1. 引言

在学习面向对象的时候，我们知道在 python 中有一类特殊的方法，叫做魔法方法，这种方法的特点如下：

1. 方法定义的时候以两个下划线开头和两个下划线结尾：如`__init__`、`__str__`和`__repr__`
2. 这类方法一般不需要我们手动调用，在满足某个条件的时候会自动调用，这个满足的条件我们可以成为调用时机。

在Python 中有两个魔法方法都是用来描述对象信息的，`__str__`和`__repr__`，那为什么要定义两个这样的方法呢，其实是他们设计的目的是不一样的：

1. `__repr__`的目标是准确性，或者说，`__repr__`的结果是让解释器用的。
2. `__str__`的目标是可读性，或者说，`__str__`的结果是让人看的。

2. 分析

那下边，我们详细的来看一下，他们的用法：
在不重写`__str__`和`__repr__`的情况下，打印对象的输出结果不太友好，是对象的内存地址，即 id 的结果。

```
# 定义 Person 类
class Person(object):
    def __init__(self, name):
        self.name = name


p = Person("isaac")
```

以下为测试输出的结果：

```
>>> print(p)
<__main__.Person object at 0x10f29b940>
>>> p
<__main__.Person object at 0x10f29b940>
>>> p.__str__()
'<__main__.Person object at 0x10f29b940>'
>>> p.__repr__()
'<__main__.Person object at 0x10f29b940>'
```

这样的输出结果，并不是我们想要的结果，此时我们重写`__str__`和`__repr__`方法。

2.1 重写 `__str__` 方法

```
# 定义 Person 类
class Person(object):
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return "__str__ 方法 " + self.name


p = Person("isaac")
```

以下为测试结果：

```
>>> print(p)
__str__ 方法 isaac
>>> str(p)
'__str__ 方法 isaac'
>>> f"{p}"
'__str__ 方法 isaac'
>>> p.__str__()
'__str__ 方法 isaac'
>>> p
<demo.Person object at 0x10e0e3588>
```

此时我们发现在使用 print 打印对象、对象的格式化输出以及调用 str 方法，调用的都是`__str__`方法。但在交互环境下，直接输出 对象的时候，没有调用`__str__` 方法，输出的结果仍然是 id 的结果。

2.2 重写`__repr__`方法

```
# 定义 Person 类
class Person(object):
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return "__str__ 方法 " + self.name

    def __repr__(self):
        return "__repr__ 方法 " + self.name


p = Person("isaac")
```

此时，我们再来看输出的结果，

```
>>> p
__repr__ 方法isaac
>>> p.__repr__()
'__repr__ 方法isaac'
>>> print(p)
__str__ 方法 isaac
```

通过简单的对比，我们发现，在交互环境下，直接输出对象，调用的`__repr__`方法。
另外还需要注意的是，如果将对象放在容器中进行输出，调用的是`__repr__`方法。

```
>>> [p]
[__repr__ 方法 isaac]
>>> (p)
__repr__ 方法 isaac
>>> {"1":p}
{'1': __repr__ 方法 isaac}
>>> print([p])
[__repr__ 方法 isaac]
```

3. 总结

Python 中的`__str__`和`__repr__`方法都是用来显示的，即描述对象信息的。

1. `__str__`的目标是可读性，或者说，`__str__`的结果是让人看的。主要用来打印，即 print 操作，
2. `__repr__`的目标是准确性，或者说，`__repr__`的结果是让解释器用的。`__repr__`用于交互模式下提示回应，
3. 如果没有重写`__str__`方法，但重写了`__repr__`方法时，所有调用`__str__`的时机都会调用`__repr__`方法。