

实验三：调试及性能分析、元编程演示实验、PyTorch编程学习实验报告

邓林 23020007014

中国海洋大学 23软件工程

摘要

本实验报告主要记录了作者通过课程网站及B站学习调试及性能分析、元编程演示实验、PyTorch编程的学习过程以及心得。

1 实验内容

pdb指令	
python -m pdb filename	进入pdb
list/l	显示当前行附近的 11 行或继续执行之前的显示
list ./l .	列出当前函数的代码
long list/ll	显示所有代码
next/n	继续执行直到当前函数的下一条语句或者 return 语句
step/s	执行当前行，并在第一个可能的地方停止
continue/c	执行代码到下一个断点
break 文件行数	设置断点（基于传入的参数）
break	显示所有断点
cl 序号	清除指定断点
cl	清除所有断点
disable	暂停使用指定断点
enable	恢复使用指定断点
until 行号	执行代码到指定行号
p	在当前上下文对表达式求值并打印结果
r	继续执行直到当前函数返回
q	退出调试器

1.1 调试及性能分析

1. 使用日志

```
18501@C:\Users\18501: MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ python logger.py
2
4
6

18501@C:\Users\18501: MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ python logger.py log
2
4
6
```

图 1: 空和 log

```
18501@C:\Users\18501: MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ python logger.py log ERROR
2
4
6

18501@C:\Users\18501: MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ python logger.py color
2
4
6
```

图 2: log ERROR 和 color

2. 改变输出结果的颜色

```
18501@C:\Users\18501: MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ vim color.sh

18501@C:\Users\18501: MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ ./color.sh
```

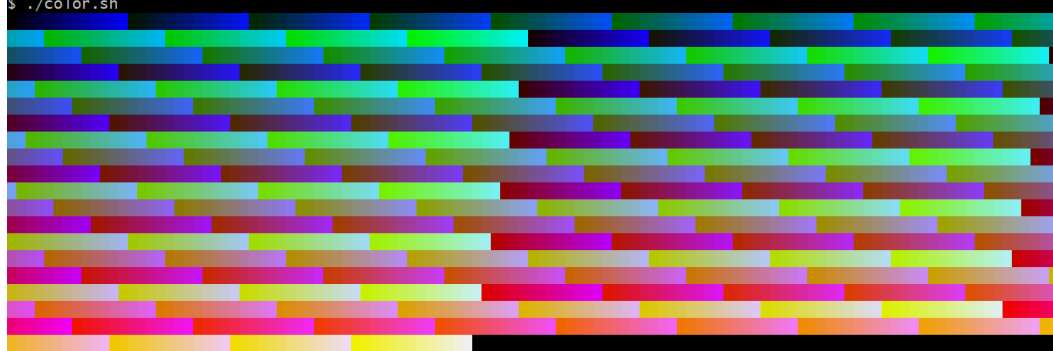
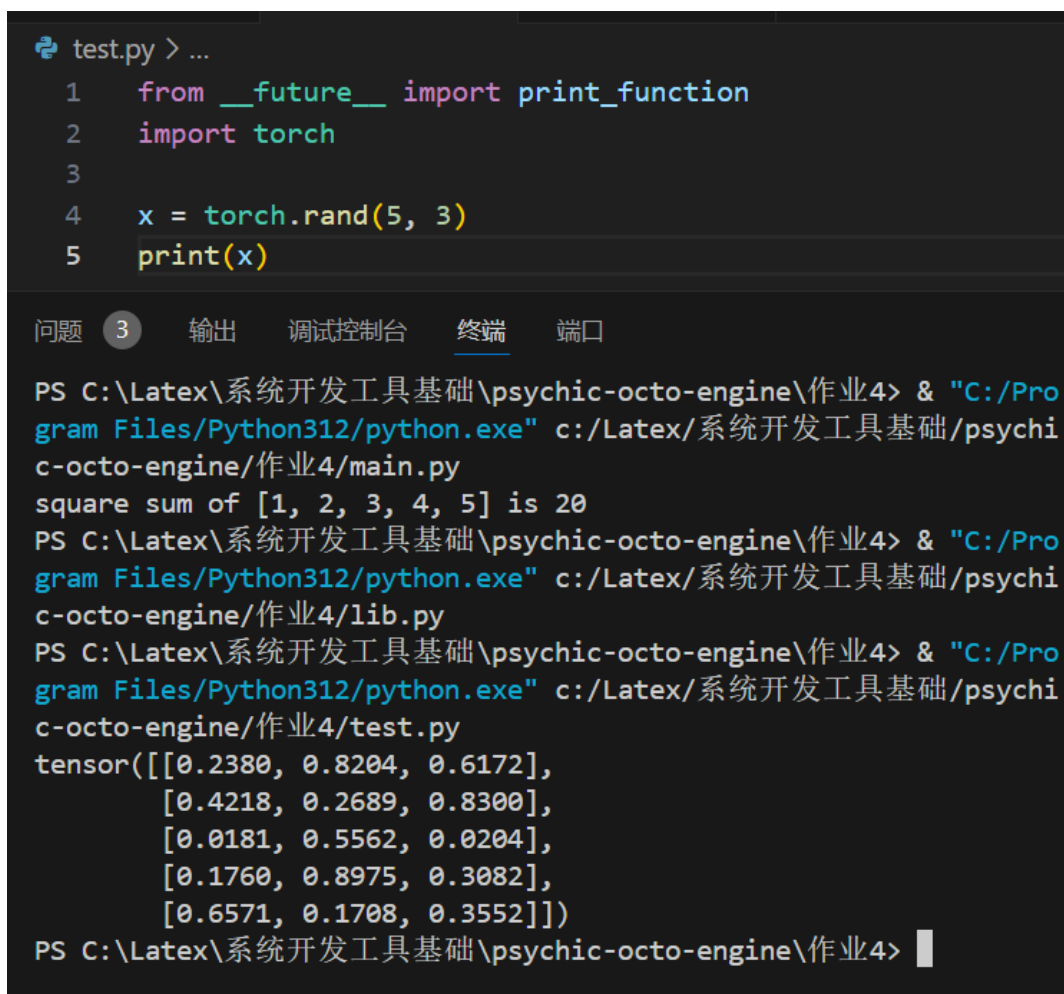


图 3: 在终端中打印多种颜色

1.2 PyTorch编程

1. 下载PyTorch并配置环境，调试



The image shows a code editor window with a file named `test.py` containing the following Python code:

