Open in Colab

How To Install Modules/Libraries?

In [ ]:
```
# install the required modules
!pip install matplotlib
!pip install numpy
!pip install pandas
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages
(3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-pa
ckages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-p
ackages (from matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-p
ackages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-package
s (from matplotlib) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-pac
kages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packa
ges (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-pa
ckages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages
(from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.2
5.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.
0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/d
ist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packag
es (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-pack
ages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packa
ges (from pandas) (1.25.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages
(from python-dateutil>=2.8.2->pandas) (1.16.0)
```

In [ ]:
```
# importing the libraries/modules
import math as m
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:
```
# see versions
print(np.__version__)
print(pd.__version__)
```

```
1.25.2
2.0.3
```

In [ ]:
```
# to see matplotlib version
import matplotlib
print(matplotlib.__version__)
```

```
3.7.1
```

What we need for Research data Analysis and visualization?

- Data fetching/importing/creating
- Data Analysis
- Data Cleaning
- Data Visualization

In [ ]:
```
# .CSV - Comma seperated value
# "Name", "Addr"
# "janak", "ktm"
# "ram", "pkr"
```

Data Fetching/Importing

In [ ]:
```
# Data Creating and reading/importing/fetching from file and converting dataFrame
# Read csv file and store as a dataFrame
df = pd.read_csv('https://raw.githubusercontent.com/jsdhami/Python-For-Research/mai
df.head()
# read excel file
# df = pd.read_excel("titanic.xlsx", sheet_name="passengers")

# read html file
# df = pd.read_html("https://www.fdic.gov/bank/individual/failed/banklist.html")

# read json data read
# df = pd.read_json('data.json')
# and so...on
```

Out[ ]:

	AtomicNumber	Symbol	Name	AtomicMass	CPKHexColor	ElectronConfiguration	E
0	1	H	Hydrogen	1.008000	FFFFFF	1s1	
1	2	He	Helium	4.002600	D9FFFF	1s2	
2	3	Li	Lithium	7.000000	CC80FF	[He]2s1	
3	4	Be	Beryllium	9.012183	C2FF00	[He]2s2	
4	5	B	Boron	10.810000	FFB5B5	[He]2s2 2p1	

Data Creating by using numpy array

In [ ]:

```
## creating by using python in-build data type
X = [1, 2, 3, 4, 5]
Y = [1, 2, 3, 4, 5]

Xarray = np.array(X)
Yarray = np.array(Y)

# Or
# Xarray = np.array([1, 2, 3, 4, 5])
# Yarray = np.array([1, 2, 3, 4, 5])

# Or
# arange function syntax:
# start, end, step_interval
# Xarray = np.arange(1, 6, 0.5)
# Yarray = np.arange(1, 6, 0.5)

# linspace function syntax:
# start, end, total_number
# Xarray = np.linspace(1, 5, 6)
# Yarray = np.linspace(1, 5, 5)

# Or
# random function syntax:
# start, end, total_number
# Xarray = np.random.randint(1, 5, 6)
# Yarray = np.random.randint(1, 5, 5)

# Or
# another random... functions
# Xarray = np.random.normal(1, 5, 6)
# Yarray = np.random.normal(1, 5, 5)
# Or
# Xarray = np.random.uniform(1, 5, 6)
# Yarray = np.random.uniform(1, 5, 5)
# Or
# Xarray = np.random.random(6)
```

```
# Yarray = np.random.random(5)
# Or
# Xarray = np.random.rand(6)
# Yarray = np.random.rand(5)
# Or
# Xarray = np.random.randn(6)
# Yarray = np.random.randn(5)

# Others see on numpy




print(Xarray)
print(Yarray)
```

```
[2 1 2 3 1 3]
[3 2 4 3 4]
```

DataFrame and Series

In [ ]:
```
# Series
'''
A Pandas Series is like a column in a table.
It is a one-dimensional array holding data of any type.
'''
X = [1, 2, 3, 4, 5]
Y = [1, 2, 3, 4, 5]
# Create Series
# ser = pd.Series(X)
# print(ser)


# print(ser[2])

# Labeling series data
ser = pd.Series(X, index=['a', 'b', 'c', 'd', 'e'])
print(ser)
print(ser['d'])
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
4
```

In [ ]:
```
# Data Frame
'''
A Pandas DataFrame is a 2-dimensional data structure,
like a 2-dimensional array, or a table with rows and columns.
'''
```

```python
myData = {
    "students":["Janak", None, "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, 60, 50],
}

# Create DataFrame
df = pd.DataFrame(myData)
# print(df)

# column select
# print(df['students'])
# print(df['marks'])

#refer to the row index:
# print(df.loc[0])

#use a list of indexes:
# print(df.loc[[0, 1]])


# Labeling
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])
print(df)
# print(df.loc['a'])
df.info()
```

```
  students  marks
a    Janak     90
b     None     80
c    Shyam     70
d     Sita     60
e     Gita     50
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, a to e
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   students  4 non-null      object
 1   marks     5 non-null      int64 
dtypes: int64(1), object(1)
memory usage: 120.0+ bytes
```

Data Analysis

In [ ]:

```python
# to see top datas
# df.head()

#  to see bottom datas
# df.tail()

# to see custom no. of datas
df.head(1)

# to see info about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, a to e
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   students  4 non-null      object
 1   marks     5 non-null      int64 
dtypes: int64(1), object(1)
memory usage: 120.0+ bytes
```

Data cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

In [ ]:
```
# removing empty cell [row x column]
myData = {
    "students":["Janak", None, "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, 60, 50]
}

# Create DataFrame
df = pd.DataFrame(myData)
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])

new_df = df.dropna()

print(new_df.to_string())
```

```
  students  marks
a    Janak     90
c    Shyam     70
d     Sita     60
e     Gita     50
```

In [ ]:
```
# remove all rows with Null Value
myData = {
    "students":["Janak", None, "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, 60, 50]
}

# Create DataFrame
df = pd.DataFrame(myData)
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])
df.dropna(inplace = True)
```

