1.

'is' and '==' in python. Python beginners always make mistakes when working with 'is' and '==='. Let's understand this with a simple example.

Consider this code:
```
x=[1,2,3]
y=x
z=[1,2,3]
print(x is y)
print(x is z)
print(y is z)
```

The answer you will get:
True
False
False

But in this case:
```
x=[1,2,3]
y=x
z=[1,2,3]
print(x==y)
print(y==z)
print(x==z)
```

The answer will be:
True
True
True

So why this difference?
Let's check the memory addresses of all these variables in hexadecimal format

```
print(hex(id(x),hex(id(y)),hex(id(z))
```

and this statement print these addresses
('0x1b8ced53e48', '0x1b8ced53e48', '0x1b8c8117c48')

We can see that addresses of both x and y are the same while it is different for z. So 'is'

command compares the addresses, not values, however '==' compares the values. 👍

Good night 🤗

2.

Avoid bad practice in python. Never import everything from a package in python like
from numpy import *

It may create a potential conflict with other functions. For instance consider this code:

```
from numpy import *
from math import *
x=linspace(-2*pi,2*pi,100)
y=sin(x)
```

Since "pi,sin" presents in both the libraries and we imported #math after #numpy so the above script will use "sin" from math library instead of "numpy" library and it will return an error.

Best Practice:

```
import numpy as np
import math as m
x=linspace(-2*np.pi,2*np.pi,100)
y=np.sin(x)
```

3.

Listing all your .csv or any other files from your current directory and sub-directories in one line code.

```
import glob
all_files=glob.glob('**/*.csv')
```

or from a specified path

```
path=r'C:\Users\UT\Desktop\Python\**\*.csv'
all_files=glob.glob(path)
```

4.
Use a #ternary #operator in place of simple if-else to increase the readability of the code.

Instead of:

```
x=-10
a=5
b=7
if x<0:
    f=abs(x)-a
else:
    f=x-b
```

Use:

```
f=abs(x)-a if x<0 else x-b
```

It is same as writing

f(x)=|x|-a if x<0 else x-b

General Syntax of ternary operator:
[True Condition] if [Expression] else [False Condition]

5.
How to calculate the #numerical #derivative in #python?
Suppose we have a function f(x) then we can calculate the derivative of f(x) using:

$$f'(x)=\lim_{dx\to 0}\frac{(f(x+dx)-f(x))}{dx}$$

Here is the way to calculate using python:

```
import numpy as np
import matplotlib.pyplot as plt
f=lambda x:np.sin(x)
dx=0.01
x=np.linspace(-2*np.pi,2*np.pi,100)
difx=(f(x+dx)-f(x))/dx
plt.plot(x,f(x),linewidth=2,label='sin(x)')
plt.plot(x,difx,linewidth=2,label='cos(x)')
plt.legend(loc='best')
```

Can you create a function to calculate the nth derivative of f(x)?