作业 1

1. 编程实现任意两个正整数的最小公倍数。
2. 编程实现求解模素数 p 构成的有限域 GF（p）上的全部非零元素的加法和乘法逆元。

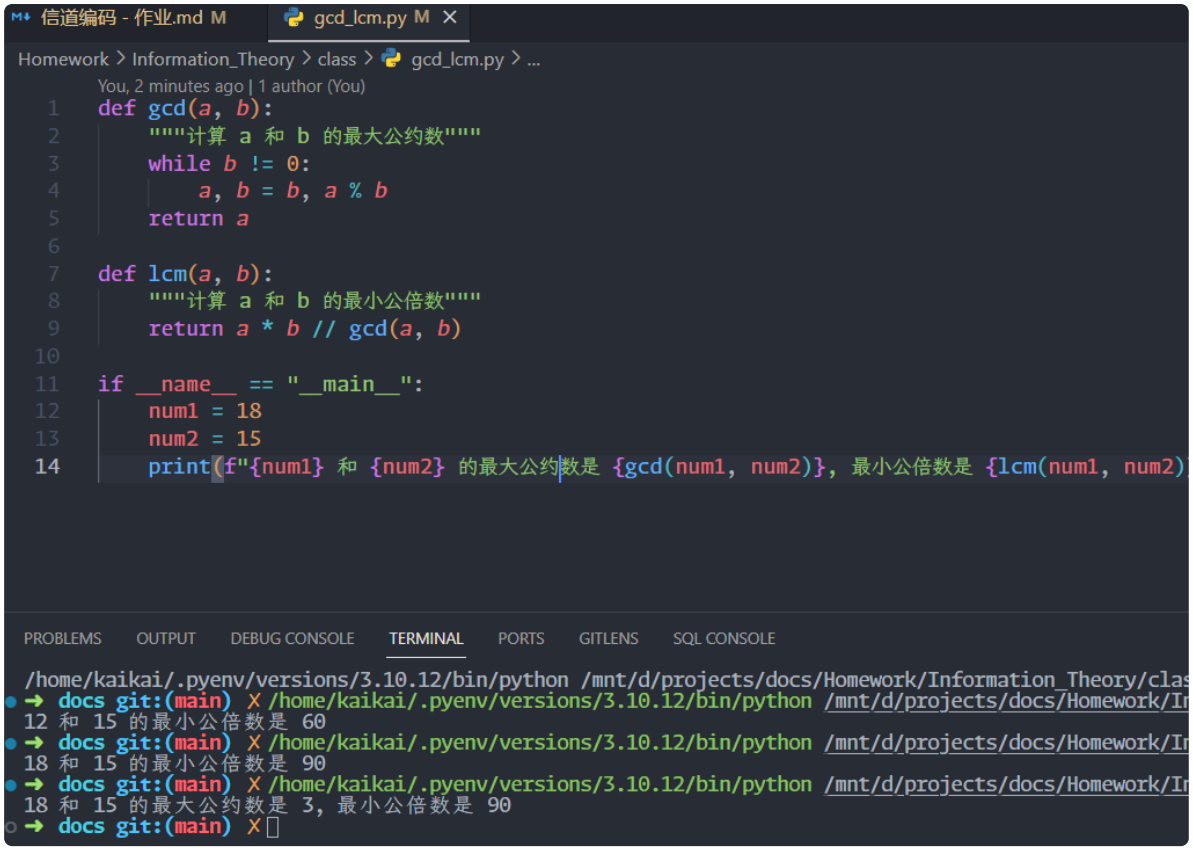
1. 求最大公约数与最小公倍数

```python
def gcd(a, b):
    """计算 a 和 b 的最大公约数"""
    while b != 0:
        a, b = b, a % b
    return a

def lcm(a, b):
    """计算 a 和 b 的最小公倍数"""
    return a * b // gcd(a, b)

if __name__ == "__main__":
    num1 = 18
    num2 = 15
    print(f"{num1} 和 {num2} 的最大公约数是 {gcd(num1, num2)}, 最小公倍数是 {lcm(num1, num2)}")
```

输出：



2. 求乘法逆元与加法逆元

Copy

```python
def is_prime(n):
    """检查 n 是否为素数"""
    if n ≤ 1:
        return False
    if n ≤ 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i ≤ n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True


def find_inverse_elements(p):
    # 首先检查 p 是否为素数
    if not is_prime(p):
        raise ValueError("输入的数必须是素数")

    # 初始化存储逆元的列表
    additive_inverse = [0] * p
    multiplicative_inverse = [0] * p
```

```
    # 计算加法逆元
    for i in range(p):
        additive_inverse[i] = (-i) % p

    # 计算乘法逆元
    for i in range(1, p):
        for j in range(1, p):
            if (i * j) % p == 1:
                multiplicative_inverse[i] = j
                break

    # 返回逆元
    return additive_inverse, multiplicative_inverse


if __name__ == "__main__":
    # 设置素数
    p = 8

    # 求解逆元
    additive_inverse, multiplicative_inverse =
find_inverse_elements(p)

    # 打印结果
    print(f"模 {p} 的有限域上的全部非零元素的加法逆元是:
{additive_inverse[1:]}")
    print(f"模 {p} 的有限域上的全部非零元素的乘法逆元是:
{multiplicative_inverse[1:]}")
```

输出：