```
print(df.to_string())
```

```
  students  marks
a    Janak     90
c    Shyam     70
d     Sita     60
e     Gita     50
```

In [ ]:
```
# Replace NULL values with the number "Ram":
myData = {
    "students":["Janak", None, "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, 60, 50]
}

# Create DataFrame
df = pd.DataFrame(myData)
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])
df.fillna("Ram", inplace = True)
print(df.to_string())
```

```
  students  marks
a    Janak     90
b      Ram     80
c    Shyam     70
d     Sita     60
e     Gita     50
```

In [ ]:
```
# Replace Only For Specified Columns

myData = {
    "students":["Janak", None, "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, 60, None]
}

# Create DataFrame
df = pd.DataFrame(myData)
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])

# df['students'].fillna("Ramu", inplace = True)
# df['marks'].fillna(100, inplace = True) # not updating because of non Null data
print(df.to_string())
```

```
  students  marks
a    Janak   90.0
b     None   80.0
c    Shyam   70.0
d     Sita   60.0
e     Gita  100.0
```

In [ ]:
```
# Replace Using Mean, Median, or Mode
# Pandas uses the mean() median() and mode() methods to calculate the respective va

myData = {
    "students":["Janak", "Ram", "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, None, 50]
}
```

```
# Create DataFrame
df = pd.DataFrame(myData)
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])

# using mean
# Mean = the average value (the sum of all values divided by number of values).
sitaNum = df['marks'].mean()
df['marks'].fillna(sitaNum, inplace = True)
print(df.to_string())
```

```
  students  marks
a    Janak   90.0
b      Ram   80.0
c    Shyam   70.0
d     Sita   72.5
e     Gita   50.0
```

In [ ]:
```
# using median
# Median = the value in the middle, after you have sorted all values ascending.
myData = {
    "students":["Janak", "Ram", "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, None, 50]
}

# Create DataFrame
df = pd.DataFrame(myData)
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])
df['marks'].fillna(df['marks'].median(), inplace = True)
print(df.to_string())
```

```
  students  marks
a    Janak   90.0
b      Ram   80.0
c    Shyam   70.0
d     Sita   75.0
e     Gita   50.0
```

In [ ]:
```
# using mode
# Mode = the value that appears most frequently.
myData = {
    "students":["Janak", "Ram", "Shyam", "Sita", "Gita"],
    "marks":[90, 80, 70, None, 50]
}

# Create DataFrame
df = pd.DataFrame(myData)
df = pd.DataFrame(myData, index=['a', 'b', 'c', 'd', 'e'])

df['marks'].fillna(df['marks'].mode()[0], inplace = True)
print(df.to_string())
```

```
  students  marks
a    Janak   90.0
b      Ram   80.0
c    Shyam   70.0
d     Sita   50.0
e     Gita   50.0
```

Data Cleaning - Worng Format

- Cells with data of wrong format can make it difficult, or even impossible, to analyze data.
- To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format.

In [ ]:
```
df = pd.read_csv('https://raw.githubusercontent.com/jsdhami/Python-For-Research/mai

print(df.to_string())
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
18        45  '2020/12/18'     90       112       NaN
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
22        45           NaN    100       119     282.0
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
26        60      20201226    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
28        60  '2020/12/28'    103       132       NaN
29        60  '2020/12/29'    100       132     280.0
30        60  '2020/12/30'    102       129     380.3
31        60  '2020/12/31'     92       115     243.0
```

In [ ]:
```
#  remove null value included row
df.dropna(subset=['Date'], inplace = True)

df.dropna(subset=['Calories'], inplace = True)

# print(df.to_string())



df['Date'] = pd.to_datetime(df['Date'], format='mixed')

print(df.to_string())
```

```
    Duration       Date  Pulse  Maxpulse  Calories
0         60 2020-12-01    110       130     409.1
1         60 2020-12-02    117       145     479.0
2         60 2020-12-03    103       135     340.0
3         45 2020-12-04    109       175     282.4
4         45 2020-12-05    117       148     406.0
5         60 2020-12-06    102       127     300.0
6         60 2020-12-07    110       136     374.0
7        450 2020-12-08    104       134     253.3
8         30 2020-12-09    109       133     195.1
9         60 2020-12-10     98       124     269.0
10        60 2020-12-11    103       147     329.3
11        60 2020-12-12    100       120     250.7
12        60 2020-12-12    100       120     250.7
13        60 2020-12-13    106       128     345.3
14        60 2020-12-14    104       132     379.3
15        60 2020-12-15     98       123     275.0
16        60 2020-12-16     98       120     215.2
17        60 2020-12-17    100       120     300.0
19        60 2020-12-19    103       123     323.0
20        45 2020-12-20     97       125     243.0
21        60 2020-12-21    108       131     364.2
23        60 2020-12-23    130       101     300.0
24        45 2020-12-24    105       132     246.0
25        60 2020-12-25    102       126     334.5
26        60 2020-12-26    100       120     250.0
27        60 2020-12-27     92       118     241.0
29        60 2020-12-29    100       132     280.0
30        60 2020-12-30    102       129     380.3
31        60 2020-12-31     92       115     243.0
```

Removing Wrong Data

- "Wrong data" does not have to be "empty cells" or "wrong format", it can just be wrong, like if someone registered "199" instead of "1.99".

In [ ]:
```
# replace value
df.loc[7, 'Duration'] = 5
print(df.to_string())
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409.1
1	60	2020-12-02	117	145	479.0
2	60	2020-12-03	103	135	340.0
3	45	2020-12-04	109	175	282.4
4	45	2020-12-05	117	148	406.0
5	60	2020-12-06	102	127	300.0
6	60	2020-12-07	110	136	374.0
7	5	2020-12-08	104	134	253.3
8	30	2020-12-09	109	133	195.1
9	60	2020-12-10	98	124	269.0
10	60	2020-12-11	103	147	329.3
11	60	2020-12-12	100	120	250.7
12	60	2020-12-12	100	120	250.7
13	60	2020-12-13	106	128	345.3
14	60	2020-12-14	104	132	379.3
15	60	2020-12-15	98	123	275.0
16	60	2020-12-16	98	120	215.2
17	60	2020-12-17	100	120	300.0
19	60	2020-12-19	103	123	323.0
20	45	2020-12-20	97	125	243.0
21	60	2020-12-21	108	131	364.2
23	60	2020-12-23	130	101	300.0
24	45	2020-12-24	105	132	246.0
25	60	2020-12-25	102	126	334.5
26	60	2020-12-26	100	120	250.0
27	60	2020-12-27	92	118	241.0
29	60	2020-12-29	100	132	280.0
30	60	2020-12-30	102	129	380.3
31	60	2020-12-31	92	115	243.0