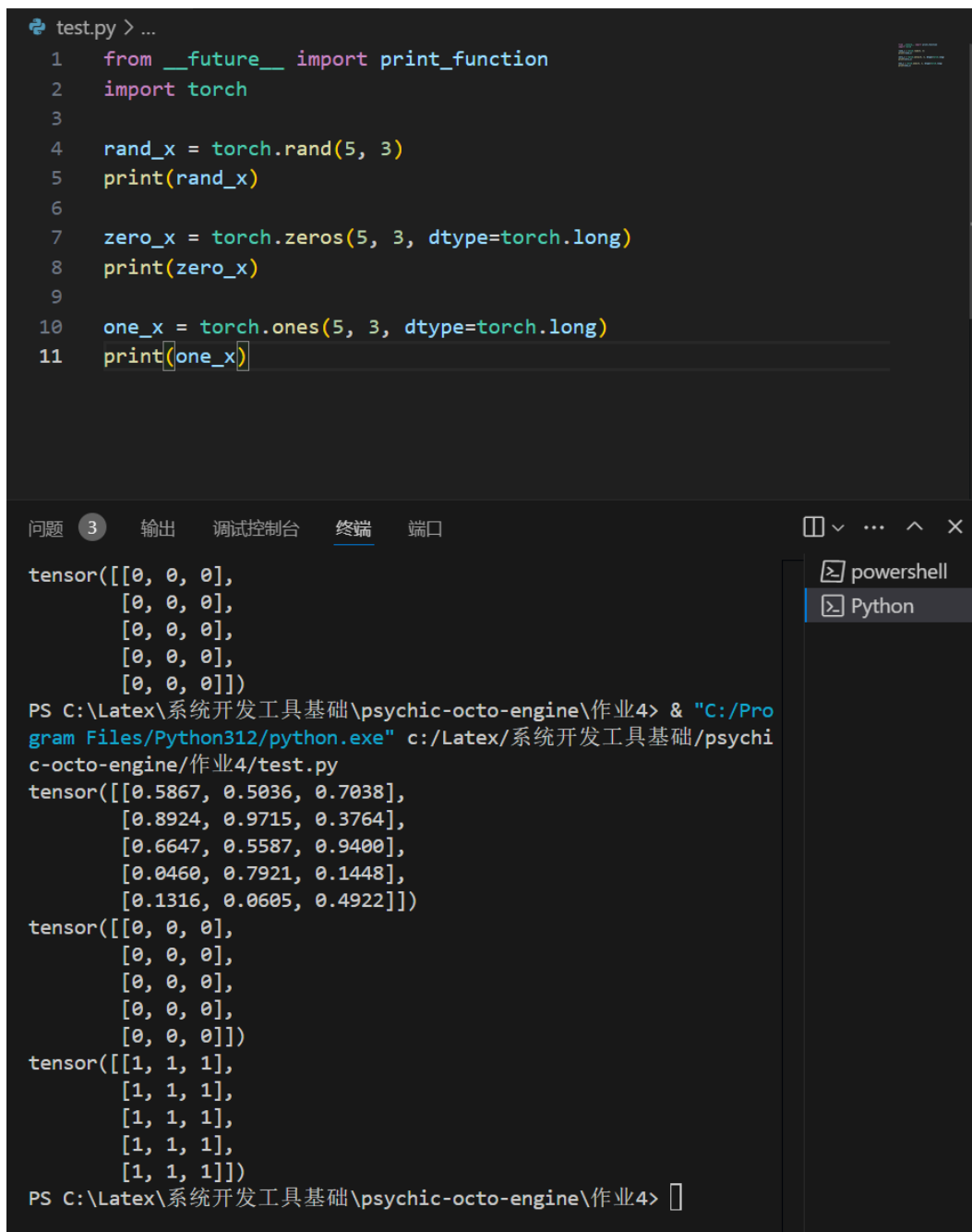
```
1 from __future__ import print_function
2 import torch
3
4 x = torch.rand(5, 3)
5 print(x)
```

Below the code editor is a terminal window with tabs for "问题", "3", "输出", "调试控制台", "终端", and "端口". The "终端" tab is active, showing the following commands and output:

```
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/main.py
square sum of [1, 2, 3, 4, 5] is 20
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/lib.py
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([[0.2380, 0.8204, 0.6172],
        [0.4218, 0.2689, 0.8300],
        [0.0181, 0.5562, 0.0204],
        [0.1760, 0.8975, 0.3082],
        [0.6571, 0.1708, 0.3552]])
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> |
```

图 4: 安装PyTorch

2. 张量(Tensors)



The image shows a code editor window with a Python script named `test.py` and a terminal window below it. The script defines three tensors: `rand_x` (random values), `zero_x` (zeros), and `one_x` (ones). The terminal shows the output of these operations.

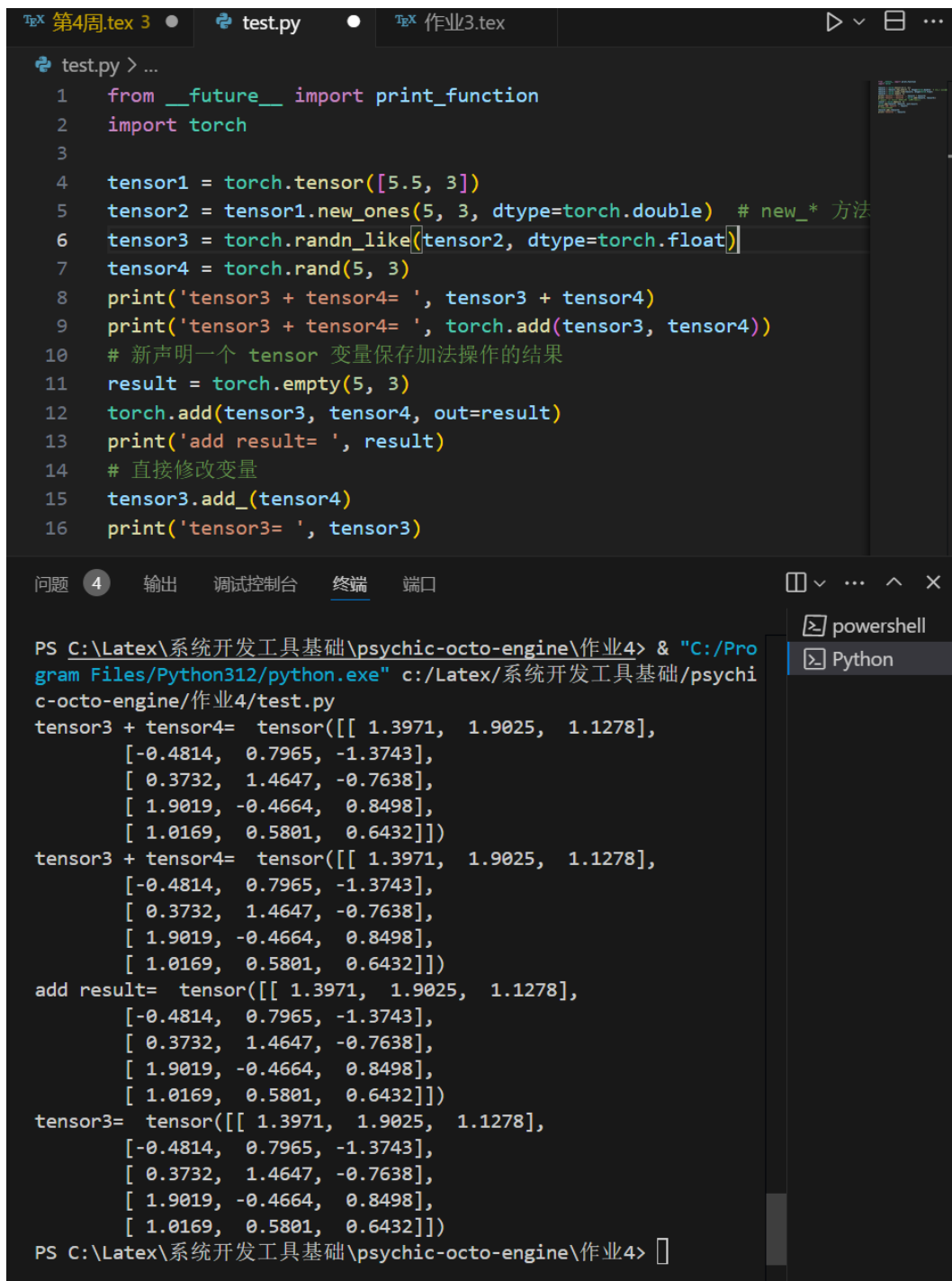
```
test.py > ...
1  from __future__ import print_function
2  import torch
3
4  rand_x = torch.rand(5, 3)
5  print(rand_x)
6
7  zero_x = torch.zeros(5, 3, dtype=torch.long)
8  print(zero_x)
9
10 one_x = torch.ones(5, 3, dtype=torch.long)
11 print(one_x)
```

