Python 数据结构与算法分析（第一章 导论）

1. 基本概念

- **算法**：算法是具有有限步骤的过程，依照这个过程便能解决问题。因此，算法就是解决方案。
- **可计算**：若存在能够解决某个问题的算法，那么该问题便是可计算的。
- **编程**：是指通过编程语言将算法编码以使其能被计算机执行的过程。
- **抽象数据类型（ADT）**：从逻辑上描述了如何看待数据及其对应运算而无须考虑具体实现。
- **数据结构**：抽象数据类型的实现方式常被称为数据结构。如，First in First out (FIFO)为抽象数据类型，队列（queue）为数据结构。
- **类**：对于Python以及其它所有面向对象编程语言中， 类都是对数据的构成（状态）以及数据能做什么（行为）的描述。类 = 数据状态 + 数据行为。
- **对象**：在面向对象编程范式中，数据项被称作对象，一个对象就是类的一个实例。

2. 编程练习

1. 欧几里得辗转相除法寻找最大公因数：

```
def euclid(frag_a, frag_b):
    ## 确定分子分母
    if frag_a >= frag_b:
        num = frag_b
        den = frag_a
    else:
        num = frag_a
        den = frag_b
    ## 辗转相除法求解最大公因数
    if den % num == 0:
        return num
    else:
        while den % num != 0:
            temp = num
            num = den % num
            den = temp
        return num


print("{a}和{b}最大公因数为{num}。".format(a=1997, b=615, num=euclid(1997, 615)))
```

> 1997和615最大公因数为1。

2. 8位全加器：

```
def half_adder(add_a, add_b, carry):
    if add_a not in [0, 1] or add_b not in [0, 1] or carry not in [0, 1]:
        assert print("Error! Please check inputs, each bit must be 0 or 1.")
    if add_a + add_b == 2:
        if carry == 1:
            add_result = 1
        else:
            add_result = 0
        carry_result = 1
    elif add_a + add_b + carry == 2:
        add_result = 0
        carry_result = 1
    else:
        add_result = add_a + add_b + carry
        carry_result = 0
    return add_result, carry_result


def adder_8bit(add_a, add_b):
    if len(add_a) != len(add_b) != 8:
        assert print('Error! Please check the inputs, the digit mush be eight.')
    carry_temp = 0
    result = ''
    for add_a_temp, add_b_temp in zip(add_a[::-1], add_b[::-1]):
        add_temp_result, carry_temp = half_adder(int(add_a_temp), int(add_b_temp), carry_temp)
        result += str(add_temp_result)
    return result[::-1], carry_temp


print('The sum of {} and {} is {}, and the carry bit is {}.'.format('10010011', '00110100',
                                                                    adder_8bit('10010011', '00110100')[0],
                                                                    adder_8bit('10010011', '00110100')[1]))
```

> The sum of 10010011 and 00110100 is 11000111, and the carry bit is 0.

参考文献

- Python数据结构与算法分析（第2版）