In [ ]:
```
# remove
df.drop(index=15, inplace=True)
print(df.to_string())
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409.1
1	60	2020-12-02	117	145	479.0
2	60	2020-12-03	103	135	340.0
3	45	2020-12-04	109	175	282.4
4	45	2020-12-05	117	148	406.0
5	60	2020-12-06	102	127	300.0
6	60	2020-12-07	110	136	374.0
7	5	2020-12-08	104	134	253.3
8	30	2020-12-09	109	133	195.1
9	60	2020-12-10	98	124	269.0
10	60	2020-12-11	103	147	329.3
11	60	2020-12-12	100	120	250.7
12	60	2020-12-12	100	120	250.7
13	60	2020-12-13	106	128	345.3
14	60	2020-12-14	104	132	379.3
16	60	2020-12-16	98	120	215.2
17	60	2020-12-17	100	120	300.0
19	60	2020-12-19	103	123	323.0
20	45	2020-12-20	97	125	243.0
21	60	2020-12-21	108	131	364.2
23	60	2020-12-23	130	101	300.0
24	45	2020-12-24	105	132	246.0
25	60	2020-12-25	102	126	334.5
26	60	2020-12-26	100	120	250.0
27	60	2020-12-27	92	118	241.0
29	60	2020-12-29	100	132	280.0
30	60	2020-12-30	102	129	380.3
31	60	2020-12-31	92	115	243.0

Removing Duplicates

- Duplicate rows are rows that have been registered more than one time.

In [ ]:
```
print(df.duplicated())
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12     True
13    False
14    False
16    False
17    False
19    False
20    False
21    False
23    False
24    False
25    False
26    False
27    False
29    False
30    False
31    False
dtype: bool
```

In [ ]:
```
# to remove duplicate value
df.drop_duplicates(inplace=True)
print(df.to_string())
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409.1
1	60	2020-12-02	117	145	479.0
2	60	2020-12-03	103	135	340.0
3	45	2020-12-04	109	175	282.4
4	45	2020-12-05	117	148	406.0
5	60	2020-12-06	102	127	300.0
6	60	2020-12-07	110	136	374.0
7	5	2020-12-08	104	134	253.3
8	30	2020-12-09	109	133	195.1
9	60	2020-12-10	98	124	269.0
10	60	2020-12-11	103	147	329.3
11	60	2020-12-12	100	120	250.7
13	60	2020-12-13	106	128	345.3
14	60	2020-12-14	104	132	379.3
16	60	2020-12-16	98	120	215.2
17	60	2020-12-17	100	120	300.0
19	60	2020-12-19	103	123	323.0
20	45	2020-12-20	97	125	243.0
21	60	2020-12-21	108	131	364.2
23	60	2020-12-23	130	101	300.0
24	45	2020-12-24	105	132	246.0
25	60	2020-12-25	102	126	334.5
26	60	2020-12-26	100	120	250.0
27	60	2020-12-27	92	118	241.0
29	60	2020-12-29	100	132	280.0
30	60	2020-12-30	102	129	380.3
31	60	2020-12-31	92	115	243.0

Data Correlations

- The corr() method calculates the relationship between each column in your data set.

In [ ]:
```
# df.corr()
df['Pulse'].describe()
```

Out[ ]:
```
count     27.000000
mean     104.481481
std        7.939048
min       92.000000
25%      100.000000
50%      103.000000
75%      108.500000
max      130.000000
Name: Pulse, dtype: float64
```

In [ ]:
```
# result explained
# The number varies from -1 to 1.

# 1 means that there is a 1 to 1 relationship (a perfect correlation), and for this

# 0.9 is also a good relationship, and if you increase one value, the other will pr

# -0.9 would be just as good relationship as 0.9, but if you increase one value, th
```

```
# 0.2 means NOT a good relationship, meaning that if one value goes up does not mea
```

Plotting On Python

- Pandas uses the plot() method to create diagrams.