Terminal Output:

```
tensor([[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]])
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([[0.5867, 0.5036, 0.7038],
        [0.8924, 0.9715, 0.3764],
        [0.6647, 0.5587, 0.9400],
        [0.0460, 0.7921, 0.1448],
        [0.1316, 0.0605, 0.4922]])
tensor([[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]])
tensor([[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]])
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> 
```

图 5: `torch.zeros()`: 创建数值皆为 0 的矩阵

3. 操作(Operations)



The image shows a code editor with a file named `test.py` and a terminal window below it. The code in `test.py` defines two 5x3 tensors, `tensor3` and `tensor4`, and performs several operations on them. The terminal output shows the results of these operations.

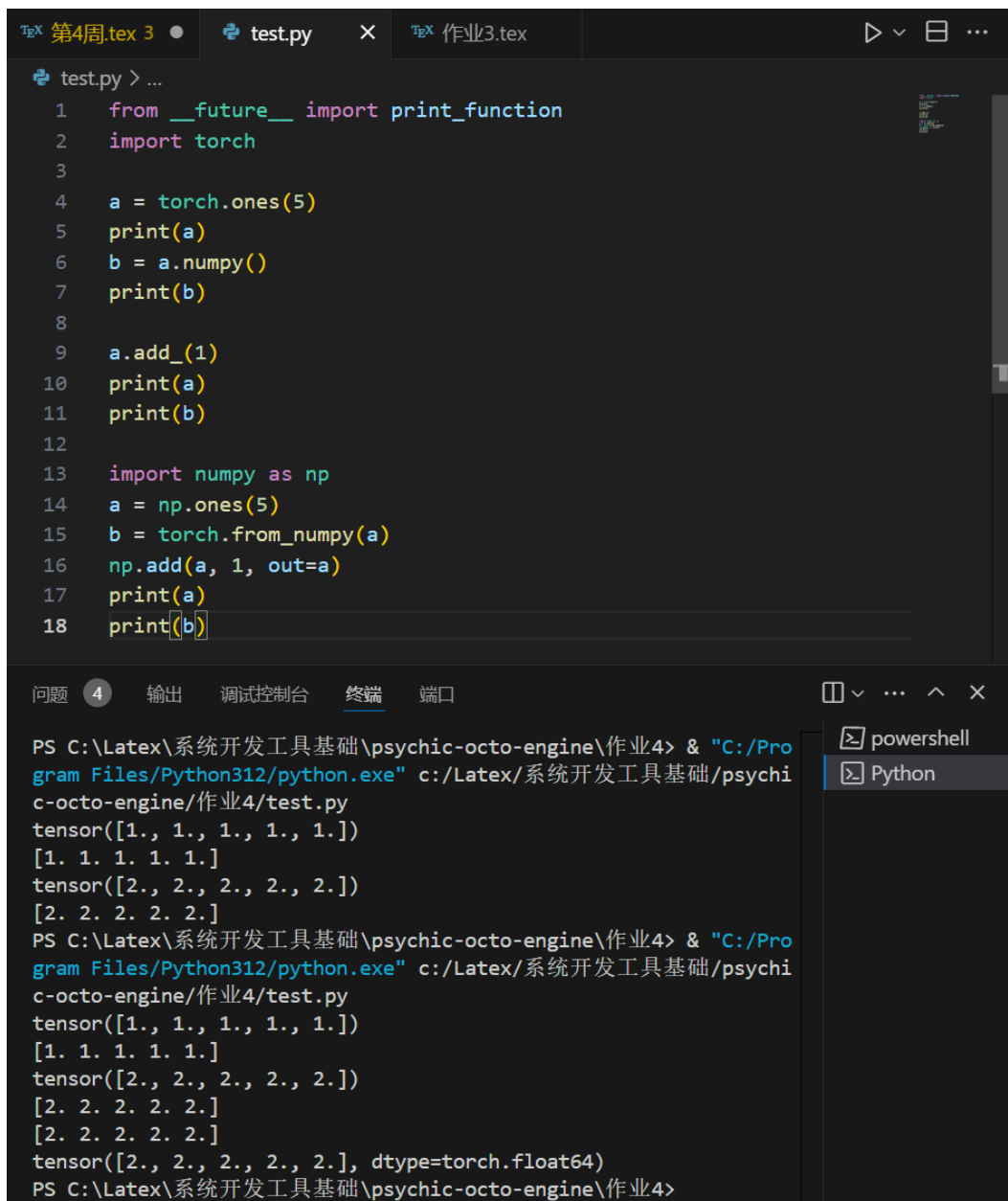
```
test.py > ...
1  from __future__ import print_function
2  import torch
3
4  tensor1 = torch.tensor([5.5, 3])
5  tensor2 = tensor1.new_ones(5, 3, dtype=torch.double) # new_* 方法
6  tensor3 = torch.randn_like(tensor2, dtype=torch.float)
7  tensor4 = torch.rand(5, 3)
8  print('tensor3 + tensor4= ', tensor3 + tensor4)
9  print('tensor3 + tensor4= ', torch.add(tensor3, tensor4))
10 # 新声明一个 tensor 变量保存加法操作的结果
11 result = torch.empty(5, 3)
12 torch.add(tensor3, tensor4, out=result)
13 print('add result= ', result)
14 # 直接修改变量
15 tensor3.add_(tensor4)
16 print('tensor3= ', tensor3)
```

