姓名：孔启楠

学号：2020155036

Problem1:

P65 4.We defined the tensor `X` of `shape (2, 3, 4)` in this section. What is the output of `len(X)` ?

翻译：本节中定义了形状的张量`X(2 , 3, 4)`,`len(X)`的输出结果是什么?

In [5]:
```
import torch
X = torch.arange(24).reshape(2, 3, 4)
print(X)
print("长度: "+str(len(X)))
```

```
tensor([[[ 0,  1,  2,  3],
         [ 4,  5,  6,  7],
         [ 8,  9, 10, 11]],

        [[12, 13, 14, 15],
         [16, 17, 18, 19],
         [20, 21, 22, 23]]])
长度: 2
```

Problem2:

P65 5.For a tensor `X` of arbitrary shape, does `len(X)` always correspond to the length of a certain axis of `X` ? What is that axis?

翻译：对于任意形状的张量`X`,`len(X)`是否总是对应于X特定轴的长度?这个轴是什么?

In [8]:
```
import torch
X = torch.tensor([3.0,2.0,1.0])
X1 = torch.tensor([[3.0],[2.0],[1.0]])
print("张量X的长度: "+str(len(X)))
print("张量X1的长度: "+str(len(X1)))
```

```
张量X的长度: 3
张量X1的长度: 3
```

分析：对于任意形状的张量，`len(X)`的值是第一个维度的大小,即x轴。

Problem3:

P65 6.Run `A / A.sum(axis=1)` and see what happens. Can you analyze the reason?

翻译：运行`A/A.sum(axis=1)`，看看会发生什么。请分析一下原因？

In [11]:
```
import torch
A = torch.arange(20, dtype=torch.float32).reshape(5, 4)
print(A/A.sum(axis=1,keepdims=True))
```

```
tensor([[0.0000, 0.1667, 0.3333, 0.5000],
        [0.1818, 0.2273, 0.2727, 0.3182],
        [0.2105, 0.2368, 0.2632, 0.2895],
        [0.2222, 0.2407, 0.2593, 0.2778],
        [0.2286, 0.2429, 0.2571, 0.2714]])
```

分析： `sum`函数是求和函数，若指定了参数axis，那么将对该轴进行求和，题目中制定了`axis=1`，即列求和。因此`A.sum`得到的是每一行求和的结果的一个张量。而`keepdims`则表示结果张量和输入张量的维度保持不变，下面是`A.sum`的输出，可以看到，是一个二维张量。每行代表原始张量在该行的求和。

In [12]:
```
import torch
A = torch.arange(20, dtype=torch.float32).reshape(5, 4)
print(A.sum(axis=1,keepdims=True))
```

```
tensor([[ 6.],
        [22.],
        [38.],
        [54.],
        [70.]])
```

Problem4:

P65 8.Consider a tensor with `shape (2, 3, 4)`. What are the shapes of the summation outputs along axis 0, 1, and 2?

翻译：考虑一个具有形状`shape (2, 3, 4)`的张量，在轴0、1、2上的求和输出是什么形状?

In [18]:
```
import torch
A = torch.arange(24, dtype=torch.float32).reshape(2,3,4)
print("在轴0上求和输出的形状： "+str(A.sum(axis=0).shape))
print("在轴1上求和输出的形状： "+str(A.sum(axis=1).shape))
print("在轴2上求和输出的形状： "+str(A.sum(axis=2).shape))
```

```
在轴0上求和输出的形状：torch.Size([3, 4])
在轴1上求和输出的形状：torch.Size([2, 4])
在轴2上求和输出的形状：torch.Size([2, 3])
```

Problem5:

P65 9.Feed a tensor with 3 or more axes to the linalg.norm function and observe its output. What does this function compute for tensors of arbitrary shape?

翻译：为linalg.norm函数提供3个或更多轴的张量，并观察其输出。对于任意形状的张量这个函数计算得到什么?

In [4]:
```
import torch
import numpy as np
A = torch.arange(24, dtype=torch.float32).reshape(2,3,4)
print(np.linalg.norm(A))
```

```
65.757126
```