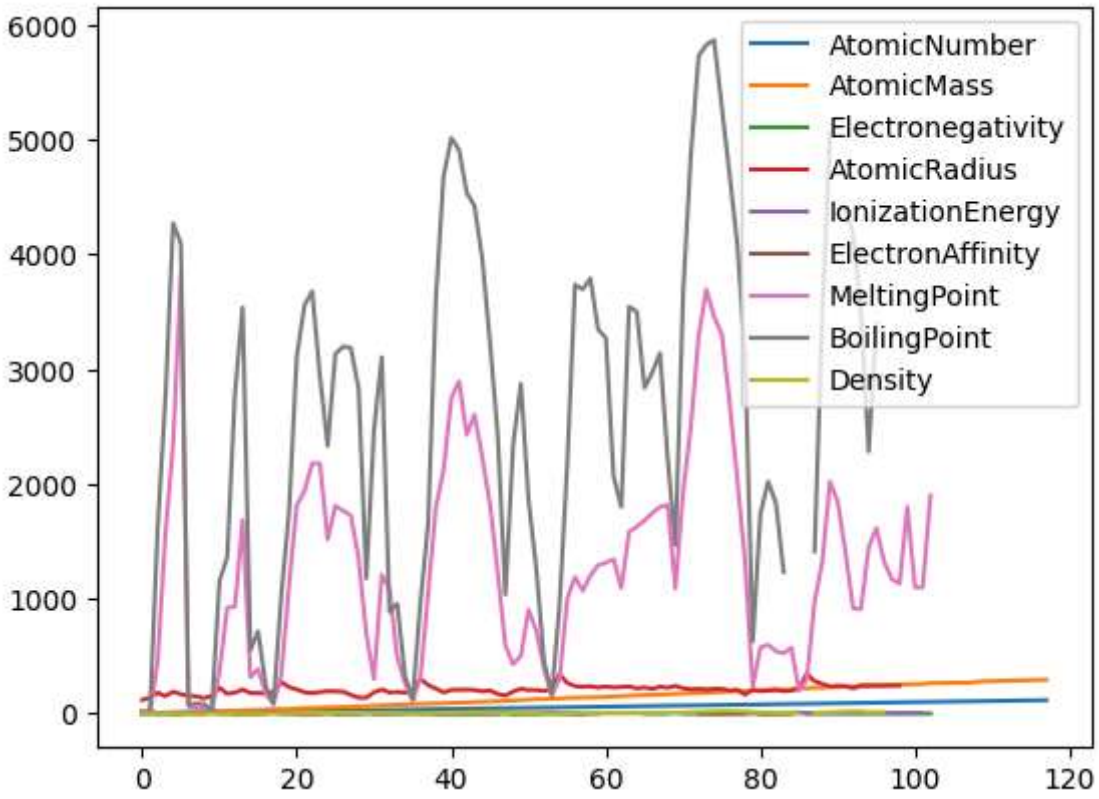
In [ ]:
```
dataFrame = pd.read_csv('https://raw.githubusercontent.com/jsdhami/Python-For-Resea
dataFrame.head()
```

Out[ ]:

	AtomicNumber	Symbol	Name	AtomicMass	CPKHexColor	ElectronConfiguration	E
0	1	H	Hydrogen	1.008000	FFFFFF	1s1	
1	2	He	Helium	4.002600	D9FFFF	1s2	
2	3	Li	Lithium	7.000000	CC80FF	[He]2s1	
3	4	Be	Beryllium	9.012183	C2FF00	[He]2s2	
4	5	B	Boron	10.810000	FFB5B5	[He]2s2 2p1	

