1. anaconda 安装 python 环境

（1） 下载 anaconda

官方网站 https://www.anaconda.com/download/success

速度非常慢，建议去镜像网站上下载。Index of /anaconda/archive/ | 清华大学开源软件镜像站 | Tsinghua Open Source Mirror

二者对应版本，本项目安装 python 3.8 以上版本即可。示例下载 2022.10-Win 版本，如图所示，其基础环境（base 环境）下的 Python 为 3.9 版本。

Anaconda3-2022.10-Linux-aarch64.sh	534.5 MiB	2022-10-18 05:24
Anaconda3-2022.10-Linux-ppc64le.sh	360.0 MiB	2022-10-18 05:24
Anaconda3-2022.10-Linux-s390x.sh	282.4 MiB	2022-10-18 05:24
Anaconda3-2022.10-Linux-x86_64.sh	737.6 MiB	2022-10-18 05:24
Anaconda3-2022.10-MacOSX-arm64.pkg	484.1 MiB	2022-10-18 05:24
Anaconda3-2022.10-MacOSX-arm64.sh	472.5 MiB	2022-10-18 05:25
Anaconda3-2022.10-MacOSX-x86_64.pkg	688.6 MiB	2022-10-18 05:25
Anaconda3-2022.10-MacOSX-x86_64.sh	681.6 MiB	2022-10-18 05:25
Anaconda3-2022.10-Windows-x86_64.exe	621.2 MiB	2022-10-18 05:26

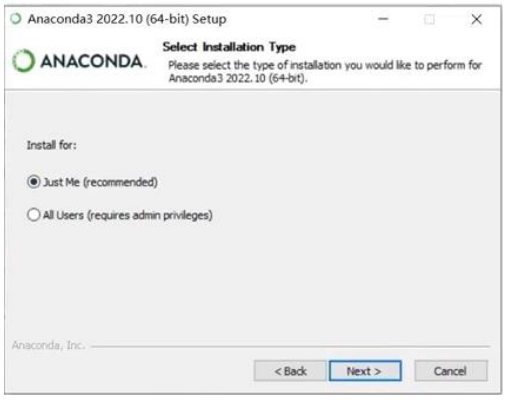
考虑到后面会用虚拟环境，创建虚拟环境时可以设置此环境中的 Python 释器版本，所以这里下载哪一版 Anaconda 并不重要。