Terminal Output:

```
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor3 + tensor4= tensor([[ 1.3971,  1.9025,  1.1278],
                        [-0.4814,  0.7965, -1.3743],
                        [ 0.3732,  1.4647, -0.7638],
                        [ 1.9019, -0.4664,  0.8498],
                        [ 1.0169,  0.5801,  0.6432]])
tensor3 + tensor4= tensor([[ 1.3971,  1.9025,  1.1278],
                        [-0.4814,  0.7965, -1.3743],
                        [ 0.3732,  1.4647, -0.7638],
                        [ 1.9019, -0.4664,  0.8498],
                        [ 1.0169,  0.5801,  0.6432]])
add result= tensor([[ 1.3971,  1.9025,  1.1278],
                    [-0.4814,  0.7965, -1.3743],
                    [ 0.3732,  1.4647, -0.7638],
                    [ 1.9019, -0.4664,  0.8498],
                    [ 1.0169,  0.5801,  0.6432]])
tensor3= tensor([[ 1.3971,  1.9025,  1.1278],
                 [-0.4814,  0.7965, -1.3743],
                 [ 0.3732,  1.4647, -0.7638],
                 [ 1.9019, -0.4664,  0.8498],
                 [ 1.0169,  0.5801,  0.6432]])
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4>
```

图 6: 效果如图

4. 和 Numpy 数组的转换



The image shows a VS Code editor window with a Python file named `test.py` and a terminal window below it. The Python script defines two tensors, `a` and `b`, and performs operations on them. The terminal shows the output of the script, including the creation of tensors and the results of the operations.

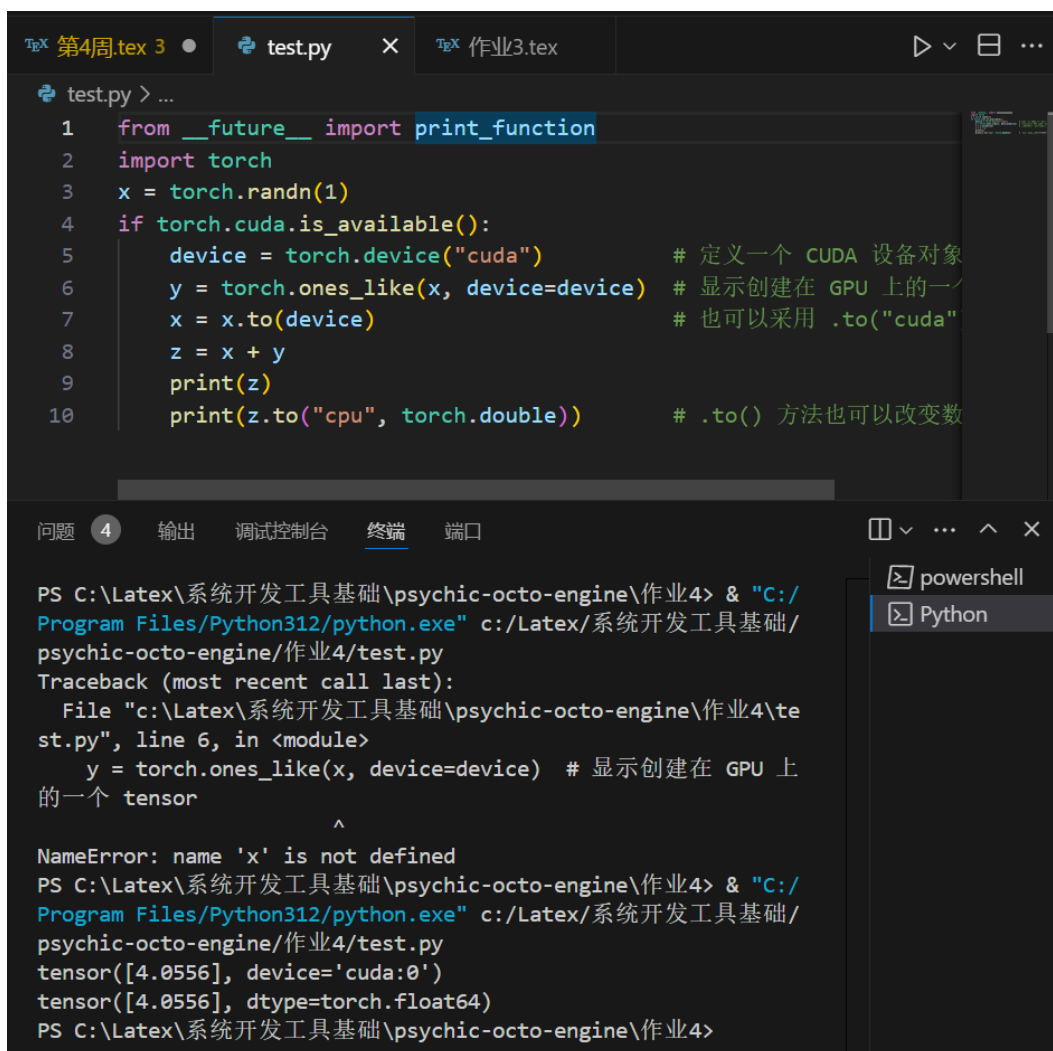
```
test.py > ...
1  from __future__ import print_function
2  import torch
3
4  a = torch.ones(5)
5  print(a)
6  b = a.numpy()
7  print(b)
8
9  a.add_(1)
10 print(a)
11 print(b)
12
13 import numpy as np
14 a = np.ones(5)
15 b = torch.from_numpy(a)
16 np.add(a, 1, out=a)
17 print(a)
18 print(b)
```

Terminal Output:

```
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([1., 1., 1., 1., 1.])
[1. 1. 1. 1. 1.]
tensor([2., 2., 2., 2., 2.])
[2. 2. 2. 2. 2.]
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([1., 1., 1., 1., 1.])
[1. 1. 1. 1. 1.]
tensor([2., 2., 2., 2., 2.])
[2. 2. 2. 2. 2.]
[2. 2. 2. 2. 2.]
tensor([2., 2., 2., 2., 2.], dtype=torch.float64)
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4>
```