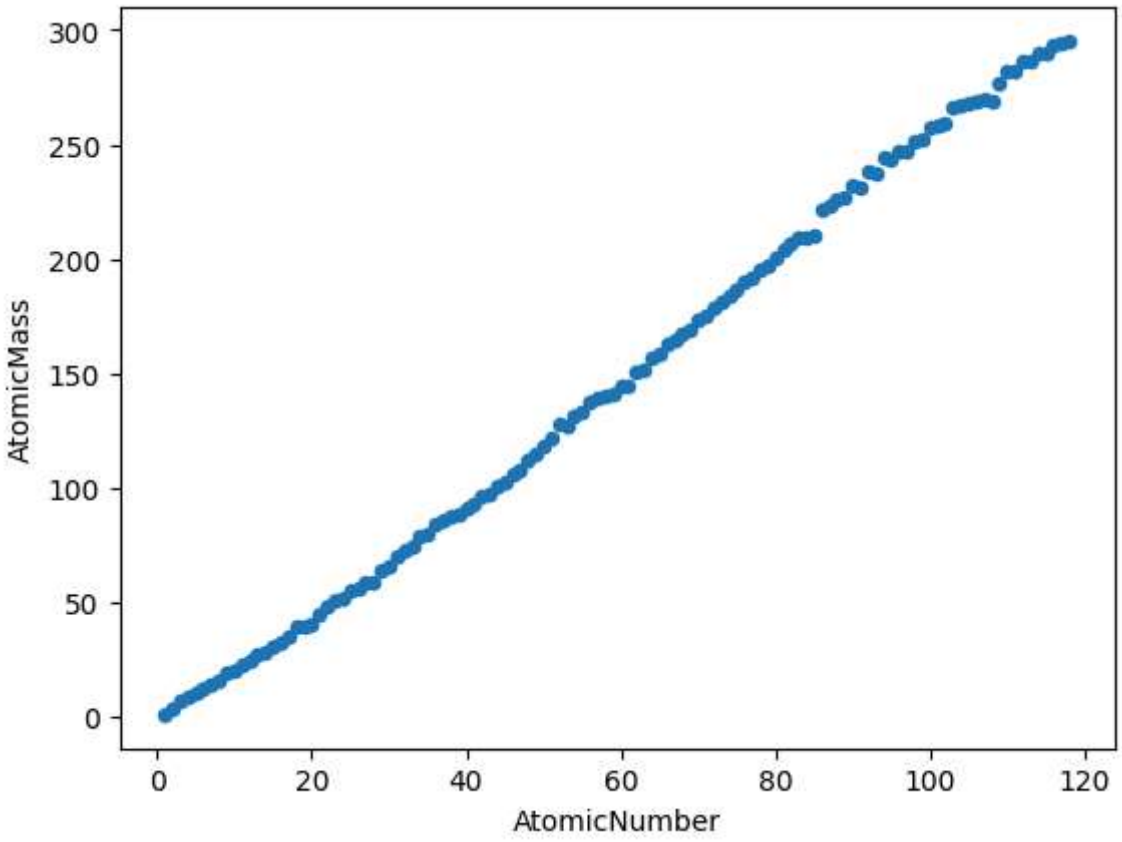
In [ ]:
```
dataFrame.plot()
```

Out[ ]:
```
<Axes: >
```



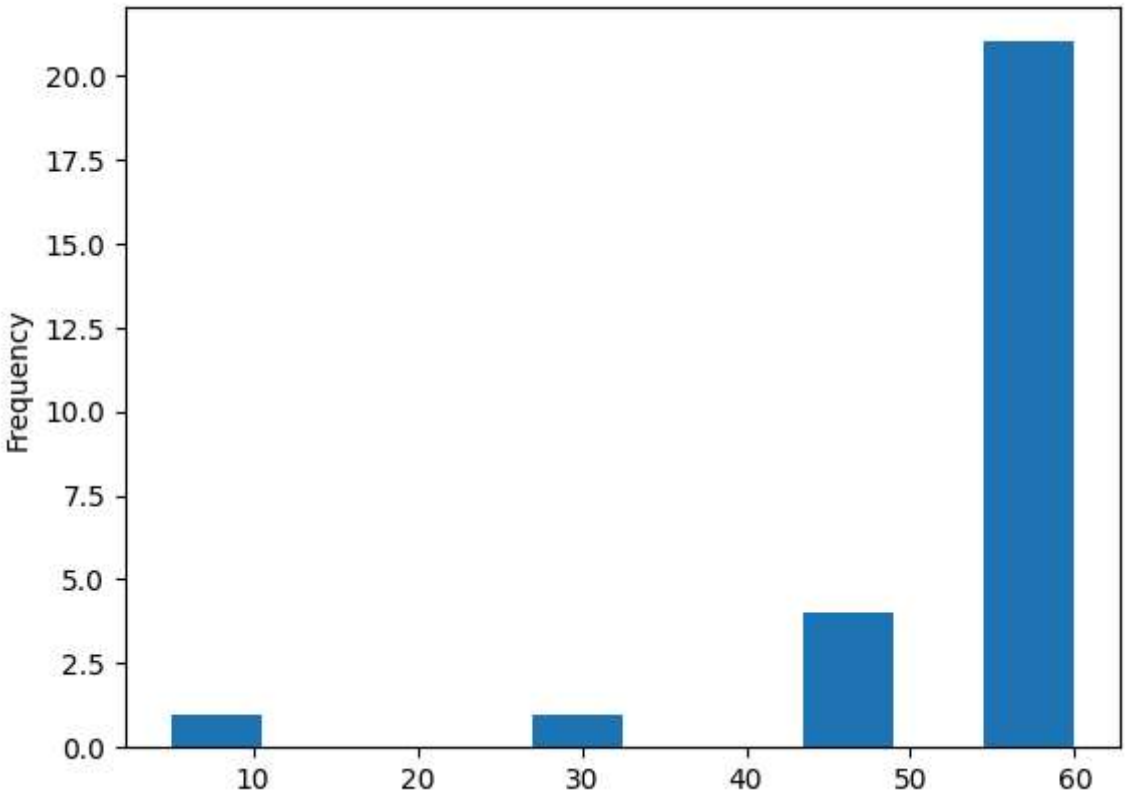
In [ ]:
```
# Scatter Plot
# A scatter plot needs an x- and a y-axis.
# df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

# from atomic data tabe
dataFrame.plot(kind = 'scatter', x = 'AtomicNumber', y = 'AtomicMass')
```

Out[ ]:
```
<Axes: xlabel='AtomicNumber', ylabel='AtomicMass'>
```



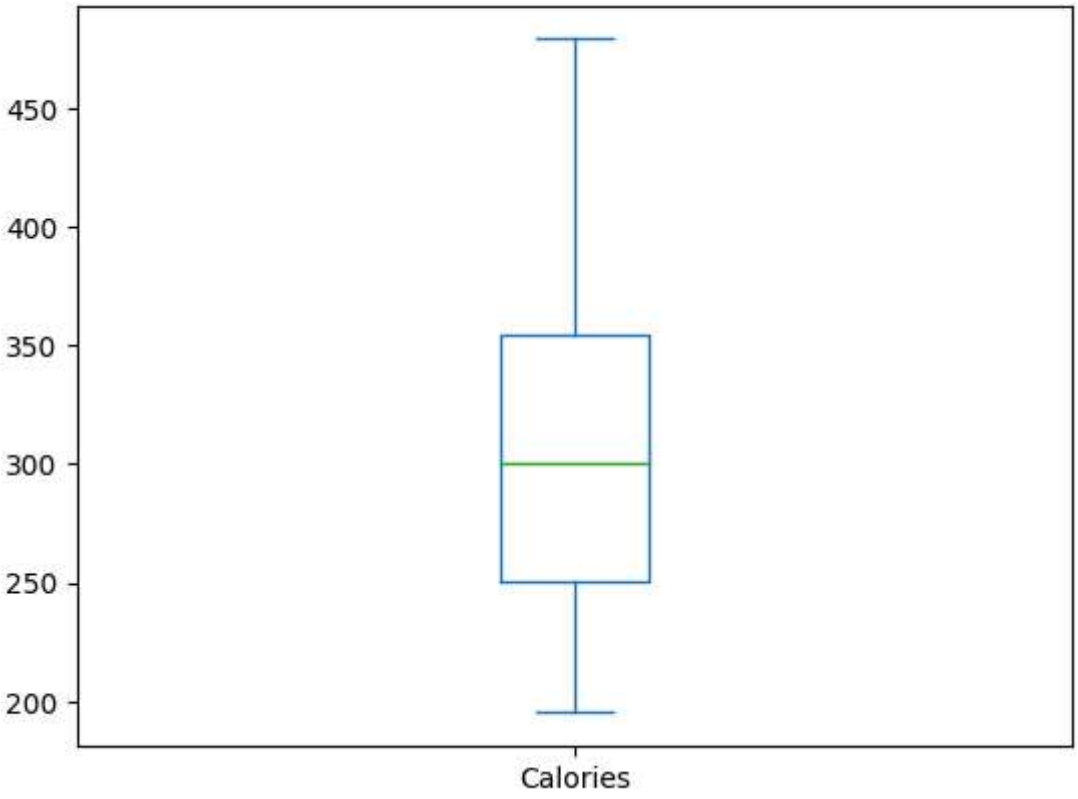
In [ ]:
```
# Histogram
# A histogram needs only one column.
df['Duration'].plot(kind = 'hist')
```

