

# python语言入门与实践 (七)

主讲人：吴陈炜 博士

复旦大学智能机器人研究院 工程博士

北京大学信息科学与技术学院 理学硕士

浙江大学信息与电子工程学系 工学学士

网易（杭州）网络有限公司 项目经理

美国项目管理协会 PMP（项目管理专业人士）

之江实验室 科研主管

**2020 年 5 月 21 日**

# 本节课计划

- 字典基本操作
- 集合基本操作
- Input函数
- while循环的使用

序列

index 0, 1, 2, ..., n-1

sort

# 字典

---

# 一个电话簿要如何存储

一个列表存名称，一个列表存号码

```
names = ['李雷', '韩梅梅', '马冬梅']  
phone_numbers = ['1234', '3456', '0123']  
print(phone_numbers[names.index('李雷')])
```

列表中奇数存名称，偶数存代码

```
phone_numbers = ['李雷', '1234', '韩梅梅', '3456', '马冬梅', '0123']  
print(phone_numbers[phone_numbers.index('李雷')+1])
```

# 字典结构

```
phone_numbers = {'李雷': '1234', '韩梅梅': '3456', '马冬梅': '0123'}  
print(phone_numbers['李雷'])
```

{key:value, key:value}

- 用花括号表示字典
- 字典内每一项都有两个元素组成: key和value
- 各个项用逗号隔开

# 字典结构-key

- key和value——对应，同一个键只能有一个对应的值，
- 键的类型是不可变的。

```
User1 = {'name': '李雷', 'numbers': '1234', 'name': '韩梅梅'}  
print(User1)
```

```
{'name': '韩梅梅', 'numbers': '1234'}
```

```
User1 = {['name']: '李雷', 'numbers': '1234'}  
print(User1)
```

```
User1 = {['name']: '李雷', 'numbers': '1234'}  
TypeError: unhashable type: 'list'
```



# 用dict函数创建字典

- 根据其他序列新建字典

```
message = [('lilei', 98), ('hanmeimei', 99)]  
d = dict(message)  
print(d)
```

- 根据关键字参数新建字典

```
d = dict(lilei = 98, hanmeimei = 99)  
print(d)
```

```
{ 'lilei': 98, 'hanmeimei': 99 }
```

# 访问字典里的数据

```
grade = {'李雷': '98', '韩梅梅': '99'}  
print(grade['李雷'])
```

利用中括号加要查询的key

x[y]其实是python优化过后的一种简洁表达式，它的函数原型是x.\_\_getitem\_\_(y)

Help on method\_descriptor:

\_\_getitem\_\_(...)

x.\_\_getitem\_\_(y) <==> x[y]

None

[Finished in 0.2s]



# 更新字典的元素

```
grade = {'李雷': '98', '韩梅梅': '99'}  
grade['韩梅梅'] = 100  
print(grade)
```

更新字典中的键值对

```
grade['马冬梅'] = '95'  
print(grade)
```

添加一个键值对

```
{ '李雷': '98', '韩梅梅': 100 }
```

```
{ '李雷': '98', '韩梅梅': 100, '马冬梅': '95' }
```

# 字典的删除操作

```
grade = {'李雷': '98', '韩梅梅': '99', '马冬梅': '95'}  
print(grade)
```

```
del grade['李雷']  
print(grade)
```

```
grade.clear()  
print(grade)
```

```
del grade  
print(grade)
```

删除了字典里的某一项

删除了字典里的每一项

删除了字典本身

```
{'李雷': '98', '韩梅梅': '99', '马冬梅': '95'}  
{'韩梅梅': '99', '马冬梅': '95'}  
{}
```

Traceback (most recent call last):

File "C:\Users\liym\Desktop\test\varia.py", line 165, in <module>

print(grade)

NameError: name 'grade' is not defined

# 字典的遍历

## ■ 根据key遍历

```
for friend in favorite_languages:
```

## ■ 根据value遍历

```
for language in favorite_languages.values():
```

## ■ 根据items遍历

```
for friend, language in favorite_languages.items():
```

```
favorite_languages = {  
    'jen'      : 'python',  
    'sarah'    : 'c',  
    'edward'   : 'ruby',  
    'phil'     : 'python'  
}
```