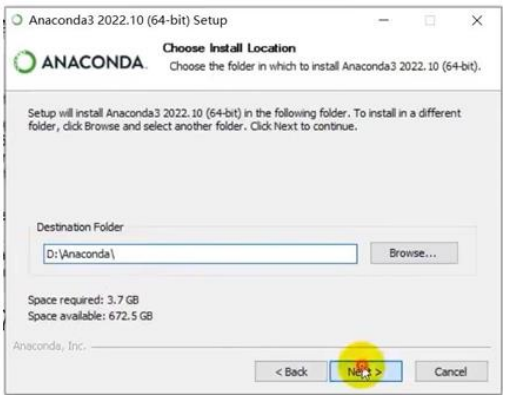
（2） 安装 anaconda

双击刚刚下载的 exe 文件，会有三个分岔口，分别按下列规则选择。

① Just me 和 All Users，选择 Just me；

② 安装路径选择最大的盘（一般是 D 盘），放在新建的【D:\Anaconda】里；

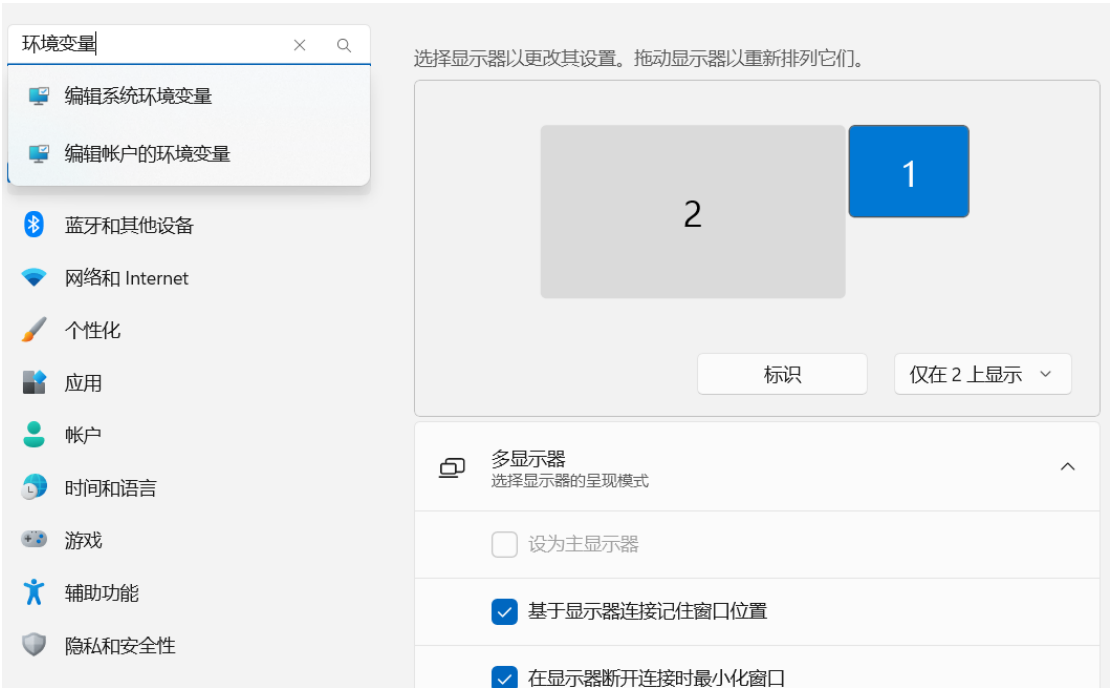




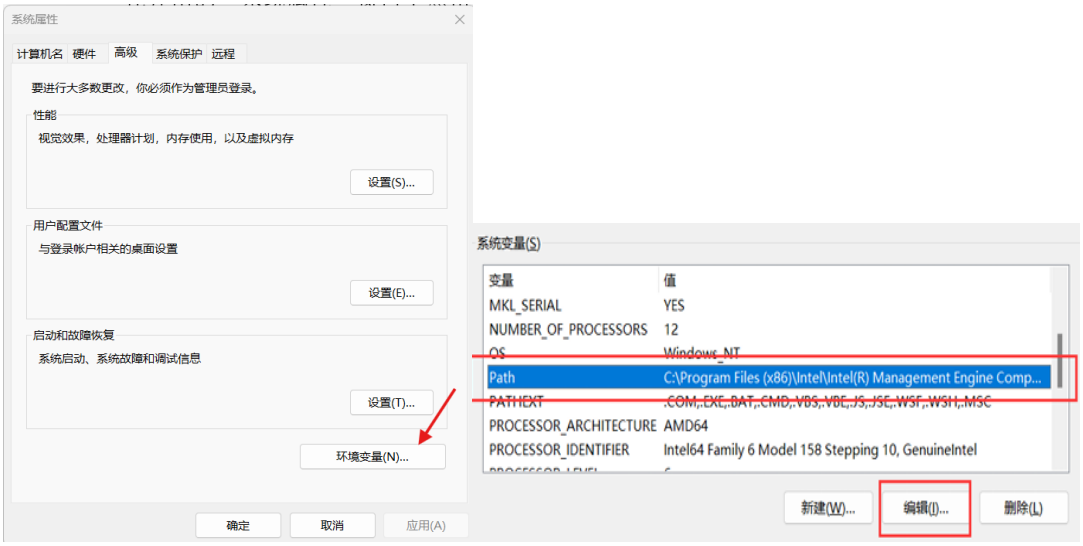
（3） 配置环境变量

桌面按下鼠标反键，点击“显示设置”。

左上角“查找设置”中输入“环境变量”，点击“编辑系统环境变量”。

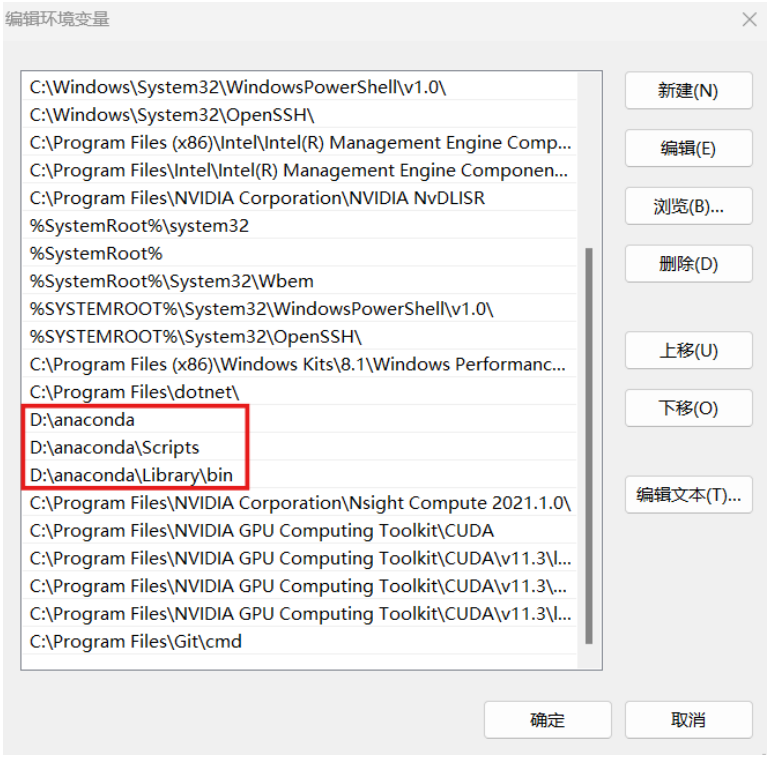