图 7: 效果如图

5. CUDA 张量



The image shows a code editor window with a file named `test.py` open. The code in the editor is as follows:

```
1 from __future__ import print_function
2 import torch
3 x = torch.randn(1)
4 if torch.cuda.is_available():
5     device = torch.device("cuda")           # 定义一个 CUDA 设备对象
6     y = torch.ones_like(x, device=device)    # 显示创建在 GPU 上的一个
7     x = x.to(device)                        # 也可以采用 .to("cuda")
8     z = x + y
9     print(z)
10    print(z.to("cpu", torch.double))        # .to() 方法也可以改变数
```

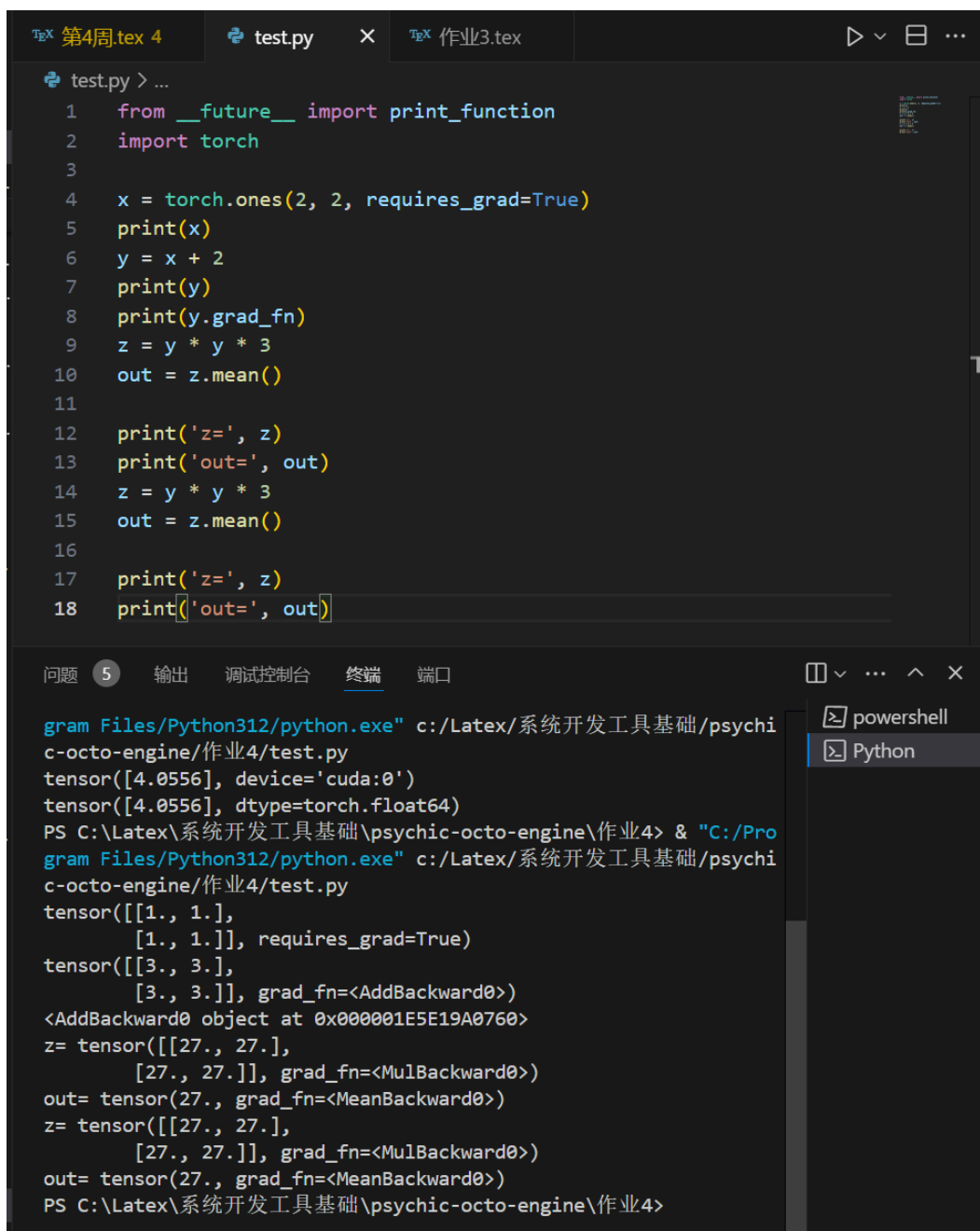
Below the code editor is a terminal window. The terminal shows the command to run the script and the resulting error message:

```
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
Traceback (most recent call last):
  File "c:\Latex\系统开发工具基础\psychic-octo-engine\作业4\test.py", line 6, in <module>
    y = torch.ones_like(x, device=device) # 显示创建在 GPU 上的一个 tensor
        ^
NameError: name 'x' is not defined

PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([4.0556], device='cuda:0')
tensor([4.0556], dtype=torch.float64)
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4>
```

图 8: 效果如图

6. autograd 张量



The image shows a code editor with a dark theme. The top bar has tabs for '第4周.tex 4', 'test.py', and '作业3.tex'. The main editor area contains a Python script named 'test.py' with the following code:

```
1 from __future__ import print_function
2 import torch
3
4 x = torch.ones(2, 2, requires_grad=True)
5 print(x)
6 y = x + 2
7 print(y)
8 print(y.grad_fn)
9 z = y * y * 3
10 out = z.mean()
11
12 print('z=', z)
13 print('out=', out)
14 z = y * y * 3
15 out = z.mean()
16
17 print('z=', z)
18 print('out=', out)
```

Below the editor is a terminal window with tabs for '问题 5', '输出', '调试控制台', '终端', and '端口'. The '终端' tab is active, showing the command prompt and the output of the script:

```
gram Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([4.0556], device='cuda:0')
tensor([4.0556], dtype=torch.float64)
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([[1., 1.],
        [1., 1.]], requires_grad=True)
tensor([[3., 3.],
        [3., 3.]], grad_fn=<AddBackward0>)
<AddBackward0 object at 0x000001E5E19A0760>
z= tensor([[27., 27.],
          [27., 27.]], grad_fn=<MulBackward0>)
out= tensor(27., grad_fn=<MeanBackward0>)
z= tensor([[27., 27.],
          [27., 27.]], grad_fn=<MulBackward0>)
out= tensor(27., grad_fn=<MeanBackward0>)
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4>
```