# 字典结构——嵌套字典

嵌套：

将一系列字典存储在列表中，或将列表作为值存在字典中。

---

- 字典列表
- 在字典中存储列表
- 在字典中存储字典

# 字典列表

- 由字典构成的列表。

```
student1 = {'name': '李雷', 'age': 18, 'grade': 98}
student2 = {'name': '韩梅梅', 'age': 19, 'grade': 99}
student3 = {'name': '马冬梅', 'age': 18, 'grade': 95}
```

```
student = [student1, student2, student3]
```

```
print(student)
```

```
[{'name': '李雷', 'age': 18, 'grade': 98},
{'name': '韩梅梅', 'age': 19, 'grade': 99},
{'name': '马冬梅', 'age': 18, 'grade': 95}]
```



# 在字典中存储列表

```
favorite_class = {  
    '李雷': ['数学', '英语'],  
    '韩梅梅': ['语文'],  
    '马冬梅': ['计算机', '物理', '数学'],  
}  
  
print(favorite_class['李雷'])  
print(favorite_class['李雷'][0])
```

```
['数学', '英语']  
数学
```



# 在字典中存储字典

```
#用一个字典表示一个学生信息
student1 = {'name': '李雷', '成绩': '98', '实验班': True}

#用一个字典表示全班学生信息
class1 = {
    '李雷': {'成绩': '98', '实验班': True},
    '韩梅梅': {'成绩': '95', '实验班': False},
}

print(class1['李雷'])
print(class1['李雷']['成绩'])
```

```
{ '成绩': '98', '实验班': True }
98
```

# 字典的排序

```
d = {  
    'a':2,  
    'b':5,  
    'c':3  
}
```

·根据key排序

```
for k in sorted(d):  
    print(k,d[k])
```

```
a 2  
b 5  
c 3  
[Finished in 0.3s]
```

·根据value排序

```
for k in sorted(d,key=d.__getitem__,reverse=True):  
    print(k,d[k])
```

```
b 5  
c 3  
a 2  
[Finished in 0.3s]
```

·根据items排序

```
res = sorted(d.items(),key=lambda d:d[1],reverse=True)  
print(res)
```

```
[('b', 5), ('c', 3), ('a', 2)]  
[Finished in 0.2s]
```

# 补充知识点1

## ■ sorted函数的用法

key参数用来在进行比较之前指定每个列表元素上要调用的函数，将函数的返回值作为比较的依据。那么如何使用这个key参数呢？有2种办法：

1、使用一个函数，这个函数的操作对象就是要比较的数据元素，返回结果就是某个可以用来比较的Python类型的数据；

2、使用匿名函数lambda，通过简单表达式返回某个可以用来比较的Python类型的数据。

## 补充知识点2

lambda函数又叫做匿名函数，函数的定义直接使用而不用起名字；  
lambda函数又称一句话函数、逻辑简单到一行代码就能表达的函数；  
常用于一些简单的、不会重复多次调用的场景。

```
1 g = lambda x:x+1
2 print(g(1))
3
4 def f(x):
5     return x+1
6 print(f(1))
```

返回值

形参

# 回顾总结

- 创建字典
- 访问字典里的元素
- 更新字典里的元素
- 字典的删除操作
  1. 删除字典中的某个元素
  2. 删除字典中每一个元素
  3. 删除字典本身
- 字典嵌套
  1. 字典列表 2.在字典里存储列表
  2. 在字典里存储字典

花括号/dict()

利用中括号加要查询的key

通过key更改value

del 字典[key]

字典.clear()

del 字典

- 字典的遍历和排序

# 集合

---



# 创建集合

1. 直接用花括号创建集合。

```
set1 = {1, 2, 4, 5, 8}
```

2. 使用set()方法

```
set1 = set([1, 2, 4, 1, 2, 8, 5, 5])  
set2 = set([1, 9, 3, 2, 5])  
  
print(set1)  
print(set2)
```

注意：

set()里需要加上[], 否则将默认集合里只有一个元素。

```
{1, 2, 4, 5, 8}  
{1, 2, 3, 5, 9}
```

# 集合的交集

交集 (Intersection) : 求两个集合中都出现了的元素。用&运算符实现。

```
set1 = {1, 2, 4, 5, 8}
set2 = {1, 2, 3, 5, 9}

print(set1 & set2)
```

```
{1, 2, 5}
```

