

## 第二课： 算法思维逻辑思维初探

### I.逻辑运算符

and or not

A and B 是 True 当且仅当 A 和 B 都是 True

A or B 是 True 只要 A, B 中有一个是 True 即可

not A 是 True 当且仅当 A 是 False

```
>>> (1==1)and(2==2)
True
>>> (1==1)or(2==2)
True
>>> not (1!=1)
True
```

注意不是=而是==

### II.if 选择执行语句

基本格式：注意 语句 1 可以写很多行语句

```
if <condition 1>:
    <sentence 1>
    elif<condition 2>:
        <sentence 2>
    .....
else:
    <sentence n>
```

条件 1 成立执行语句 1，后面的就都不会执行了。条件 1 不成立，条件 2 成立执行语句 2. 都不成立执行语句 n

```
>>> x=10
>>> if x<0:
...     print(str(x)+'<0')
... elif x>9:
...     print(str(x)+'>9')
... elif x>8:
...     print(str(x)+'>8')
... else:
...     print('i do not know')
...
10>9
```

if 1==1:

print('True') 是==

值得注意的是：>8 永远不会被执行

III.循环语句。

(1).for 基础，形式如下：注意缩进，tab 键是一个缩进，遍历对象可以是元组字典列表

```
for <循环变量> in <遍历对象>:
```

```
    <语句 1>
```

```
else:
```

```
    <语句 2>
```

实例：

(2)for 与 range

Range([start,],stop[,step])

start 可选参数 默认值为 0 stop 终止数 step 步长

实例：

```
>>> for i in range(0,3,1):
...     print(i)
...
0
1
2
>>> for i in range(0,3,2):
...     print(i)
...
0
2
```

同学们注意一下：range(0,3,1)并没有打出 3 .

还有一些函数比如 break, continue 有兴趣的同学可以自己查询资料。还有一种循环是 while 循环。格式为：有兴趣的同学可以自己查资料。

while <条件>:

```
    <语句 1>
```

```
else:
```


```
    <语句 2>
```

VI.如何从 txt 文档中读取数据

file.open() file.read() file.write() file.close 记得在最后要 close

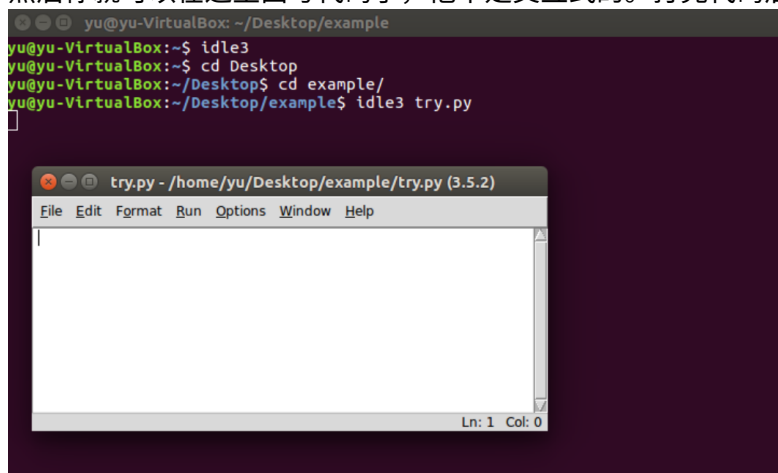
桌面上有一个 txt 文档，其中有两个五位数的自然数（如果不足五位用零不齐，如 12 在 txt 文档中写作 00012）。两个自然数中间用一个空格隔开。 其他的同学可以自行尝试想一想 1 和 5 代表的含义。同学们也可以尝试读一张图片。

```
yu@yu-VirtualBox:~/Desktop$ python3
Python 3.5.2 (default, Jul 5 2016, 12:43:10)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> new=open('example.txt')
>>> a=int(new.read(5))
>>> new.read(1)
' '
>>> b=int(new.read(5))
>>> a
12345
>>> b
992
>>>
```

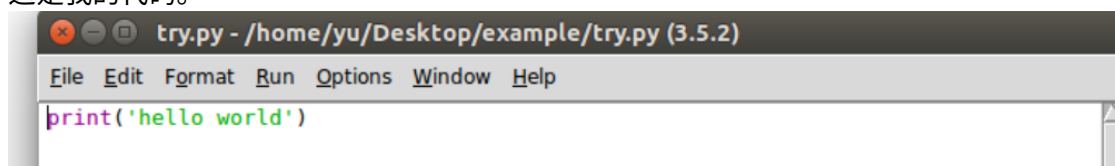


我们之前所有程序都是在 terminal 中完成的，现在我们在一个类似 txt 文档中写入自己的代码。

- (1)在桌面新建一个文件夹，New Folder。重命名为 example。
  - (2)打开文件夹，新建一个 document，重命名为 try.py
  - (3)打开命令行，使用 cd 和 cd filename 到达 example 文件夹。（桌面是 Desktop，注意大小写）
  - (4)输入 idle3 try.py
- (事实上，可以通过右键，打开方式 选择 idle3)
- 然后你就可以在这里面写代码了，他不是交互式的。打完代码后保存即可。)



