**Tìm hiểu thư viện PANDAS, NUMPY, MATHPLOTLIP**

* Truy xuất dữ liệu từ file CSV/Excel, từ CSDL Oracle/SQL Server
* Kết hợp dữ liệu từ nhiều sheet trong 1 file CSV/Excel
* Xử lý Dataframe
* Xử lý làm sạch dữ liệu
* Thư viện trực quan hóa dữ liệu
* Thực hiện các thống kê và vẽ biểu đồ

## Truy xuất dữ liệu từ file CSV/Excel, từ CSDL Oracle/SQL Server

### Truy xuất dữ liệu từ file CSV/Excel

* Các đơn giản nhất là dùng thư viện Pandas, với hàm đọc read\_csv và read\_excel

In [1]:

*# Đọc file CSV*

**import** pandas **as** pd

df\_data **=** pd**.**read\_csv('iris.csv')

print(df\_data**.**info()) *# Xem thông tin dữ liệu*

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Id 150 non-null int64

1 SepalLengthCm 150 non-null float64

2 SepalWidthCm 150 non-null float64

3 PetalLengthCm 150 non-null float64

4 PetalWidthCm 150 non-null float64

5 Species 150 non-null object

dtypes: float64(4), int64(1), object(1)

memory usage: 7.2+ KB

None

In [2]:

*# Đọc file Excel*

file\_path **=** "D:/Data/VNPT/KHDT.xlsx"

df\_kh **=** pd**.**read\_excel(file\_path)

df\_kh**.**info() *# Xem thông tin dữ liệu*

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 13 entries, 0 to 12

Data columns (total 10 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 TT 13 non-null int64

1 ID 9 non-null object

2 Lĩnh vực 13 non-null object

3 Y0 13 non-null int64

4 TH\_Y2 13 non-null int64

5 KH\_Y1 12 non-null float64

6 TT\_Y1 13 non-null int64

7 TH\_Y1 10 non-null float64

8 KH\_Y0 10 non-null float64

9 TT\_Y0 13 non-null int64

dtypes: float64(3), int64(5), object(2)

memory usage: 1.1+ KB

Chi tiết hàm pandas.read\_csv

pandas.read\_csv(filepath\_or\_buffer, \*, sep=<no\_default>, delimiter=None, header='infer', names=<no\_default>, index\_col=None, usecols=None, dtype=None, engine=None, converters=None, true\_values=None, false\_values=None, skipinitialspace=False, skiprows=None, skipfooter=0, nrows=None, na\_values=None, keep\_default\_na=True, na\_filter=True, verbose=<no\_default>, skip\_blank\_lines=True, parse\_dates=None, infer\_datetime\_format=<no\_default>, keep\_date\_col=<no\_default>, date\_parser=<no\_default>, date\_format=None, dayfirst=False, cache\_dates=True, iterator=False, chunksize=None, compression='infer', thousands=None, decimal='.', lineterminator=None, quotechar='"', quoting=0, doublequote=True, escapechar=None, comment=None, encoding=None, encoding\_errors='strict', dialect=None, on\_bad\_lines='error', delim\_whitespace=<no\_default>, low\_memory=True, memory\_map=False, float\_precision=None, storage\_options=None, dtype\_backend=<no\_default>)

* Có rất nhiều tham số tùy vào dữ liệu đầu vào, một số tham số chú ý như sau:
* sep: ký tự ngăn cách trong file CSV
* skiprows: số dòng bỏ qua từ đầu file

Chi tiết hàm pandas.read\_excel

pandas.read\_excel(io, sheet\_name=0, \*, header=0, names=None, index\_col=None, usecols=None, dtype=None, engine=None, converters=None, true\_values=None, false\_values=None, skiprows=None, nrows=None, na\_values=None, keep\_default\_na=True, na\_filter=True, verbose=False, parse\_dates=False, date\_parser=<no\_default>, date\_format=None, thousands=None, decimal='.', comment=None, skipfooter=0, storage\_options=None, dtype\_backend=<no\_default>, engine\_kwargs=None)