在弹出的“系统属性”窗口中点击“环境变量”，再在弹出的“环境变量”窗口中选中 Path 路径，并点击编辑。



通过右侧的“新建”按钮，可新建环境变量的路径，将【D:\Anaconda】、

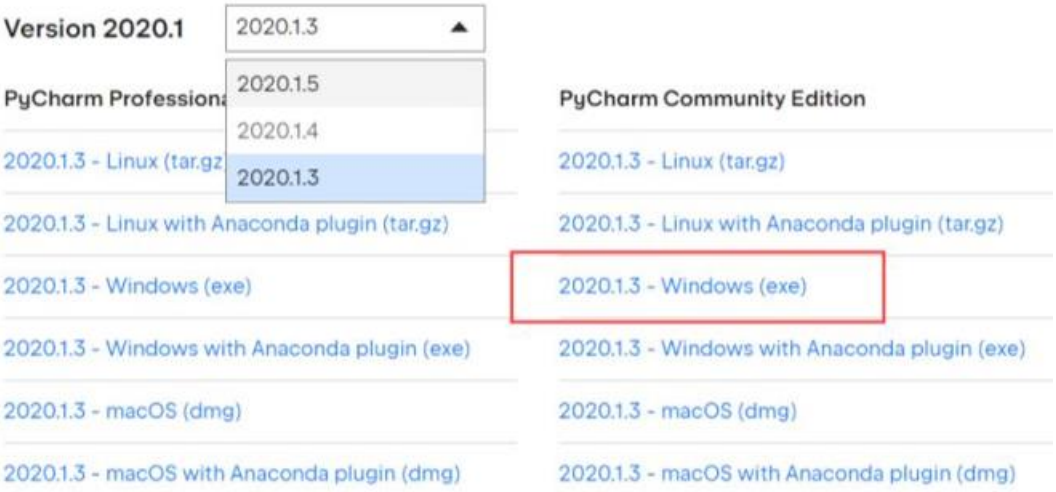
【D:\Anaconda\Scripts】与【D:\Anaconda\Library\bin】添加到环境变量。 若 Anaconda 安装路径不是 D:\Anaconda，而是 E:\Anaconda，以上三个 环境变量需要对应地进行更改，即改为【E:\Anaconda】、【 E:\Anaconda\Scripts】 与【E:\Anaconda\Library\bin】。