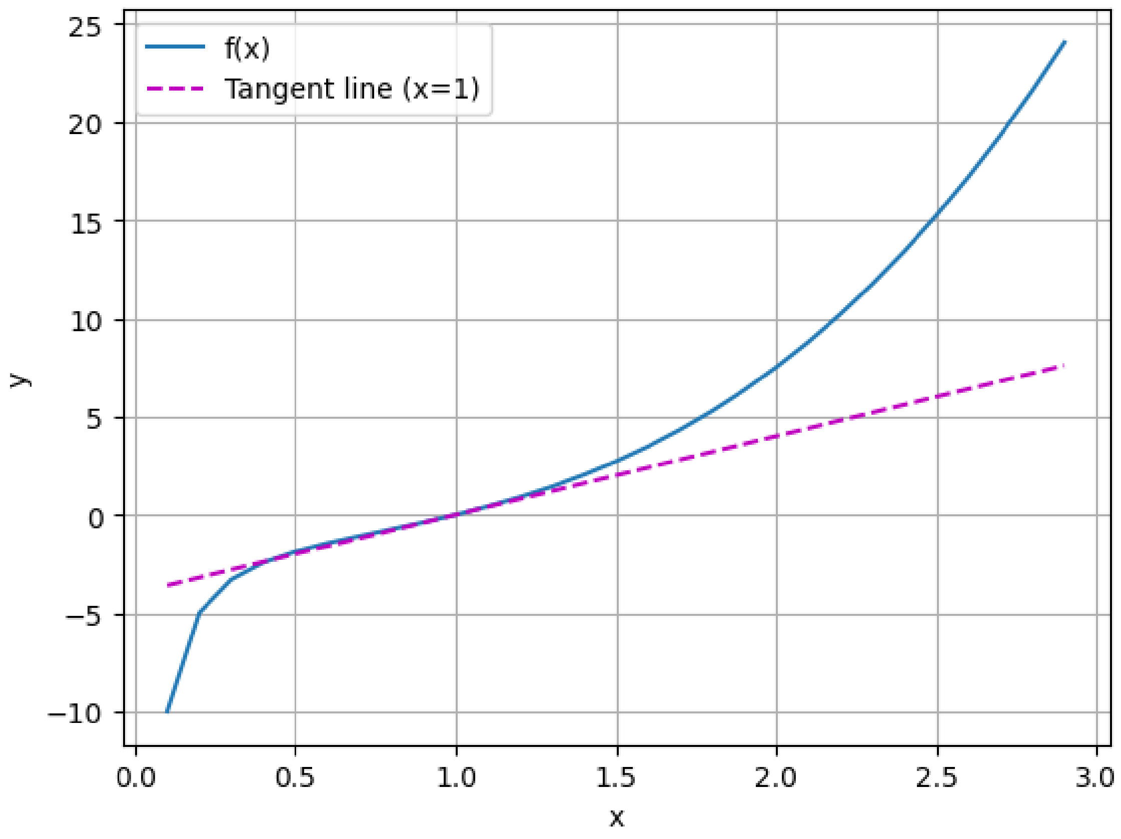
分析：`np.linalg.norm(x, ord=None, axis=None, keepdims=False)` 函数用于求范数。其中参数 `x` 表示需要求范数的张量，参数 `ord` 表示范数类型，默认是二范数，参数 `axis` 表示以哪个轴来求范数，默认为 `none`，表示求整个张量的范数，参数 `keepdims` 表示是否保存和原张量一样的维度

Problem6:

P71 1.Plot the function `y = f(x) = x^3-1/x` and its tangent line when `x = 1`.

翻译：画出函数 `y = f(x) = x^3-1/x` 以及它在 `x = 1` 时的切线

In [1]:
```
%matplotlib inline
import numpy as np
from matplotlib_inline import backend_inline
from d2l import torch as d2l
def f(x):
    return x**3 - 1 / x
X = np.arange(0.1, 3, 0.1)
Y0 = f(X)  #计算函数
Y1 = 4*X-4  #计算x = 1 时的切线
X=[X]
Y = [Y0,Y1]
if len(X)!=len(Y):
    X=X*len(Y)
axis = d2l.plt.gca()
fmts=('-', 'm--', 'g-.', 'r:')
for x,y,fmt in zip(X,Y,fmts):
    axis.plot(x,y,fmt)
axis.legend(['f(x)', 'Tangent line (x=1)'])
axis.set_xlabel("x")
axis.set_ylabel("y")
axis.grid()
```



Problem7

P71 2.Find the gradient of the function $f(\mathbf{x}) = 3x_1^2 + 5e^{x_2}$

翻译：求函数$f(\mathbf{x}) = 3x_1^2 + 5e^{x_2}$的梯度

解：$\nabla f(\mathbf{x}) = [6x_1, 5e^{x_2}]$

Problem8

P75 2.After running the function for backpropagation, immediately run it again and see what happens.

翻译：在运行反向传播函数之后，立即再次运行它，看看会发生什么。

In [ ]:
```
import torch
x = torch.arange(4.0)
x.requires_grad=True
print(x)
y = 2 *torch.dot(x,x)
y.backward()
y.backward()
print(x.grad)
```

运行时错误： Trying to backward through the graph a second time (or directly access saved tensors after they have already been freed). Saved intermediate values of the graph are freed when you call .backward() or autograd.grad(). Specify retain_graph=True if you need to backward through the graph a second time or if you need to access saved tensors after calling backward.

分析：在第一次进行backward之后，计算的中间变量已被释放，再次backward便会出错。如果想再次backward，可以在第一次backward时，加上参数retain_graph=True。

Problem9

P75 3.In the control flow example where we calculate the derivative of d with respect to a, what would happen if we changed the variable a to a random vector or matrix. At this point, the result of the calculation f(a) is no longer a scalar. What happens to the result? How do we analyze this?