* Một số tham số đáng chú ý như:
* sheet\_name: tên sheet, nếu muốn lấy dữ liệu từ sheet nào thì đưa tên sheet vào
* skiprows, nrows: có thể đọc từ dòng thứ mấy (bỏ qua skprows dòng), và đọc số lượng (nrows) dòng theo yêu cầu

Kết hợp nhiều file CSV/Excel

Giả sử đã có một hàm đọc dữ liệu từ file CSV/Excel trả về đối tượng DataFrame (Pandas) Sử dụng hàm pd.concat() để kết hợp dữ liệu từ nhiều file => trả về một sối tượng DataFrame

In [3]:

*# Đọc dữ liệu nhiều tháng*

**def** csv\_months(folder, template, m1, y1, m2**=None**, y2**=None**, **\*\***kwargs):

results **=** [csv\_month(folder, template, mm, yyyy, **\*\***kwargs)

**for** mm, yyyy **in** month\_year\_range(m1, y1, m2 **or** m1, y2 **or** y1)]

**return** pd**.**concat(results) **if** len(results) **>** 0 **else** **None**

*# Đọc dữ liệu nhiều tháng từ báo cáo file Excel*

**def** excel\_months(file\_config, m1, y1, m2**=None**, y2**=None**, **\*\***kwargs):

results **=** [excel\_month(file\_config, mm, yyyy, **\*\***kwargs)

**for** mm, yyyy **in** month\_year\_range(m1, y1, m2 **or** m1, y2 **or** y1)]

**return** pd**.**concat(results) **if** len(results) **>** 0 **else** **None**

### Truy xuất dữ liệu từ CSDL Oracle / SQL Server

* Đối với SQL Server: sử dụng thư viện pyodbc
* Đối với Oracle: oracledb

Để đơn giản viết thành các thư viện truy xuất dữ liệu từ câu lện SQL query hoặc thực thi các thủ tục lưu sẵn (Stored Procedure)

* Gồm các hàm cơ bản sau:
  + connect(self) => kết nối CSDL với tài khoản và thông tin server được cấp quyền truy cập
  + exec\_sql\_df(self, sqlString) => Thực thi câu lệnh SQL => trả về dữ liệu => DataFrame
  + exec\_sp\_df(self, spName, \*args): =>Thực thi một thủ tục Stored Procedure => trả về dữ liệu => DataFrame
  + exec\_non\_query(self, spName, \*args) =>Thực thi một thủ tục Stored Procedure => cập nhật/xóa => không trả về dữ liệu
  + exec\_scalar(self, spName, \*args) => Thực thi một thủ tục Stored Procedure => trả về chỉ 1 giá trị

In [4]:

*# Thư viện đọc dữ liệu từ SQL Server*

*#phnampy@gmail.com*

**import** pyodbc

**import** pandas **as** pd

**class** MsSqlDB:

**def** \_\_init\_\_(self, dbServer, dbUsername, dbPassword):

*#self.\_server = f'{dbServer.get("host", "localhost")},{dbServer.get("port", 1443)}'*

self**.**\_server **=** f'{dbServer**.**get("host", "localhost")}'

self**.**\_dbname **=** dbServer**.**get("db\_name")

self**.**\_user **=** dbUsername

self**.**\_password **=** dbPassword

**def** info(self):

**return** f'Server:{self**.**\_server}, DbName:{self**.**\_dbname}, DbUsername:{self**.**\_user}, DbPassword:{self**.**\_password}'

**def** connect(self):

**return** pyodbc**.**connect(driver**=**'{SQL Server}',

server**=**self**.**\_server,

database**=**self**.**\_dbname,

uid**=**self**.**\_user,

pwd**=**self**.**\_password)

*# Thực thi câu lệnh SQL => trả về dữ liệu => DataFrame*

**async** **def** exec\_sql\_df(self, sqlString):

**try**:

**with** self**.**connect() **as** connection:

**return** pd**.**read\_sql\_query(sqlString, connection)

**except** pyodbc**.**Error **as** error:

**raise** Exception(error)

**def** \_\_parseParams(self, **\***args):

params **=** ()

sqlParams **=** ''

**if** len(args) **>** 0:

**if** len(args) **==** 1 **and** type(args[0]) **is** dict:

**for** para **in** args[0]**.**items():

sqlParams **+=** f' @{para[0]} = ?,'

params **+=** (para[1],)