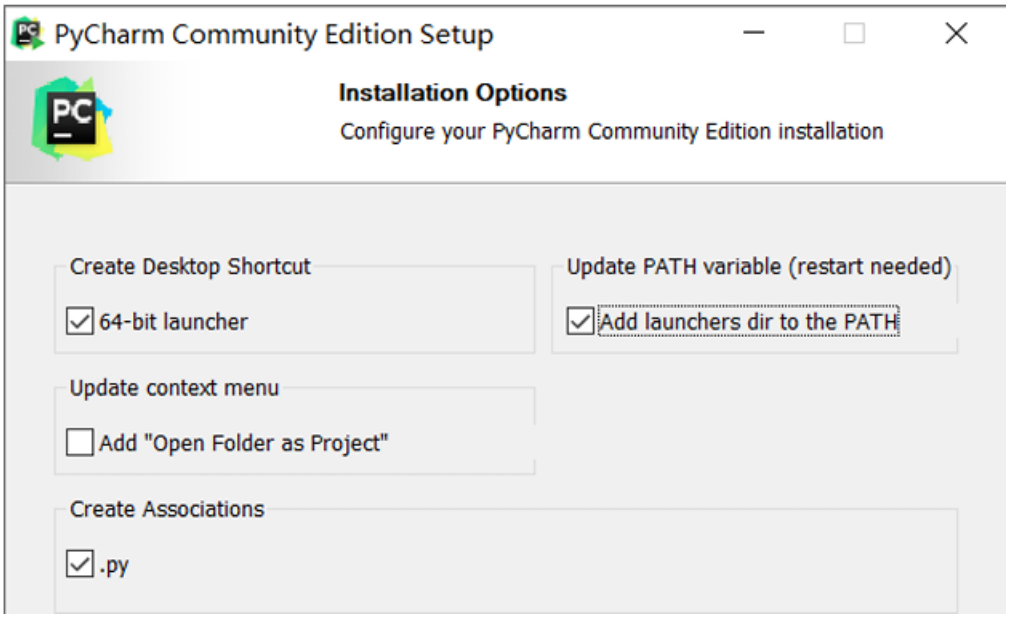
2. 安装 pycharm

（1）首先，去 jetbrains 公司的官网下载 PyCharm，地址为 https://www.jetbrains.com/pycharm/download/other.html