Out[ ]:
```
<Axes: ylabel='Frequency'>
```



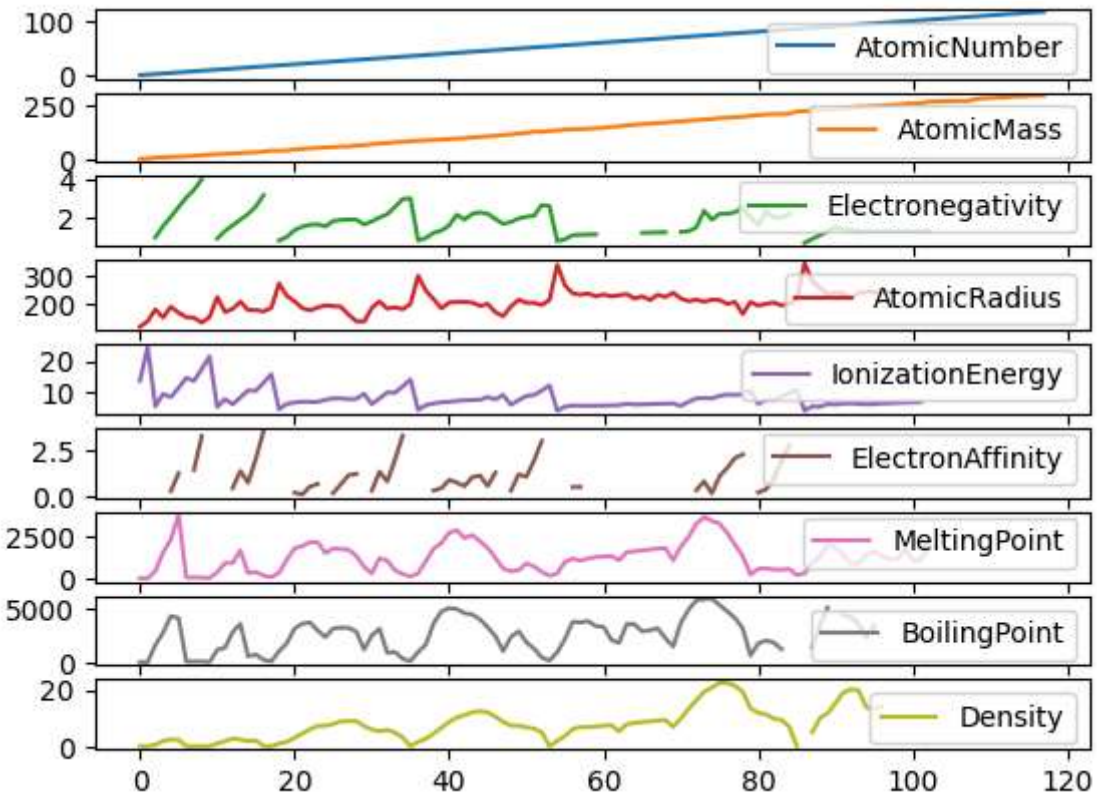
In [ ]:
```
# Box Plot
# A box plot needs only one column.
df['Calories'].plot(kind = 'box')
```

Out[ ]: <Axes: >



In [ ]:
```
# create subplot using pandas
dataFrame.plot(subplots = True)
```

Out[ ]:
```
array([<Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >, <Axes: >,
       <Axes: >, <Axes: >, <Axes: >], dtype=object)
```

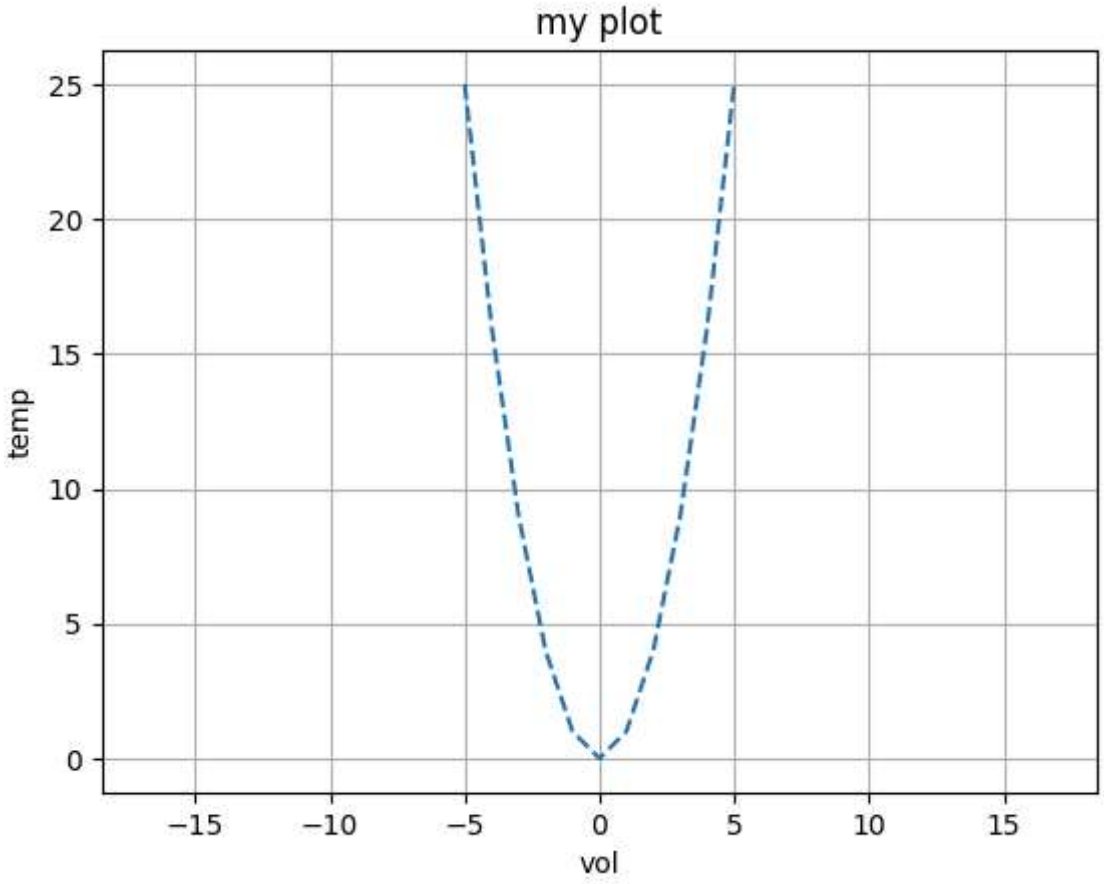


Matplotlib Complete

See on next Jupyter Notebook...