**else**:

**for** para **in** args:

sqlParams **+=** ' ?,'

params **+=** (para,)

sqlParams **=** sqlParams[:**-**1]

**return** sqlParams, params

*# Thực thi một thủ tục Stored Procedure => trả về dữ liệu => DataFrame*

**async** **def** exec\_sp\_df(self, spName, **\***args):

**try**:

**with** self**.**connect() **as** connection:

sqlParams, params **=** self**.**\_\_parseParams(**\***args)

sql **=** 'SET NOCOUNT ON;\r' **+** '{CALL ' **+** spName **+** '(' **+** sqlParams **+** ')}'

**return** pd**.**read\_sql(sql**=**sql, con**=**connection, params**=**params)

**except** pyodbc**.**Error **as** error:

**raise** Exception(error)

*# Thực thi một thủ tục Stored Procedure => cập nhật/xóa => không trả về dữ liệu*

**async** **def** exec\_non\_query(self, spName, **\***args):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

cursor**.**execute(spName, args)

connection**.**commit()

**except** pyodbc**.**Error **as** error:

**raise** Exception(error)

*# Thực thi một thủ tục Stored Procedure => trả về chỉ 1 giá trị*

**async** **def** exec\_scalar(self, spName, **\***args):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

**return** cursor**.**execute(spName, args)**.**fetchval()

**except** pyodbc**.**Error **as** error:

**raise** Exception(error)

In [5]:

*# Thư viện đọc dữ liệu từ Oracle*

*#phnampy@gmail.com*

**import** oracledb **as** ora

**import** pandas **as** pd

**class** OracleDB:

**def** \_\_init\_\_(self, dbServer, dbUsername, dbPassword):

self**.**\_server **=** ora**.**ConnectParams(

host**=**dbServer**.**get("host", "localhost"),

port **=** dbServer**.**get("port", 1521),

service\_name**=**dbServer**.**get("service\_name"))

self**.**\_user **=** dbUsername

self**.**\_password **=** dbPassword

**def** info(self):

**return** f'Server:{self**.**\_server["host"]}:{self**.**\_server["port"]}, DbName:{self**.**\_server["service\_name"]}, DbUsername:{self**.**\_user}, DbPassword:{self**.**\_password}'

**def** connect(self):

**return** ora**.**connect(user**=**self**.**\_user, password**=**self**.**\_password, params**=**self**.**\_server)

*# Thực thi câu lệnh SQL => trả về dữ liệu => DataFrame*

**async** **def** exec\_sql\_df(self, slqQuery, arraysize**=**10000):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

**if** arraysize:

cursor**.**arraysize **=** arraysize

cursor**.**execute(slqQuery)

**return** cursor2df(cursor)

**except** ora**.**Error **as** error:

**raise** Exception(error)

*# Thực thi câu lệnh SQL => trả về dữ liệu => DataFrame*

**async** **def** exec\_sql\_df2(self, slqQuery, arraysize**=**10000):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

**if** arraysize:

cursor**.**arraysize **=** arraysize

cursor**.**execute(slqQuery)

**return** cursor2df2(cursor)

**except** ora**.**Error **as** error:

**raise** Exception(error)

*# Thực thi một thủ tục Stored Procedure => trả về dữ liệu => DataFrame*

**async** **def** exec\_sp\_df(self, spName, **\***args):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

outCursor **=** connection**.**cursor()

cursor**.**callproc(spName, args **+** (outCursor,))

**return** cursor2df(outCursor)

**except** ora**.**Error **as** error:

**raise** Exception(error)

*# Thực thi một thủ tục Stored Procedure => cập nhật/xóa => không trả về dữ liệu*

**async** **def** exec\_non\_query(self, spName, **\***args):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

outCursor **=** connection**.**cursor()

cursor**.**callproc(spName, args)

connection**.**commit()

**except** ora**.**Error **as** error:

**raise** Exception(error)

*# Thực thi một thủ tục Stored Procedure => trả về chỉ 1 giá trị*

**async** **def** exec\_scalar(self, spName, valueType, **\***args):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

outVar **=** cur**.**var(valueType)

outCursor **=** connection**.**cursor()

cursor**.**callproc(spName, args **+** (outVar,))