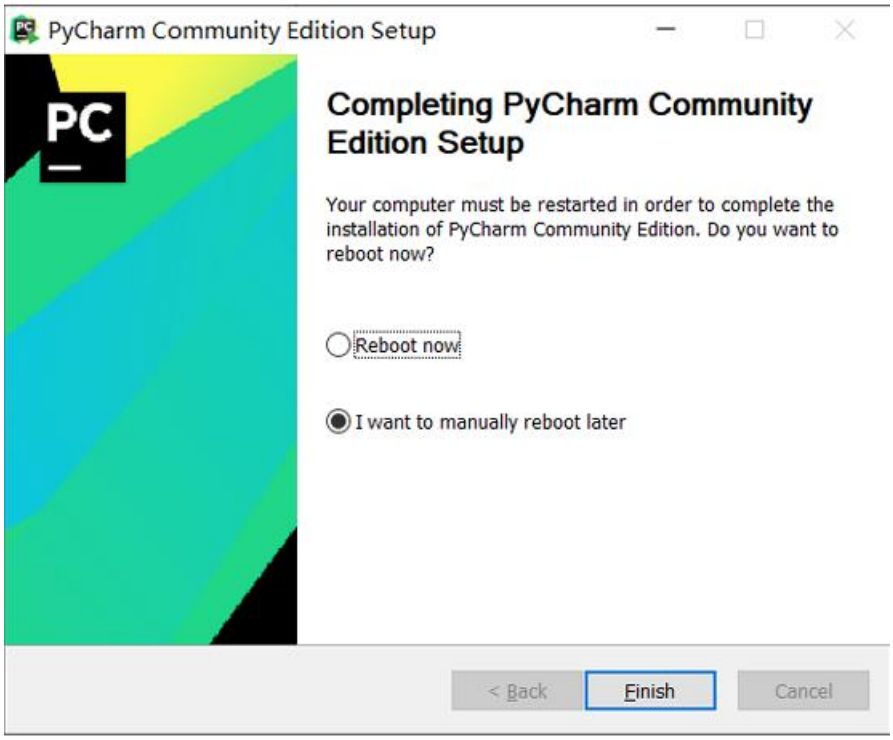
示例下载社区版（足够个人使用）的 2020.1.3-win 版本。



安装时，请放在 D 盘的新建文件夹：D:\PyCharm 里。选好安装地址后，请 勾选如图：

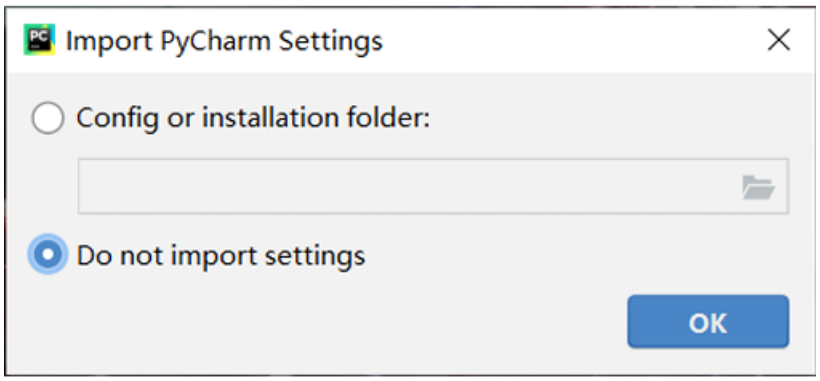


接着下一个窗口选择默认的 jetbrains 即可；最后一个窗口问你要不要重启，不重启好像也没啥事，如图

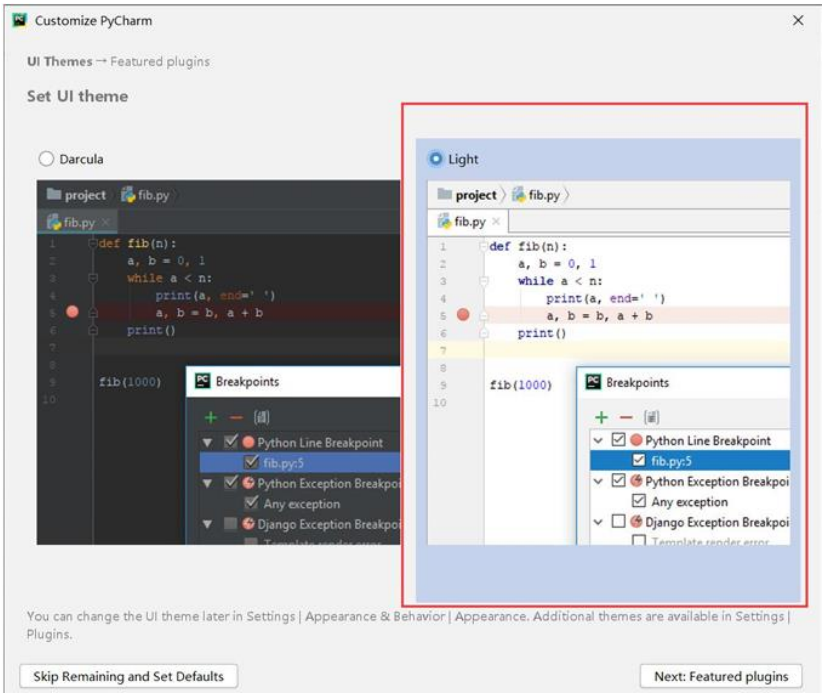


（2）设置 pycharm

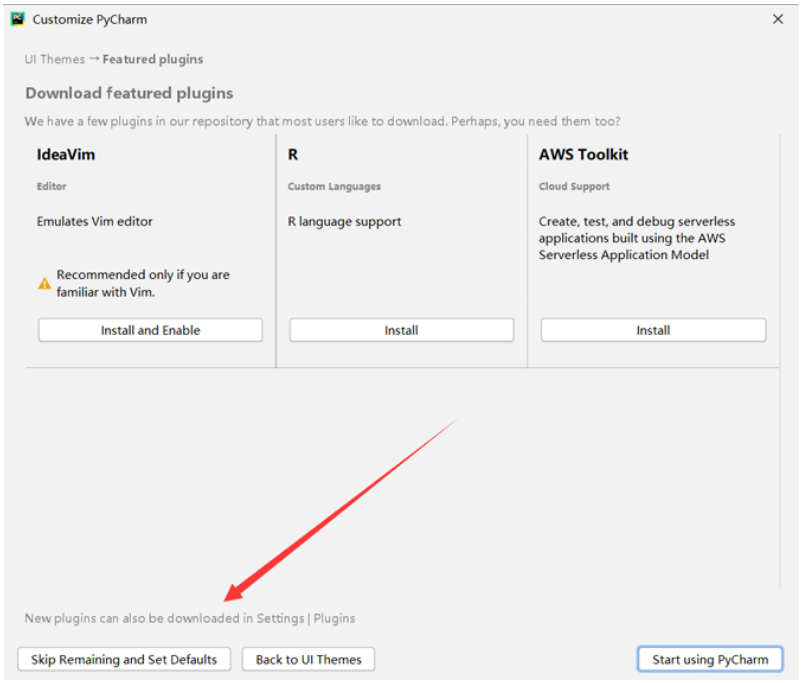
第一个岔路，选择第二个。



第二个岔路，选择主题，默认为黑色。



最后一个配置，点左下角的按钮跳过即可。


Customize PyCharm
UI Themes → Featured plugins
Download featured plugins
We have a few plugins in our repository that most users like to download. Perhaps, you need them too?
IdeaVim
Editor
Emulates Vim editor
Recommended only if you are familiar with Vim.
Install and Enable
R
Custom Languages
R language support
Install
AWS Toolkit
Cloud Support
Create, test, and debug serverless applications built using the AWS Serverless Application Model
Install
New plugins can also be downloaded in Settings | Plugins
Skip Remaining and Set Defaults
Back to UI Themes
Start using PyCharm

简单练习

1. 计算斐波那契数列（递归与迭代）

```
# 递归方式
def fibonacci_recursive(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

# 迭代方式
def fibonacci_iterative(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a