图 9: 效果如图

7. 梯度


```

第4周.tex 4 • test.py × 作业3.tex
test.py > ...
19 out.backward()
20 # 输出梯度 d(out)/dx
21 print(x.grad)
22 x = torch.randn(3, requires_grad=True)
23
24 y = x * 2
25 while y.data.norm() < 1000:
26     y = y * 2
27
28 print(y)
29 v = torch.tensor([0.1, 1.0, 0.0001], dtype=torch.float)
30 y.backward(v)
31
32 print(x.grad)
33 print(x.requires_grad)
34 print((x ** 2).requires_grad)
35
36 with torch.no_grad():
37     print((x ** 2).requires_grad)

问题 5 输出 调试控制台 终端 端口
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/test.py
tensor([[1., 1.],
        [1., 1.]], requires_grad=True)
tensor([[3., 3.],
        [3., 3.]], grad_fn=<AddBackward0>)
<AddBackward0 object at 0x0000021E9A9057B0>
z= tensor([[27., 27.],
          [27., 27.]], grad_fn=<MulBackward0>)
out= tensor(27., grad_fn=<MeanBackward0>)
z= tensor([[27., 27.],
          [27., 27.]], grad_fn=<MulBackward0>)
out= tensor(27., grad_fn=<MeanBackward0>)
tensor([[4.5000, 4.5000],
        [4.5000, 4.5000]])
tensor([ 704.7581, -209.6495, 1831.2552], grad_fn=<MulBackward0>)
tensor([1.0240e+02, 1.0240e+03, 1.0240e-01])
True
True
False
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4>

```

图 10: 效果如图

2 课后习题

2.1 调试及性能分析

1. 使用 Linux 上的 `journalctl` 或 macOS 上的 `log show` 命令来获取最近一天中超级用户的登录信息及其所执行的指令。

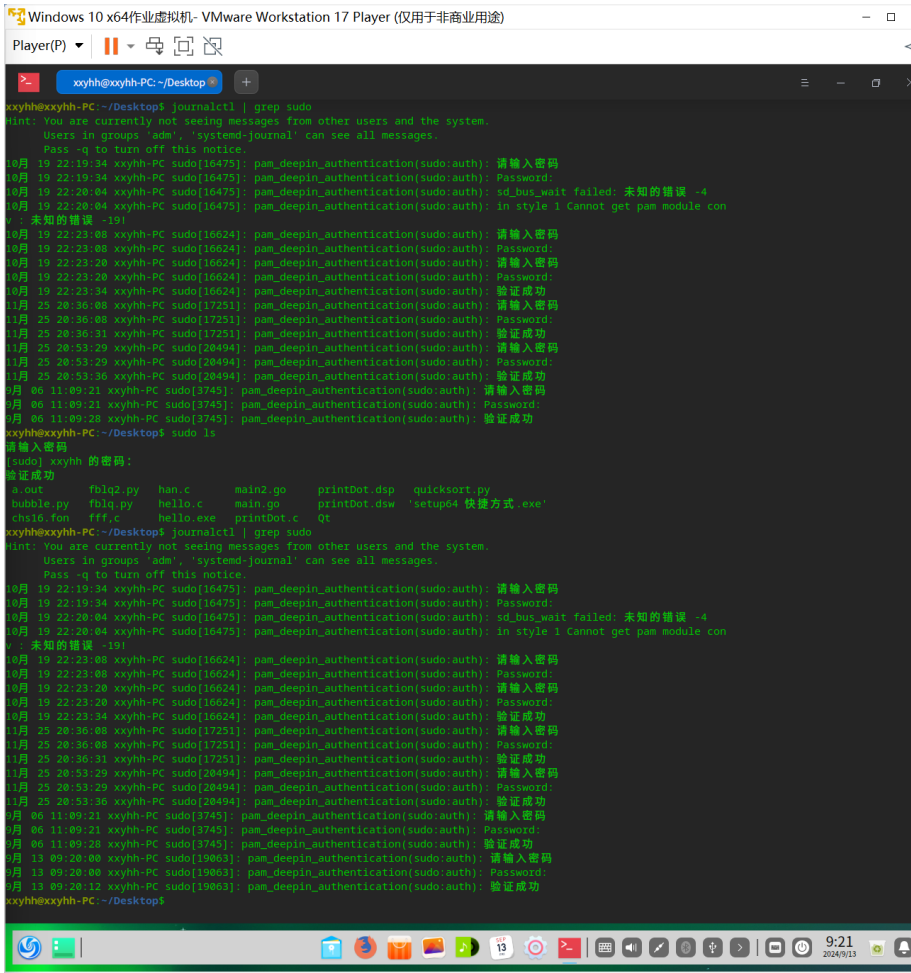
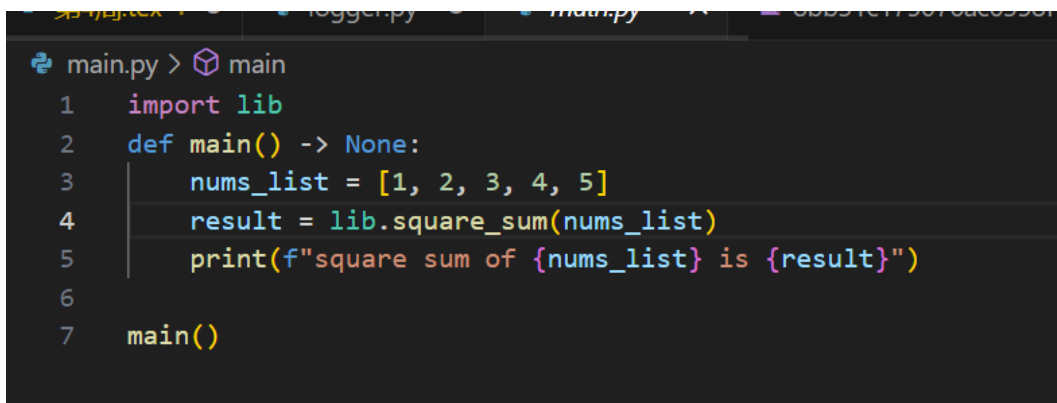


图 11: 在终端中打印多种颜色

2. pdb学习 1.首先我们编写了一个函数，用于计算算术平方和，在main函数中调用函数计算1-5的平方和：

```
lib.py > square_sum
1 from typing import Sequence
2
3 def square_sum(nums: Sequence[int]) -> int:
4     result = 0
5     for i in range(len(nums)):
6         n = nums[i]
7         result += n*n
8
9     return result
```

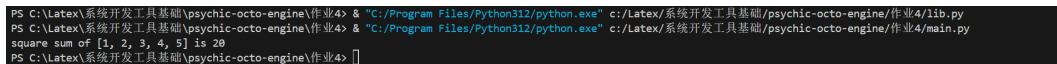
图 12: lib.py



```
main.py > main
1  import lib
2  def main() -> None:
3      nums_list = [1, 2, 3, 4, 5]
4      result = lib.square_sum(nums_list)
5      print(f"square sum of {nums_list} is {result}")
6
7  main()
```