这是我的代码。



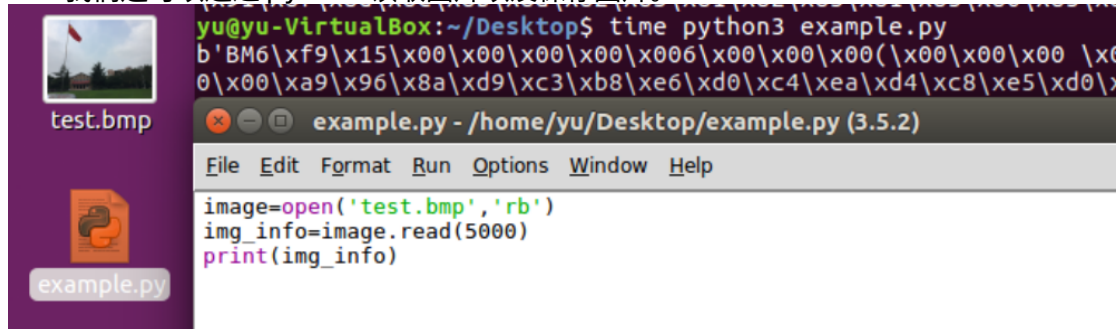
可以这样执行：

也可以点击 run，F5。个人建议在调试程序时使用第二种方式。但是第一种更加接近

我们的实验目的。

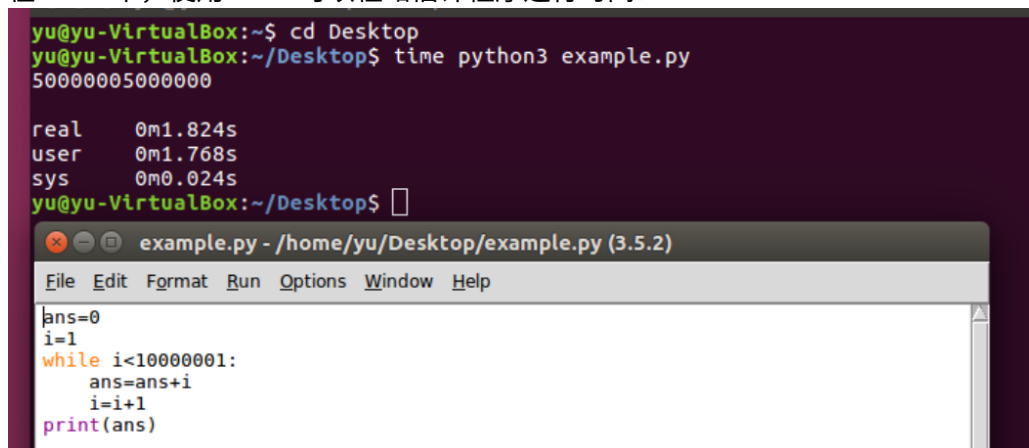
```
yu@yu-VirtualBox:~/Desktop/example$ python try.py
hello world
```

我们还可以通过 python 读取图片以及保存图片。



大家可以看到其实，每两个斜杠\\之间其实是十六进制数，图片

在 shell 中，使用 time 可以粗略估计程序运行时间：



使用程序可以查看内存使用，程序会发给大家。

第一张图是表示运行 PRmemory.py 所需要的内存

```
yu@yu-VirtualBox:~/Desktop$ python3 PRmemory.py
Memory Usage: 37498880.0 Byte

PRmemory.py - /home/yu/Desktop/PRmemory.py (3.5.2)
File Edit Format Run Options Window Help

if len(v) < 3:
    return 0.0 # invalid format?
# convert Vm value to bytes
return float(v[1]) * _scale[v[2]]

def memory(since=0.0):
    '''Return memory usage in bytes.
    ...
    return _VmB('VmSize:') - since

def resident(since=0.0):
    '''Return resident memory usage in bytes.
    ...
    return _VmB('VmRSS:') - since

print ("Memory Usage:", memory(), " Byte")
#print (call(["pmap", "-x", str(os.getpid())]))
```

第二张图是运行 memory.py 所需要的内存大小，这一个程序和上一个程序相比多了打开图片的部分：注意 test.bmp 与该程序在同一个目录下。

```
yu@yu-VirtualBox:~/Desktop$ python3 memory.py
Memory Usage: 39923712.0 Byte

memory.py - /home/yu/Desktop/memory.py (3.5.2)
File Edit Format Run Options Window Help

def resident(since=0.0):
    '''Return resident memory usage in bytes.
    ...
    return _VmB('VmRSS:') - since

img=open("test.bmp",'rb')
xx=b''
while True:
    line=img.readline()
    if not line:
        break
    else:
        xx=xx+line

print ("Memory Usage:", memory(), " Byte")
#print (call(["pmap", "-x", str(os.getpid())]))
```

小技巧：大家在调试程序时可以通过增添一个 print 语句来判断前面的语句是否正确或者是否被执行

练习：

1.提交一个名为 yourname.py 文档，其含有一个程序源代码。这个程序从一个名为

`example.txt` 的文本文档中读取两个长度为六的 `a b` 整数,分别计算 `a+b a-b a*b a mod b a/b a//b` 并将这个答案按顺序存储在一个名为 `answer` 的列表的 `0~5` 位, 并且该列表中也只能有六个元素, 打印出这个列表

例如: 输入 `4 5` 输出 `[7, -1, 3, 12, 0.75, 0]`

2. 提交一份 `pdf` 文档, 其中包含用 `python` 打开两张不同图片并且所用的时间, 所占用内存的命令行处的截图。并且写出对于时间复杂度和空间复杂度的理解 (100 字左右)。

3. 利用 `python` 程序实现稳定 `n` 个男生, `n` 个女生的稳定匹配问题 (提示: 两个循环的嵌套), 程序与结果写在 `pdf` 文档中。