# 集合的并集

**并集 (Union)**：求两个集合中共有的元素。用 `|` 运算符实现。

```
set1 = {1, 2, 4, 5, 8}
set2 = {1, 2, 3, 5, 9}
print(set1 | set2)
```

```
{1, 2, 3, 4, 5, 8, 9}
```

# 集合的差集

**差集 (Difference)**：求set1和set2的差集时，会返回在set1中但不在set2中的元素。

用 `-` 运算符实现。

```
set1 = {1, 2, 4, 5, 8}
set2 = {1, 2, 3, 5, 9}
print(set1 - set2)
```

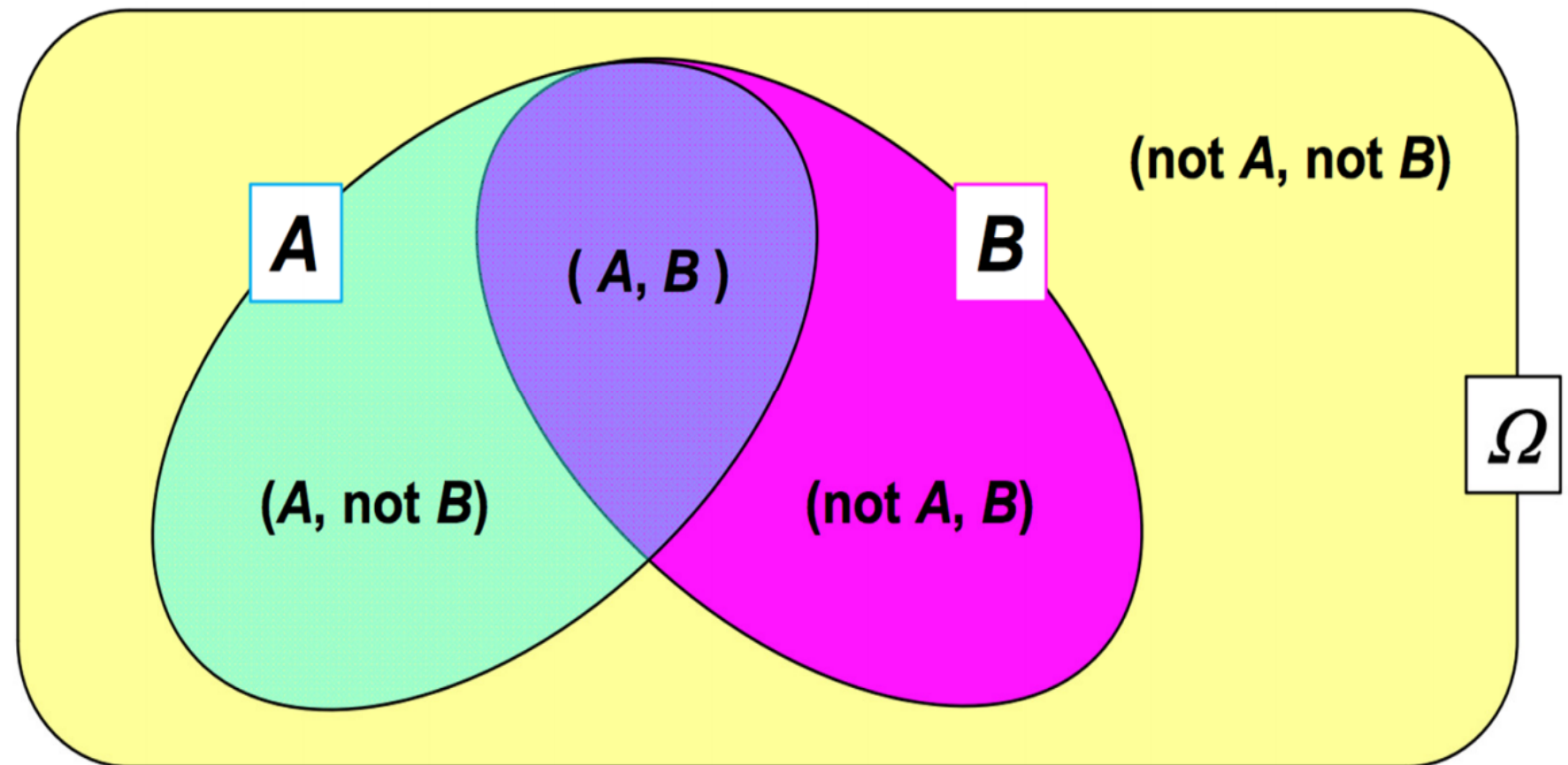
```
{8, 4}
```

# 集合的对称差集

**对称差集 (Symmetric Difference)**: 求set1和set2的对称差集 时, 会返回在set1中或在set2中, 但不同时存在于两个集合中的元素。

用 $\wedge$ 运算符实现。

```
set1 = {1, 2, 4, 5, 8}
set2 = {1, 2, 3, 5, 9}
print(set1 ^ set2)
{3, 4, 8, 9}
```



# 集合操作回顾总结

操作	运算符
交集	$\&$
并集	$ $
差集	$-$
对称差集	$\wedge$

思考一下：

对称差集可以用其他三种集合操作来实现吗？如何实现？



# Input函数的使用

---

# Input函数

我们编写程序最终目的还是来解决实际问题，所以必然会遇到输入输出的交互问题，python中提供了input函数用来获取用户的输入的“字符串”，我们可以用以下程序演示。

```
set_list.py x test.py x input_gender.py x
1 user_gender = input("Please enter your gender(F/M):")
2 print(f'Your gender is {user_gender}')
```

要注意的是在sublime编辑器中不支持input的在线输入，所以我们需要去cmd窗口运行这个程序，结果如下所示：

```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.18362.720]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\wuchenwei\Desktop\pythonCourSe\release\D2\test>python input_gender.py
Please enter your gender(F/M):M
Your gender is M
```

# 例子：判断用户输入的内容

```
user_gender = input("请输入您的性别(F/M): ")

if user_gender == 'F':
    print("你是萌妹子")
elif user_gender == 'M':
    print("你是糙汉子")
else:
    print("输入不正确, 请输入F或M")
```

如果收到的内容是F那么输出你是萌妹子

否则如果收到的内容是M那么输出你是糙汉子

如果既不是F也不是M那么告诉用户输错了

# 常见错误： 注意判断相等用双等号==

```
user_gender = input("请输入您的性别(F/M): ")  
user_is_student = input("您是学生吗?(Y/N): ")
```

单等号=是赋值

```
if user_gender == 'F':  
    if user_is_student == 'Y':  
        print("你是萌妹子学生")  
    elif user_is_student == 'N':  
        print("你是萌妹子")  
    else:  
        print("输入不正确")  
elif user_gender == 'M':  
    print("你是糙汉子")  
else:  
    print("输入不正确, 请输入F或M")
```

双等号==是判断相等

# 例子：判断用户输入的内容

```
user_gender = input("请输入您的性别(F/M): ")  
  
if user_gender == 'F':  
    print("你是萌妹子")  
elif user_gender == 'M':  
    print("你是糙汉子")  
else:  
    print("输入不正确, 请输入F或M")
```

如果没有else语句且前面条件都不符合则输出什么?

这段条件判断语句什么都不会输出

elif就是else if的缩写

条件判断会从第一个开始判断,

直到有一个符合条件的就不继续往下



# 使用多重if语句进行判断

```
user_gender = input("请输入您的性别(F/M): ")
user_is_student = input("您是学生吗?(Y/N): ")

if user_gender == 'F':
    if user_is_student == 'Y':
        print("你是萌妹子学生")
    elif user_is_student == 'N':
        print("你是萌妹子")
    else:
        print("输入不正确")
elif user_gender == 'M':
    print("你是糙汉子")
else:
    print("输入不正确, 请输入F或M")
```

注意不同层级的条件判断互不影响



# 练习题：

- 编写一个问答式简历程序。通过一句一句的提问获取用户的信息,生成一个格式化的简历。

```
C:\Users\wuchenwei\Desktop\pythoncrashcourse\D7>python test.py
请输入您的姓名：李雷
请输入您的性别：男
请输入您的年龄：20
请输入您的学校：复旦大学
正在生成您的简历.....
*****
                        简历
姓名：李雷
性别：男
年龄：20
学校：复旦大学
```