图 13: main.py

2.运行发现答案不符合预期



```
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/lib.py
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> & "C:/Program Files/Python312/python.exe" c:/Latex/系统开发工具基础/psychic-octo-engine/作业4/main.py
square sum of [1, 2, 3, 4, 5] is 20
PS C:\Latex\系统开发工具基础\psychic-octo-engine\作业4> █
```

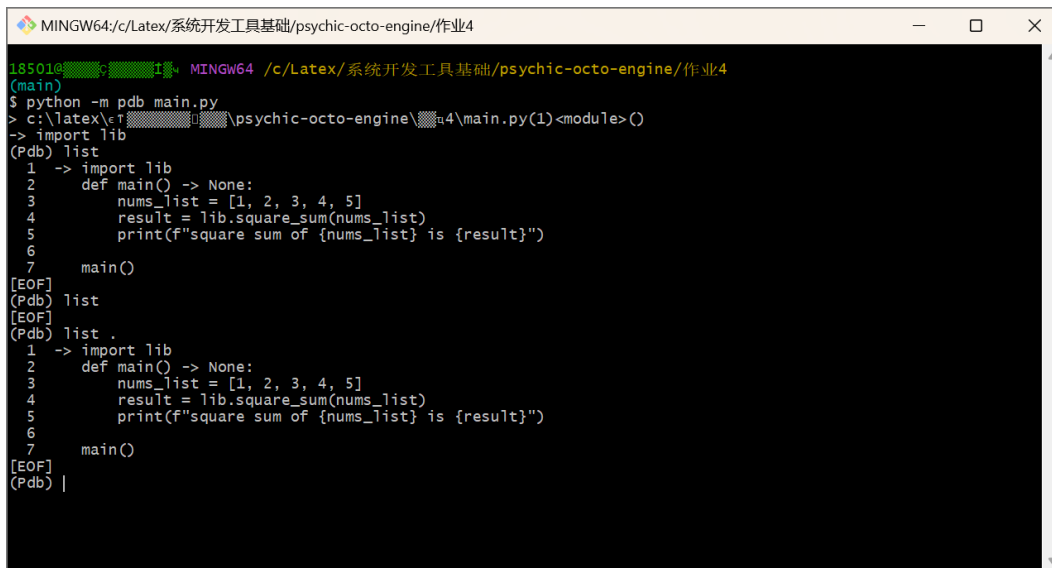
图 14: 运行结果

3.这时我们打开终端利用pdb调试代码



```
MINGW64/c:/Latex/系统开发工具基础/psychic-octo-engine/作业4
18501@C:\Latex\系统开发工具基础\psychic-octo-engine\作业4 MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4
(main)
$ python -m pdb main.py
> c:\latex\epsilon\psychic-octo-engine\作业4\main.py(1)<module>()
-> import lib
(Pdb)
```

图 15: 进入pdb

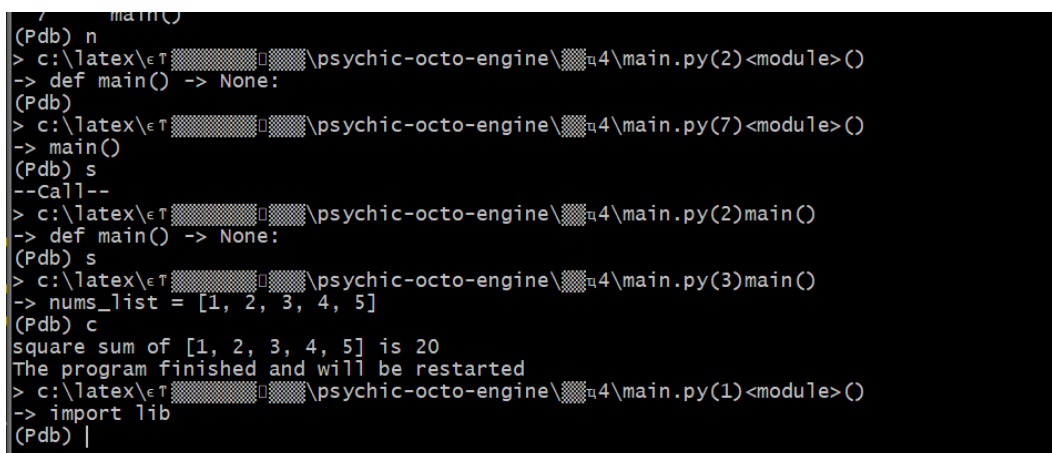


```

MINGW64/c/Latex/系统开发工具基础/psychic-octo-engine/作业4
(main)
$ python -m pdb main.py
> c:\latex\心理\psychic-octo-engine\4\main.py(1)<module>()
-> import lib
(Pdb) list
1  -> import lib
2      def main() -> None:
3          nums_list = [1, 2, 3, 4, 5]
4          result = lib.square_sum(nums_list)
5          print(f"square sum of {nums_list} is {result}")
6
7      main()
[EOF]
(Pdb) list
[EOF]
(Pdb) list .
1  -> import lib
2      def main() -> None:
3          nums_list = [1, 2, 3, 4, 5]
4          result = lib.square_sum(nums_list)
5          print(f"square sum of {nums_list} is {result}")
6
7      main()
[EOF]
(Pdb) |

```

图 16: list查看当前代码



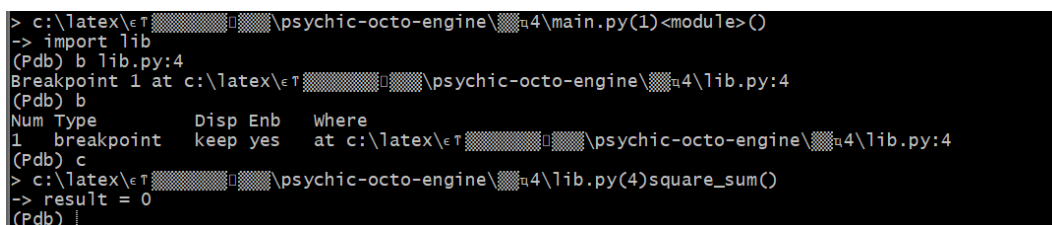
```

7      main()
(Pdb) n
> c:\latex\心理\psychic-octo-engine\4\main.py(2)<module>()
-> def main() -> None:
(Pdb)
> c:\latex\心理\psychic-octo-engine\4\main.py(7)<module>()
-> main()
(Pdb) s
--Call--
> c:\latex\心理\psychic-octo-engine\4\main.py(2)main()
-> def main() -> None:
(Pdb) s
> c:\latex\心理\psychic-octo-engine\4\main.py(3)main()
-> nums_list = [1, 2, 3, 4, 5]
(Pdb) c
square sum of [1, 2, 3, 4, 5] is 20
The program finished and will be restarted
> c:\latex\心理\psychic-octo-engine\4\main.py(1)<module>()
-> import lib
(Pdb) |