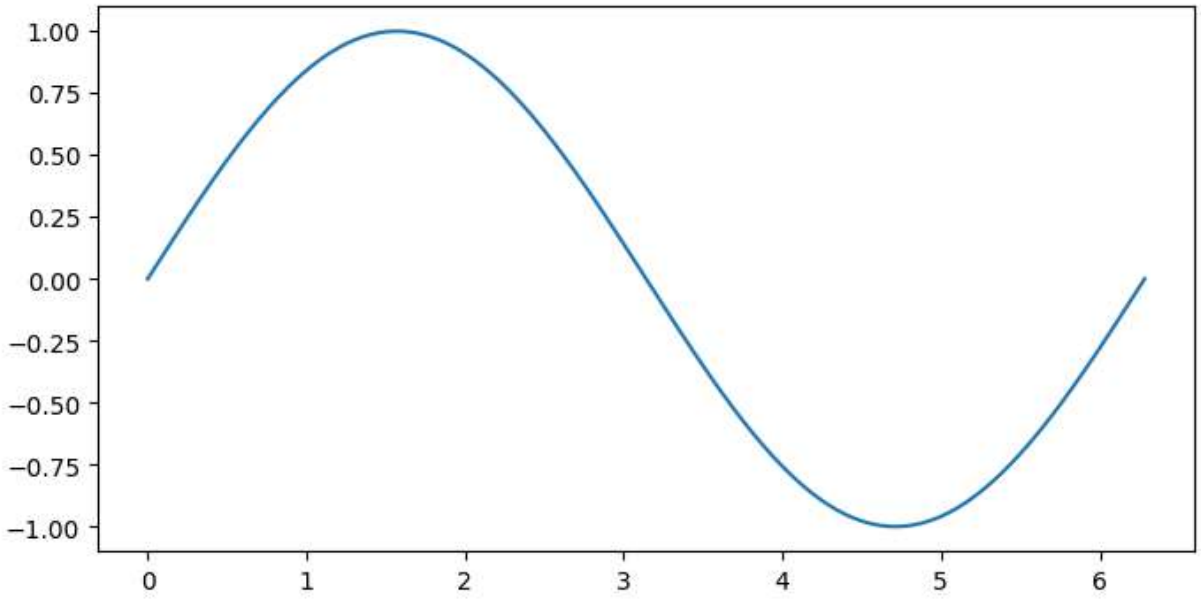
In [ ]:
```
x = np.linspace(-5, 5, 11)
y = x ** 2

# plot
plt.plot(x, y, '--')
# plote center axis
plt.axis('equal')
plt.grid(True)
plt.title('my plot')
plt.xlabel('vol')
plt.ylabel('temp')
plt.show()
```

Out[ ]:
```
Text(0, 0.5, 'temp')
```



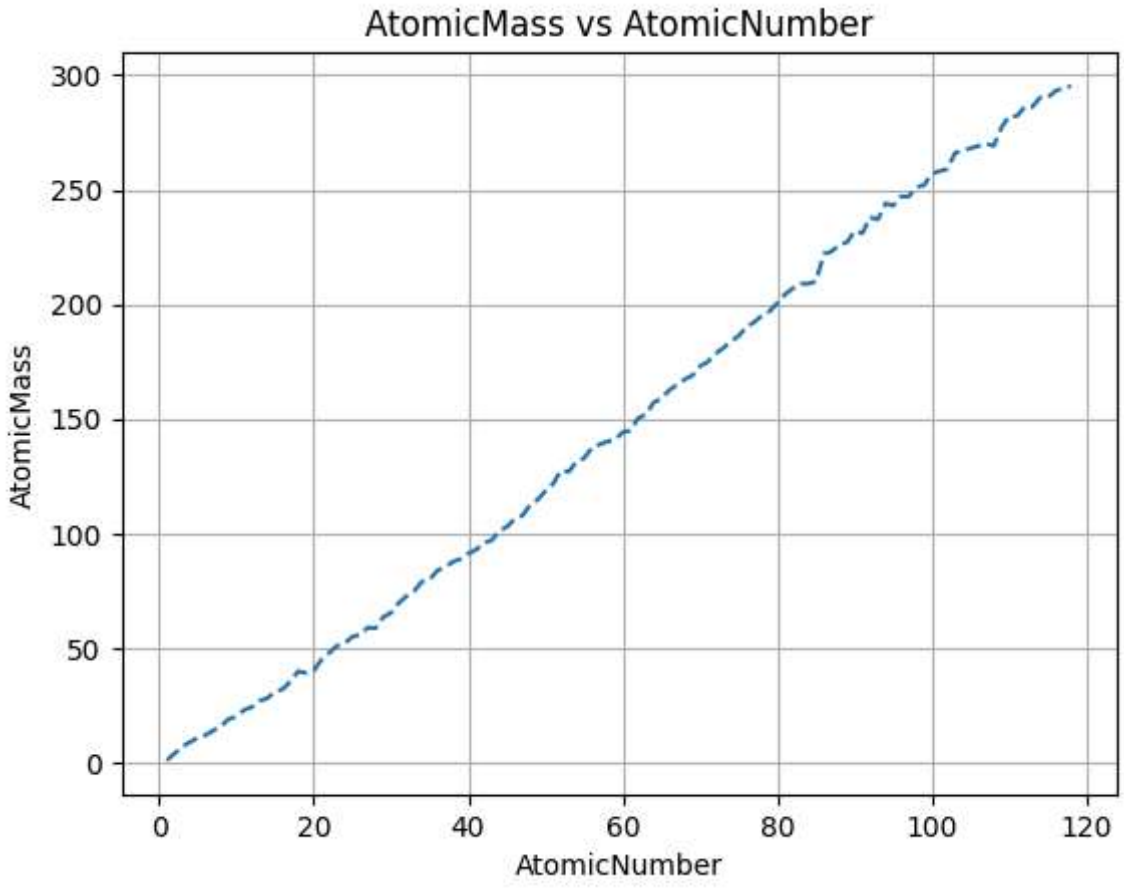
In [ ]:
```
# fig = plt.figure()              # an empty figure with no Axes
# fig, ax = plt.subplots()        # a figure with a single Axes
# with log true
fig, ax = plt.subplots(figsize=(8, 4), facecolor='w')
# fig, axs = plt.subplots(3, 2)  # a figure with a 2x2 grid of Axes
x = np.linspace(0, 2 * np.pi, 400)
ax.plot(x, np.sin(x))

# axs[1,0].set_yscale('log')
# ax.set_yscale('linear')
# a figure with one Axes on the left, and two on the right:
# fig, axs = plt.subplot_mosaic([['left', 'right_top'],
#                                ['left', 'right_bottom']])
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7f489e0e4790>]
```