**return** outVar**.**getvalue()

**except** ora**.**Error **as** error:

**raise** Exception(error)

*# Thực thi một hàm Function => trả về chỉ 1 giá trị*

**async** **def** exec\_func(self, funcName, valueType, **\***args):

**try**:

**with** self**.**connect() **as** connection:

**with** connection**.**cursor() **as** cursor:

**return** cursor**.**callfunc(spName, valueType, args)

**except** ora**.**Error **as** error:

**raise** Exception(error)

*#Bug fix taken from http://kamushin.github.io/learning/python\_oracle.html*

**def** fix\_lob(row):

**def** convert(col):

**return** str(col) **if** isinstance(col, ora**.**LOB) **else** col

**return** [convert(c) **for** c **in** row]

**def** cursor2df(cursor):

records **=** [fix\_lob(row) **for** row **in** cursor]

cols **=** [n[0] **for** n **in** cursor**.**description]

**return** pd**.**DataFrame**.**from\_records(records, columns**=**cols)

**def** cursor2df2(cursor):

records **=** [row **for** row **in** cursor]

cols **=** [n[0] **for** n **in** cursor**.**description]

**return** pd**.**DataFrame**.**from\_records(records, columns**=**cols)

### Xử lý dữ liệu với DataFrame

* Xuất dữ liệu từ DataFrame ra file CSV: df.to\_csv(filename)
* Xuất dữ liệu từ DataFrame ra file Excel: df.to\_excel(filename)
* Xuất dữ liệu từ DataFrame ra SQL: df.to\_sql(table\_name, connection\_object)
* Xuất dữ liệu từ DataFrame ra JSON: df.to\_json(filename)

**Cách tạo ra dữ liệu test**

Tạo bảng dữ liệu test với Numpy và Pandas. Ví dụ tạo ra một bảng gồm 23 dòng và 4 cột, được điền vào những giá trị ngẫu nhiên từ Numpy pd.DataFrame(np.random.rand(23,4))

* Cách tạo ra series từ một list: pd.Series(my\_list)
* Cách thêm index là cột ngày tháng: df.index = pd.date\_range('1999/1/31', periods=df.shape[0])

**Cách xem và kiểm tra dữ liệu**

Kiểm tra DataFrame hoặc Series trong Panda theo một vị trí cụ thể.

* Xem n dòng đầu tiên của DataFrame: df.head(n)
* Xem n dòng cuối cùng của DataFrame: df.tail(n)
* Lấy số dòng số cột của DataFrame: df.shape
* Lấy số dòng số cột của DataFrame: df.shape
* Xem thông tin về Index, kiểu dữ liệu và dung lượng của DataFrame: df.info()
* Tổng kết thông tin thống kê cho các cột có kiểu dữ liệu là số: df.describe()
* Xem giá trị duy nhất và đếm số giá trị này, đếm cả trường hợp NA: s.value\_counts(dropna=False)
* Tổng kết giá trị duy nhất và đếm cho tất cả các cột: df.apply(pd.Series.value\_counts)
* Trích xuất một phần dữ liệu trong Pandas DataFrame

Trích xuất dữ liệu từ DataFrame hoặc Series trong Pandas

* Trả về cột có label là col như một SeriesL df[col] => Trả về một cột của DataFrame dưới dạng Series
* Trả về các cột trong danh sách dưới dạng một DataFrame mới: df[[Col1, Col2]]
* Chọn dữ liệu theo vị trí: s.iloc[0]
* Chọn dữ liệu theo index: s.loc['index\_one']
* Chọn dữ liệu ở dòng đầu tiên: df.iloc[0,:]
* Chọn dữ liệu ở dòng đầu tiên, ô thứ nhất của DataFrame: df.iloc[0,0]