翻译：在控制流的例子中，我们计算d关于a的导数，如果将变量a更改为随机向量或矩阵，会发生什么，并分析?

In [ ]:
```python
import torch
def f(a):
    b = a * 2
    while b.norm() < 1000:
        b = b * 2
    if b.sum() > 0:
        c = b
    else:
        c = 100 * b
    return c
a = torch.randn(size=[3], requires_grad=True)  #生成一个3维向量
d = f(a)
d.backward()
print(a.grad == d/a)
```

运行时错误： grad can be implicitly created only for scalar outputs

分析：当函数值为向量时，不能直接调用backward。否则会出错。可以在backward中附带参数或者如下：

In [4]:
```python
import torch
def f(a):
    b = a * 2
    while b.norm() < 1000:
        b = b * 2
    if b.sum() > 0:
        c = b
    else:
        c = 100 * b
    return c
a = torch.randn(size=[3], requires_grad=True)
d = f(a)
d.sum().backward()   #先调用sum把向量改为标量, 在求导
print(a.grad == d/a)
```

```
tensor([True, True, True])
```

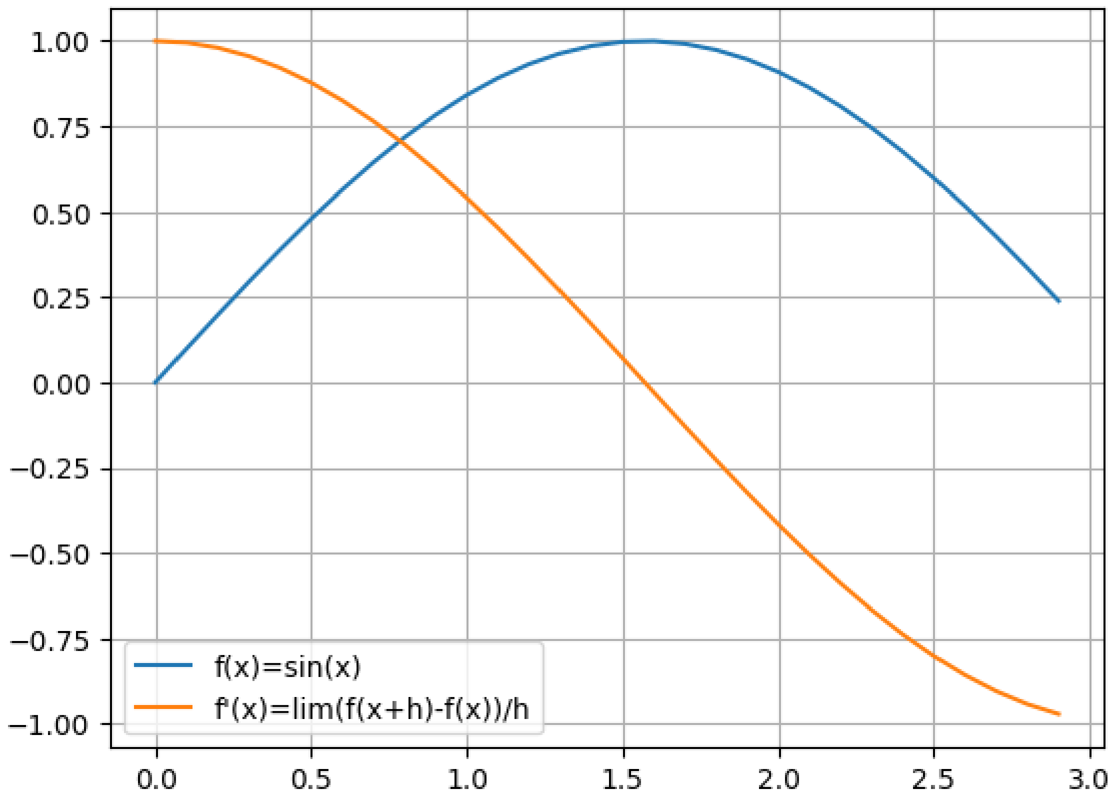
Problem10

P75 5.使得f(x)=sin(x)，绘制f(x)和df(x)/dx的图像，其中后者不使用f'(x)=cos(x)

分析：使用导数的定义求出f'(x)

In [15]:
```
import matplotlib.pyplot as plt
import numpy as np
import torch

def f(x):
    return np.sin(x)
def get_derivative(x):
    h=1e-4
    return (f(x+h)-f(x))/h
legend=['f(x)=sin(x)',"f'(x)=lim(f(x+h)-f(x))/h"]
axis=plt.gca()
X=np.arange(0,3,0.1)
Y=f(X)
axis.plot(X,Y)
axis.plot(X,get_derivative(X))
axis.legend(legend)
axis.grid()
```


f(x)=sin(x)
f'(x)=lim(f(x+h)-f(x))/h