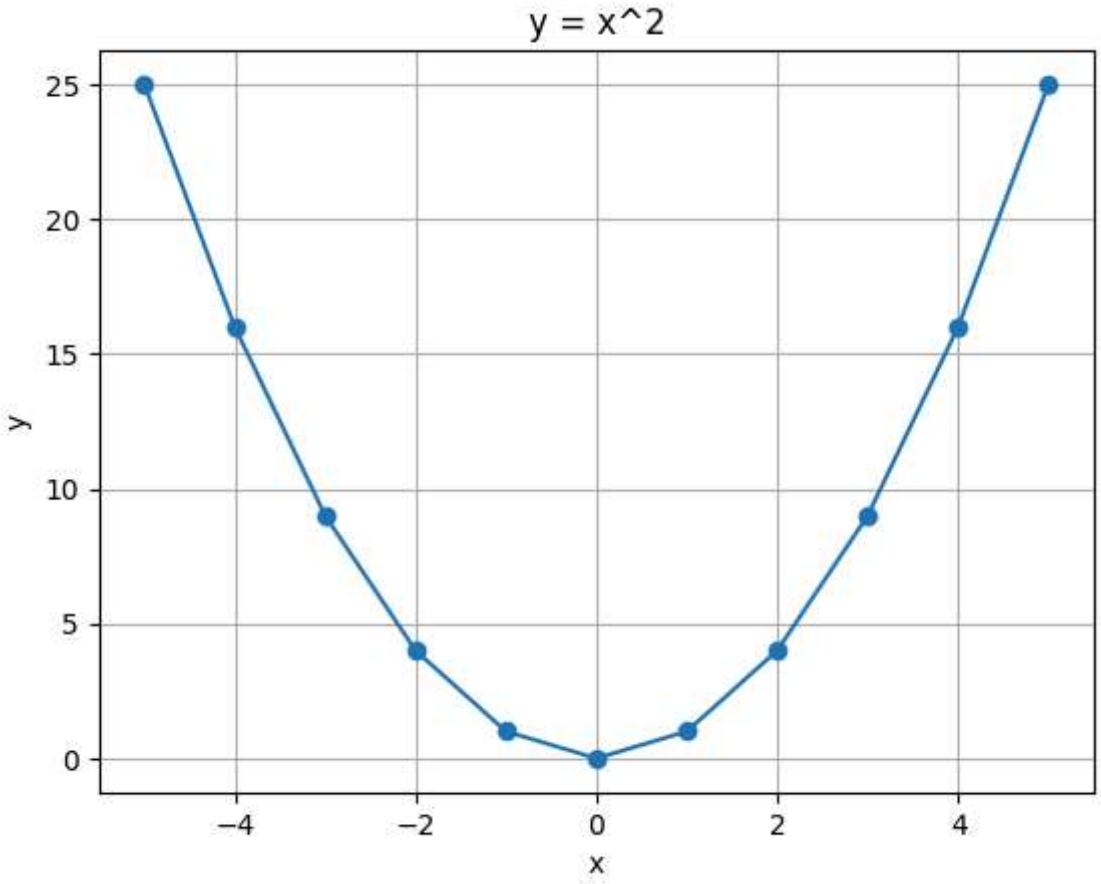
In [ ]:
```
x = dataFrame['AtomicNumber']
# x = df.iloc[:, 0]
print(x)
y = dataFrame['AtomicMass']
# x.head()

plt.plot(x, y, '--')
# # plote center axis
# plt.axis('equal')
plt.grid(True)
plt.title('AtomicMass vs AtomicNumber')
plt.xlabel('AtomicNumber')
plt.ylabel('AtomicMass')
plt.show()
```

```
0        1
1        2
2        3
3        4
4        5
      ...
113    114
114    115
115    116
116    117
117    118
Name: AtomicNumber, Length: 118, dtype: int64
```



In [ ]:
```
# mathematical questions
x = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]

ye = []
for i in x:
  y = i*i
  ye.append(y)

print(ye)

plt.plot(x, ye, '-o')
# plote center axis
# plt.axis('equal')
plt.grid()
plt.title('y = x^2')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

```
[25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25]
```



In [ ]:
```
# Syntax: import moduleName
import math

# Syntax: import moduleName as shortform
import math as m
```

In [ ]:
```
# How to use python library/module
print(m.sin(m.pi/2))
```

```
1.0
```

Something about Ploty.JS

In [ ]:
```
# something about Ploty.JS

import plotly.express as px

df = pd.DataFrame(dict(
    x = [1, 3, 2, 4],
    y = [1, 2, 3, 4]
))
fig = px.line(df, x="x", y="y", title="Unsorted Input")
fig.show()
```