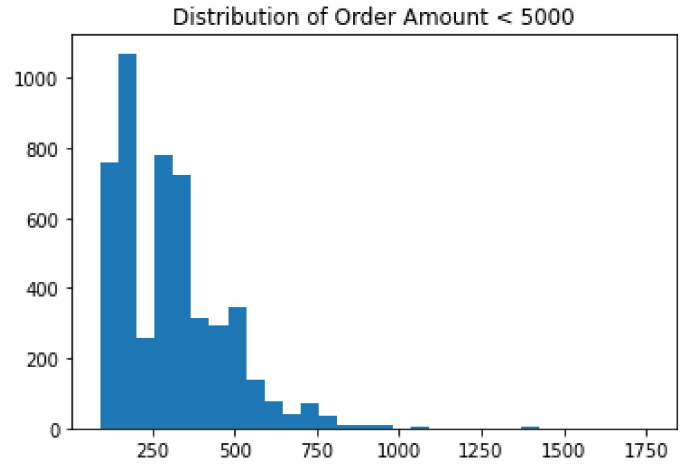
In [6]:
```
#looking at frequencies of each value for total_items

total_items = df['total_items'].unique()
counts = {i: df.loc[df['total_items'] == i].shape[0] for i in total_items}
counts
```

Out[6]:
```
{2: 1832, 1: 1830, 3: 941, 2000: 17, 4: 293, 5: 77, 6: 9, 8: 1}
```

There are many ways to approach the outlier problem. One of the simplest would be to use the median (50th percentile) instead of the mean order_amount. As seen above, the number of items ordered is commonly around 1, 2, and 3. There are also many outlier cases involving a low total_items value, such as 1 or 2, but with a high order amount, so using the median will be insensitive to these outliers. In this case, the median order_amount is $284.

However, simply using median may not always be reliable because it may not describe the distribution itself as meaningfully. Removing outliers and recalculating a better AOV is a better option.

In [7]:
```
df2 = df.loc[df['order_amount'] < 5000]
plt.hist(df2['order_amount'], bins=30)
plt.title('Distribution of Order Amount < 5000')
plt.show()
```



b)

It can be seen that the number of occurences of order amounts greater than about 600 is insignificant compared to the frequency of order amounts below. In other words, this distribution is very skewed. Instead of measuring the median of the entire dataset (including the outliers), a more reliable way would be to find the AOV on the dataset with outliers omitted. A popular technique is to use the interquartile range (between 25%-75%), but this would work best with symmetrical distributions, such as normal/gaussian. However, this distribution is heavily skewed. In this case, I will only use an upper bound, specifically 95%. I chose the 95th percentile instead of 75% because the significant outliers exist very far away from the majority of the data, so using a 75th percentile may discard important/meaningful data points.

c)

In [9]:
```
p95 = df['order_amount'].quantile(0.95)
p95
```

Out[9]:
```
640.0
```

The 95th percentile is 640, which seems to be an appropriate outlier threshold given the distribution shown above.

In [10]:
```
df_omitted = df.loc[df['order_amount'] < p95]
AOV_new = df_omitted['order_amount'].mean()
AOV_new
```

Out[10]:
```
284.1813198397639
```

The final value I give is 284.18.

This is almost exactly what the original median was! I presume this is due to the upper bound creating a more symmetric distribution.