print(fibonacci_recursive(10))
print(fibonacci_iterative(10))
```

---

2. 判断一个数是否为素数

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

print(is_prime(29))  # True
print(is_prime(30))  # False
```

---

3. 反转字符串

```
def reverse_string(s):
    return s[::-1]
```

```
print(reverse_string("hello"))  # "olleh"
```

---

4. 统计字符串中每个字符的出现次数

```
from collections import Counter

def char_count(s):
    return dict(Counter(s))

print(char_count("hello world"))
```

---

5. 计算列表中最大值和最小值

```
def find_max_min(lst):
    return max(lst), min(lst)

numbers = [3, 1, 7, 9, 2, 8]
print(find_max_min(numbers))  # (9, 1)
```

---

6. 判断字符串是否为回文

```
def is_palindrome(s):
    return s == s[::-1]

print(is_palindrome("radar"))  # True
print(is_palindrome("hello"))  # False
```

---

7. 计算列表的平均值

```
def average(lst):
    return sum(lst) / len(lst)

numbers = [10, 20, 30, 40, 50]
print(average(numbers))  # 30.0
```

---

8. 统计列表中的奇偶数

```
def count_odd_even(lst):
    odd_count = sum(1 for x in lst if x % 2 == 1)
    even_count = len(lst) - odd_count
    return {"Odd": odd_count, "Even": even_count}

