ch0-注意点

py中没有i++,++i这种自增语法

py中没有 variable= expression ? b : c的三目运算符,替代为:

- (1) variable = a if expression else b
- (2)variable = (expression and [b] or [c])[0]
- (3) variable = expression and b or c

py中没有switch-case语句
原因见知乎-Python中为什么没有switch语法结构，有什么代替方案吗？
替代方案,参考伯乐在线-为什么Python中没有Switch/Case语句?

```
def numbers_to_strings(argument):
    switcher = {
        0: "zero",
        1: "one",
        2: "two",
    }
    return switcher.get(argument, "nothing")
```

Python3.5中，open()不同模式如r、r+ 、w+、w、a、a+有何不同？

> r 只能读
> r+ 可读可写 不会创建不存在的文件 从顶部开始写 会覆盖之前此位置的内容
> w+ 可读可写 如果文件存在 则覆盖整个文件不存在则创建
> w 只能写 覆盖整个文件 不存在则创建
> a 只能写 从文件底部添加内容 不存在则创建
> a+ 可读可写 从文件顶部读取内容 从文件底部添加内容 不存在则创建

但是实验发现，这里的`r+ 可读可写 不会创建不存在的文件 从顶部开始写 会覆盖之前此位置的内容不完整`。

test.in的内容：

```
hello1
ok2
byebye3
```

执行以下：

```
with open("test.in", 'r+')as f:
    f.readline()
    f.write("addition")
```

文件的内容变成了：

```
hello1
ok2
```

```
byebye3
addition
```

而：

```
with open("test.in", 'r+')as f:
    f.write("addition")
```

文件内容则变成了：

```
additionk2
byebye3
```

因此，正确的表述应为：

r+**可读可写，不会创建不存在的文件。如果直接写文件，则从顶部开始写，覆盖之前此位置的内容，如果先读后写，则会在文件最后追加内容。**