**Cách làm sạch dữ liệu trong Pandas DataFrame**

Một số lệnh xử lý và làm sạch dữ liệu

* Đổi tên các cột trong DataFrame theo thứ tự: df.columns = ['a','b','c']
* Kiểm tra dữ liệu với giá trị null: pd.isnull()
* Kiểm tra dữ liệu với giá trị khác null: pd.notnull()
* Bỏ toàn bộ dòng có dữ liệu null: df.dropna()
* Bỏ toàn bộ cột có dữ liệu null: df.dropna(axis=1)
* Bỏ các dòng có nhiều hơn n giá trị null: df.dropna(axis=1, thresh=n)
* Thay toàn bộ giá trị null bằng giá trị x: df.fillna(x)
* Thay toàn bộ giá trị null bằng giá trị khác
* Thay giá trị null trong Series bởi giá trị mean của các giá trị trong Series: s.fillna(s.mean())
* Chuyển đổi kiểu dữ liệu của Series sang Float: s.astype(float)
* Thay tất cả các giá trị bằng 1 bởi one: s.replace(1,'one')
* Thay nhiều giá trị cùng lúc: s.replace([1,3],['one','three'])
* Đổi tên cột hàng loạt bằng lambda: df.rename(columns=lambda x: x + 1)
* Đổi tên cột cụ thể trong DataFrame: df.rename(columns={'old\_name': 'new\_ name'})
* Đổi index trong DataFrame: df.set\_index('column\_one')
* Đổi index hàng loạt trong DataFrame: df.rename(index=lambda x: x + 1)

## Cách lọc, sắp xếp, nhóm dữ liệu trong Pandas DataFrame

### Lọc dữ liệu theo điều kiện:

* Lọc ra các dòng thỏa mãn điều kiện col lớn hơn 5: df[ df[col] > 5 ]
* Lọc ra các dòng thỏa mãn điều kiện: có giá trị cột col trong khoảng 100 đến 200: df[ df[col] > 100 & df[col] < 200 ]

### Sắp xếp dữ liệu:

* Sắp xếp dữ liệu trong cột col1 theo chiều thuận (ascending): df.sort\_values(col1)
* Sắp xếp dữ liệu trong cột col2 theo chiều nghịch (descending): df.sort\_values(col2, ascending=False)
* Sắp xếp col1 theo chiều thuận và col2 theo chiều nghịch: df.sort\_values([col1,col2],ascending=[True,False])

### Nhóm dữ liệu, pivot dữ liệu với groupby

* Pivot dữ liệu theo col1: df.groupby(col1)
* Pivot dữ liệu theo nhiều cột col1, col2
* Pivot dữ liệu với pivot\_table trong DataFrame
* Tạo một Pivot Table, nhóm dữ liệu theo cột col1, tính mean của col2, col3: df.pivot\_table(index=col1,values=[col2,col3],aggfunc=mean)
* Tính mean của tất cả các cột: df.apply(np.mean)
* Tính max mỗi dòng: df.apply(np.max,axis=1)

### Nối dữ liệu, gộp dữ liệu bằng Pandas DataFrame

* Nối dữ liệu DataFrame theo chiều dọc: df1.append(df2) => Nối các dòng của df1 xuống dưới df2 (Số lượng các cột trong 2 DataFrames này phải giống nhau)
* Nối dữ liệu DataFrame theo chiều ngang: pd.concat([df1, df2], axis=1) => Nối các cột của df1 sang phải các cột của df2 (Số lượng các dòng trong 2 DataFrames này phải giống nhau)
* Join dữ liệu 2 DataFrames theo kiểu SQL: df1.join(df2,on=col1,how='inner') => Join 2 DataFrame df1 và df2 theo cột chung col1, kiểu Join là 'inner'. Ngoài ra còn hỗ trợ các kiểu join: 'left', 'right', 'outer'

### Thống kê dữ liệu bằng Pandas DataFrame

* Thống kê dữ liệu cho các cột số: df.describe()
* Tính mean cho tất cả các cột: df.mean()
* Tính correlation giữa các cột: df.corr()
* Đếm số giá trị không null cho các cột: df.count()
* Tìm giá trị lớn nhất cho mỗi cột: df.max()
* Tìm giá trị nhỏ nhất cho mỗi cột: df.min()
* Tìm giá trị median cho mỗi cột: df.median()
* Tìm giá trị độ lệch tiêu chuẩn cho mỗi cột: df.std()

## Các thư viện trực quan hóa dữ liệu

Python có rất nhiều thư viện ttrựcquan hóa dữ liệu, trong tiểu luận này trình bày 2 loại cơ bản:

### Thư viện Matplotlib

Matplotlib là một gói Python được sử dụng cho đồ họa 2D. Matplotlib có 2 objects chính:

* Figure: chứa nhiều axes
* Axes: nơi vẽ đồ thị, thực tế chính là cái đồ thị

Figure giống khung ảnh và 1 khung ảnh có thể không chứa hoặc chứa 1 đến nhiều bức ảnh, và Axes tương ứng với những bức ảnh trong cái khung ảnh.