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(count_odd_even(numbers))  # {'Odd': 5, 'Even': 4}
```

---

9. 用字典存储学生成绩并计算平均分

```
students = {
    "Alice": 85,
    "Bob": 92,
    "Charlie": 78,
    "David": 90
}

average_score = sum(students.values()) / len(students)
print(f"平均成绩: {average_score}")
```

---

10. 用 Lambda 表达式计算平方

```
square = lambda x: x ** 2
print(square(5))  # 25
```

进阶练习

1. Python 函数：默认参数 & 可变参数

```
def greet(name="Guest", age=18):
    print(f"Hello, {name}! You are {age} years old.")

greet("Alice", 25)
greet()  # 使用默认值

# *args 和 **kwargs 示例
def show_info(*args, **kwargs):
    print("位置参数:", args)
    print("关键字参数:", kwargs)

show_info(1, 2, 3, name="Alice", age=25)
```

---

2. 递归函数：计算阶乘

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # 120
```

---

3. 生成器函数：斐波那契数列

```
def fibonacci_generator(n):
    a, b = 0, 1
    for _ in range(n):
        yield a
        a, b = b, a + b

for num in fibonacci_generator(10):
    print(num, end=" ")  # 0 1 1 2 3 5 8 13 21 34
```

---

4. Python 类与对象

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def introduce(self):
        return f"My name is {self.name} and I am {self.age} years
old."

p1 = Person("Alice", 25)
print(p1.introduce())  # My name is Alice and I am 25 years old.
```

---

5. 继承与方法重写

```
class Animal:
    def speak(self):
        return "Some sound"

class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"

dog = Dog()
cat = Cat()
print(dog.speak())  # Woof!
print(cat.speak())  # Meow!
```

---

6. 静态方法和类方法

```
class MathUtils:
    @staticmethod
    def add(a, b):
        return a + b
```

```
    @classmethod
    def multiply(cls, a, b):
        return a * b

print(MathUtils.add(5, 3))  # 8
print(MathUtils.multiply(5, 3))  # 15
```

---

7. 装饰器：计算函数执行时间

```
import time

def timer(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"函数 {func.__name__} 执行时间: {end_time -
start_time:.6f} 秒")
        return result
    return wrapper

@timer
def slow_function():
    time.sleep(2)
    return "Finished"

print(slow_function())  # 计算运行时间
```

---

8. 上下文管理器（with 语句）

```
class FileManager:
    def __init__(self, filename, mode):
        self.filename = filename
        self.mode = mode

    def __enter__(self):
        self.file = open(self.filename, self.mode, encoding="utf-8")
        return self.file

    def __exit__(self, exc_type, exc_value, traceback):
        self.file.close()
```

```
with FileManager("test.txt", "w") as f:
    f.write("Hello, World!")
```

---

9. 多线程示例

```
import threading
import time

def print_numbers():
    for i in range(1, 6):
        time.sleep(1)
        print(i)

thread = threading.Thread(target=print_numbers)
thread.start()

print("主线程继续执行...")
thread.join()
print("子线程执行完毕")
```

---

10. 异常处理

```
def divide(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("错误: 不能除以零！")
        return None
    except TypeError:
        print("错误: 请输入数字！")
        return None
    else:
        print("计算成功")
        return result
    finally:
        print("计算结束")

print(divide(10, 2))  # 5.0
print(divide(10, 0))  # 错误: 不能除以零！
```

---

11. Lambda + map, filter, reduce

```
from functools import reduce

nums = [1, 2, 3, 4, 5]

# 使用 map 计算平方
squares = list(map(lambda x: x ** 2, nums))
print(squares)  # [1, 4, 9, 16, 25]

# 使用 filter 过滤偶数
evens = list(filter(lambda x: x % 2 == 0, nums))
print(evens)  # [2, 4]

# 使用 reduce 计算乘积
product = reduce(lambda x, y: x * y, nums)
print(product)  # 120
```

---

12. 计算矩阵转置

```
def transpose_matrix(matrix):
    return [list(row) for row in zip(*matrix)]

matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(transpose_matrix(matrix))
```
