# 7.3 给函数参数增加元信息¶

## 问题¶

你写好了一个函数，然后想为这个函数的参数增加一些额外的信息，这样的话其他使用者就能清楚的知道这个函数应该怎么使用。

## 解决方案¶

使用函数参数注解是一个很好的办法，它能提示程序员应该怎样正确使用这个函数。 例如，下面有一个被注解了的函数：

```
def add(x:int, y:int) -> int:
    return x + y
```

python解释器不会对这些注解添加任何的语义。它们不会被类型检查，运行时跟没有加注解之前的效果也没有任何差距。 然而，对于那些阅读源码的人来讲就很有帮助啦。第三方工具和框架可能会对这些注解添加语义。同时它们也会出现在文档中。

```
>>> help(add)
Help on function add in module __main__:
add(x: int, y: int) -> int
>>>
```

尽管你可以使用任意类型的对象给函数添加注解(例如数字，字符串，对象实例等等)，不过通常来讲使用类或者字符串会比较好点。

## 讨论¶

函数注解只存储在函数的 `__annotations__` 属性中。例如：

```
>>> add.__annotations__
{'y': <class 'int'>, 'return': <class 'int'>, 'x': <class 'int'>}
```

尽管注解的使用方法可能有很多种，但是它们的主要用途还是文档。 因为python并没有类型声明，通常来讲仅仅通过阅读源码很难知道应该传递什么样的参数给这个函数。 这时候使用注解就能给程序员更多的提示，让他们可以正确的使用函数。

参考9.20小节的一个更加高级的例子，演示了如何利用注解来实现多分派(比如重载函数)。