练习 1.1.1：编译器相对于解释器的优点是什么？解释器相对于编译器的优点是什么？

答案不唯一，只要能够对优缺点有充分讨论即可。

编译器把源代码编译成目标代码，生成的程序执行时不再需要编译器。由编译器产生的目标程序通常比解释器解释执行快很多。但由编译器产生的机器代码依赖于体系结构，移植到其他平台需要重新编译源程序。
解释器逐个语句地执行源程序，执行过程清晰直观，错误诊断效果通常比编译器更好。解释器的交互性更好，也方便对程序进行动态配置。另外，依赖解释执行的程序可移植性一般更好，移植到其他平台时可直接执行或修改较少代码。

练习 1.1.2：在一个语言处理系统中，编译器产生汇编语言而不是机器语言的好处是什么？

答案不唯一。

优点：汇编指令是机器指令的助记符，更容易阅读和理解。因此编译器产生汇编语言便于输出与调试。

练习 1.1.3：对下图中的块结构的 C 代码，指出赋给 w、x、y 和 z 的值

```c
int w, x, y, z;
int i = 21; int j = 8;
{
    int j = 4;
    i = 7;
    w = i * j;
}
x = i + j;
{
    int i = j;
    y = i + j;
}
z = i - j;
```

```c
int w, x, y, z;
int i = 9; int j = 14;
{
    int i = 5;
    w = i + j;
}
x = j - i;
{
    int j = 3;
    i = 2;
    y = i + j;
}
z = i + j;
```

1) w=28, x=15, y=16, z=-1
2) w=19, x=5, y=5, z=16

练习 1.1.4：下面的 C 代码的打印结果是什么？

24
48
undef

练习 2.1.4：有人把程序设计语言分为编译型和解释型两类，例如 C 是编译型，Python 是解释型。这个分类是否合理？能否构建 C 语言的解释器，或者 Python 的静态编译器？谈谈你的看法？

想看看学生们的想法，答得出彩应该加分，不过不知道初学者怎么看这个题……

我的想法：
编译和解释是语言实现，和语言设计本身应该分离。
我觉得可以做 C 的解释器，Python 的静态编译器不太好做，可能要把很大一部分解释器功能链接进去。
现在还是有很多人用题干的说法，我觉得辨析一下对学生的理解（或许？）有好处。