```

图 17: n,s顺序查看代码, c结束代码

4.我们先检查被调用的函数有没有出现问题, 需要在函数内部设置断点:



```

> c:\latex\心理\psychic-octo-engine\4\main.py(1)<module>()
-> import lib
(Pdb) b lib.py:4
Breakpoint 1 at c:\latex\心理\psychic-octo-engine\4\lib.py:4
(Pdb) b
Num Type      Disp Enb   Where
1 breakpoint keep yes    at c:\latex\心理\psychic-octo-engine\4\lib.py:4
(Pdb) c
> c:\latex\心理\psychic-octo-engine\4\lib.py(4)square_sum()
-> result = 0
(Pdb) |

```

图 18: 设置断点并进入到函数内部

```
(Pdb) s
> c:\latex\epsilon\psychic-octo-engine\q4\lib.py(5)square_sum()
-> for i in range(len(nums)):
(Pdb) s
> c:\latex\epsilon\psychic-octo-engine\q4\lib.py(6)square_sum()
-> n = nums[1]
(Pdb) s
> c:\latex\epsilon\psychic-octo-engine\q4\lib.py(7)square_sum()
-> result += n*n
(Pdb) p [i, n, nums]
[0, 2, [1, 2, 3, 4, 5]]
(Pdb) p [n, nums[i]]
[2, 1]
(Pdb) |
```

图 19: 检查每个变量的值是否有问题

```
(Pdb) q
18501@c:\I\ MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$
```

图 20: 发现代码问题，修正并退出pdb

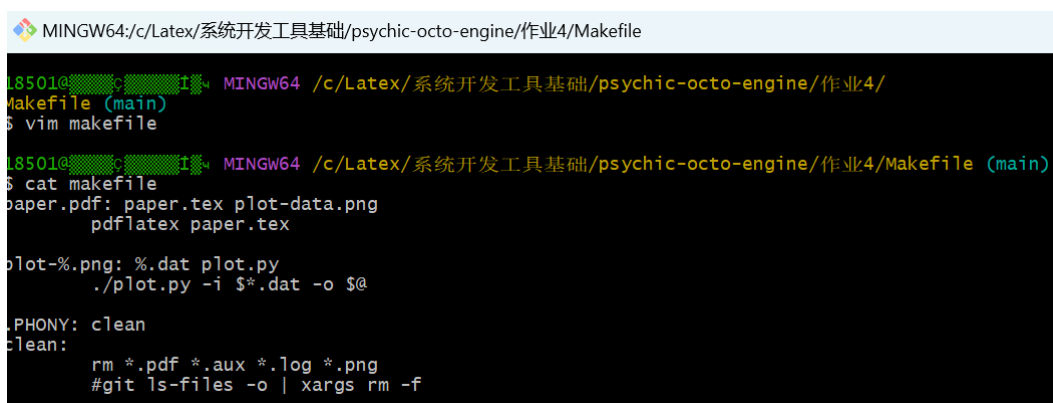
```
18501@c:\I\ MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ python main.py
square sum of [1, 2, 3, 4, 5] is 55
18501@c:\I\ MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4 (main)
$ |
```

图 21: 再运行代码，结果正确

2.2 元编程演示实验

```
MINGW64:/c/Latex/系统开发工具基础/psychic-octo-engine/作业4/Makefile
9 paper.pdf: paper.tex plot-data.png
8     pdflatex paper.tex
7
6 plot-%.png: %.dat plot.py
5     ./plot.py -i $.dat -o $@
4
3 .PHONY: clean
2 clean:
1     rm *.pdf *.aux *.log *.png
10    #git ls-files -o | xargs rm -f
~
~
~
```

图 22: 编写代码



```
MINGW64:/c/Latex/系统开发工具基础/psychic-octo-engine/作业4/Makefile
18501@PSYCHIC-OCTO-ENGINE:~$ MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4/
Makefile (main)
$ vim makefile

18501@PSYCHIC-OCTO-ENGINE:~$ MINGW64 /c/Latex/系统开发工具基础/psychic-octo-engine/作业4/Makefile (main)
$ cat makefile
paper.pdf: paper.tex plot-data.png
    pdflatex paper.tex

plot-%.png: %.dat plot.py
    ./plot.py -i $*.dat -o $@

.PHONY: clean
clean:
    rm *.pdf *.aux *.log *.png
    #git ls-files -o | xargs rm -f
```

图 23: cat展示

3 实验心得

3.1 调试及性能分析

我深刻认识到在编程中及时发现和解决问题的重要性。通过各种调试工具和技巧，我们能够精准地定位代码中的错误和瓶颈，这不仅提高了代码的质量，更确保了程序的稳定运行。性能分析则让我学会了如何优化代码，使其在资源利用和运行效率上达到更优的状态，这种对细节的关注和不断改进的追求让我在编程能力上有了显著的提升。

3.2 元编程

元编程展现出了编程领域极高的灵活性和创造性。它就像是一把神奇的钥匙，打开了通往更高级编程境界的大门。在这个过程中，我们见识到了如何通过精妙的代码设计和巧妙的编程技巧，实现一些看似不可能的功能和行为。例如，利用元编程可以动态地生成代码结构，根据特定的条件和需求在运行时构建合适的代码片段，这极大地拓展了程序的适应性和智能性。它让我们能够深入探究编程的底层机制，不再仅仅满足于表面的代码编写，而是真正理解代码背后的运作原理。这种对编程本质的更深入理解，激发了我探索更多编程可能性的强烈热情。我开始思考如何运用元编程的理念和方法，去创造更独特、高效的解决方案，去突破传统编程思维的局限，去挖掘那些隐藏在代码世界深处的宝藏。

3.3 PyTorch 编程

它强大的功能和简洁的接口使得构建深度学习模型变得相对容易和直观。在使用 PyTorch 的过程中，我逐渐掌握了如何搭建复杂的神经网络结构、进行高效的训练和优化。它让我切实感受到了深度学习的魅力和潜力，也让我对未来在这一领域的深入探索充满了期待。

总之，这一系列的学习经历让我受益匪浅。它们不仅丰富了我的知识体系，更锻炼了我的思维能力和实践能力。我深知这些宝贵的经验将在我未来的编程道路上发挥重要的作用，激励我不断追求卓越，探索更多未知的领域。

4 Github仓库ssh链接

```
git@github.com:xixiyhaha/psychic-octo-engine.git
```