Các thành phần khác:

* Title: tên của đồ thị
* X axis label,Y axis label: 2 trục (ngang, đứng) của đồ thị
* Major tick label, Minor tick label: Tick chính chứa con số, tick nhỏ có thể hoặc không chứa con số
* Legend: Chú thích
* Grid: Lưới (dễ căn chỉnh)
* Line plot: Đồ thị đường
* Scatter plot: Đồ thị phân tán
* Marker: Những chấm tròn biểu diễn điểm dữ liệu trong đồ thị phân tán
* Axis spine: Đường ghi nhận ranh giới khu vực dữ liệu
* Spine: Đường kết nối các dấu ticks trên trục và ghi nhận ranh giới của khu vực dữ liệu.

### Thư viện Seaborn

Seaborn là thư viện mở rộng được viết trên nền Matplotlib, từ đó kế thừa các chức năng biểu diễn dữ liệu từ Matplotlib. Vì vậy, những gì Seaborn làm được thì Matplotlib cũng làm được tuy nhiên không có chiều ngược lại, cũng có nghĩa là Seaborn không thể thay thế hoàn toàn được Matplotlib (Seaborn hiểu đơn giản là 1 đứa con được Matplotlib tạo ra).

Mỗi thư viện đều có những ưu điểm riêng: Seaborn sử dụng ít cú pháp hơn và có các chủ đề mặc định khá đẹp, Matplotlib dễ dàng tùy chỉnh hơn thông qua việc truy cập các lớp.

* Cài đặt: pip install matplotlib, seaborn

In [6]:

*# Sử dụng (Import)*

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

sns**.**set\_style('whitegrid')

**%matplotlib** inline

**import** warnings

warnings**.**filterwarnings('ignore')

**import** pandas **as** pd

**import** numpy **as** np

pd**.**DataFrame**.**iteritems **=** pd**.**DataFrame**.**items

## Các kiểu trực quan hoá dữ liệu cơ bản

### Đồ thị đường

Biểu đồ đường được sử dụng để theo dõi những thay đổi trong khoảng thời gian ngắn và dài.

In [7]:

*# Đồ thị 1 đường*

*# Ví dụ: Vẽ đồ thị hình sin của 100 điểm trong khoảng từ -10 đến 10.*

x **=** np**.**linspace(**-**10,10,100)

y **=** np**.**sin(x)

*# Mặc định plt.plot() là đồ thị đường*

plt**.**plot(x,y,linestyle**=**'-.') *# linestyle để thay đổi đường biểu diễn, ví dụ: linestyle=(0,(3,1,1,1)) tương ứng linestyle='-.'*

plt**.**title('Sin function') *# Tên của đồ thị*

plt**.**xlabel('X value') *# Tên trục hoàng*

plt**.**ylabel('Sin of x') *# Tên trục tung*

plt**.**grid() *# Tạo lưới*

plt**.**show() *# Hiển thị biểu đồ*

A graph of a function

AI-generated content may be incorrect.

In [8]:

*# Ví dụ: Vẽ đồ thị hình sin và hình cos của 100 điểm trong khoảng từ -10 đến 10 trên cùng 1 đồ thị.*

x **=** np**.**linspace(**-**10,10,100)

sin **=** np**.**sin(x)

cos **=** np**.**cos(x)

df **=** pd**.**DataFrame({'x':x,'sin': sin, 'cos':cos})

plt**.**figure(figsize**=**(15,5))

ax1 **=** plt**.**subplot(1,2,1)

ax2 **=** plt**.**subplot(1,2,2)

ax1**.**plot(x,sin)

ax2**.**plot(x,cos,c**=**'red')

ax1**.**set(xlabel**=**'X\_label', ylabel**=**'Y\_label', title**=**'Sin Function')

ax2**.**set(xlabel**=**'X\_label', ylabel**=**'Y\_label', title**=**'Cos Function');

plt**.**show()

A graph of a graph of a function

AI-generated content may be incorrect.

In [9]:

*# Tải các dữ liệu mẫu*

tips **=** sns**.**load\_dataset('tips')

iris **=** sns**.**load\_dataset('iris')

### Biểu đồ tròn (Pie chart)

In [10]:

sex **=** tips**.**groupby('sex')**.**total\_bill**.**sum()

plt**.**pie(sex, labels **=** sex**.**index, autopct**=**'%.2f%%')

plt**.**show();

A blue and orange pie chart

AI-generated content may be incorrect.

In [11]:

*# Nâng cao hơn => tùy chỉnh các format*

male **=** tips[tips['sex'] **==** 'Male']**.**groupby('day')**.**total\_bill**.**sum()

plt**.**figure(figsize**=**(6,6))

plt**.**pie(male, labels **=** male**.**index, autopct**=**'%.2f%%', explode**=**[0,0.1,0,0])

plt**.**show();

A pie chart with different colored sections

AI-generated content may be incorrect.

### Biểu đồ cột (Bar)

In [12]:

df **=** iris**.**groupby('species')['sepal\_length']**.**sum()**.**to\_frame()**.**reset\_index()

plt**.**bar(df['species'],df['sepal\_length'],color **=** ['cornflowerblue','lightseagreen','steelblue'])

plt**.**title('Bar Chart')

plt**.**xlabel('Species')

plt**.**ylabel('sepal\_length')

plt**.**show()

A bar chart with different colored bars

AI-generated content may be incorrect.

In [ ]:

### Đồ thị phân phối (Histogram/Distribution)

In [13]:

plt**.**hist(iris['petal\_length'], bins **=** 10)

plt**.**show()

A graph with blue bars

AI-generated content may be incorrect.

In [14]:

*# Sử dụng Seaborn*

sns**.**histplot(iris, x **=** 'petal\_length', bins **=** 10)

plt**.**show()

A graph with blue bars

AI-generated content may be incorrect.

### Một số biểu đồ phức hợp khác

**Pairplot**

Khi muốn nhìn tổng quan dữ liệu và mối tương quan giữa các chiều dữ liệu theo từng cặp với nhau, thì pair plots là lựa chọn vô cùng hoàn hảo. Ta có thể thử nghiệm đơn giản với các cột dữ liệu của tập iris.

In [15]:

sns**.**pairplot(iris, hue**=**'species', height**=**1.5)

plt**.**show()

A screenshot of a graph

AI-generated content may be incorrect.

**Đồ thị hộp (Box plot)**

In [16]:

plt**.**boxplot([iris**.**petal\_length,iris**.**petal\_width,iris**.**sepal\_length,iris**.**sepal\_width])

plt**.**show()

A graph of a graph

AI-generated content may be incorrect.

In [17]:

*# Seaborn tạo ra biểu đồ mặc định nhìn đẹp hơn*

sns**.**boxplot(data**=**iris, width**=**0.8)

plt**.**show()

A diagram of different colored squares

AI-generated content may be incorrect.

**Biểu đồ Violin**

In [18]:

sns**.**violinplot(data**=**iris) *#Thêm phân bố bên ngoài*

plt**.**show()

A graph showing different colored shapes

AI-generated content may be incorrect.

Ngoài ra còn rất nhiều lại biểu đồ khác

**Đồ thị phân tán (Scatter plot)**

Biểu đồ hay đồ thị phân tán là một đồ thị biểu hiện mối tương quan giữa nguyên nhân và kết quả hoặc giữa các yếu tố ảnh hưởng đến chất lượng. Ví dụ: Vẽ đồ thị phân tán cánh hoa và đài hoa

In [19]:

plt**.**scatter(iris**.**petal\_length, iris**.**petal\_width, label**=**'Petal')

plt**.**scatter(iris**.**sepal\_length, iris**.**sepal\_width, marker**=**'\*', label **=** 'Sepal')

plt**.**legend()

plt**.**show()

A screen shot of a graph

AI-generated content may be incorrect.