# while循环

---

# 1.while循环

## 1.1 while循环的基本格式

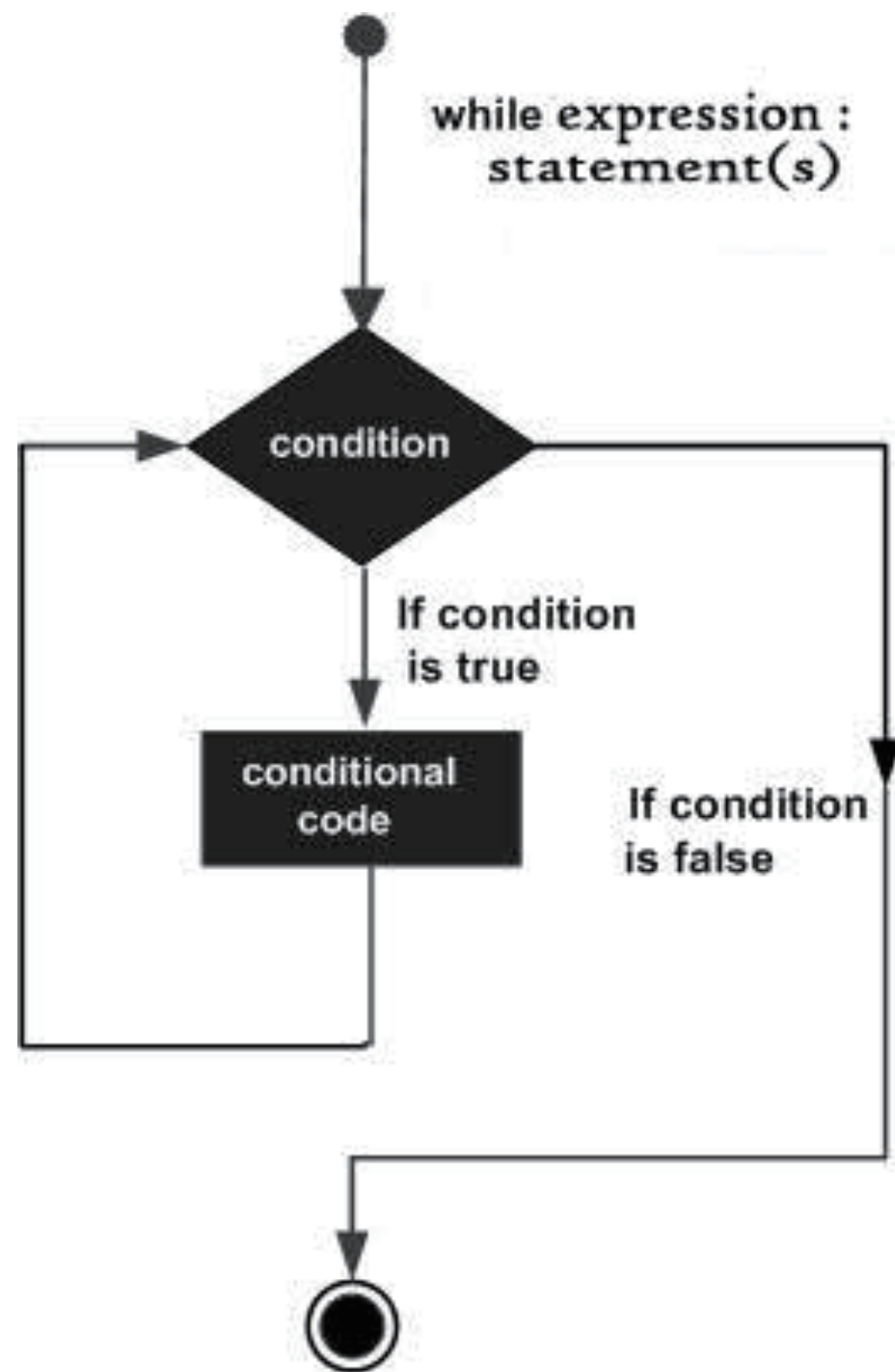
while循环的基本格式如下：

```
1 | while 条件表达式 :  
2 |     条件满足，执行循环语句；不满足，则退出循环
```

## 1.2.while循环示例程序

```
1 | count = 0  
2 | while (count<9):  
3 |     print(f'Now count is {count}')
```

```
4 |     count += 1
```



# while循环

```
i = 1

while i < 10:
    print(i)
    i = i + 1
```

```
1
2
3
4
5
6
7
8
9
```

**while循环：**当满足条件时一直执行里面的代码块

这段代码输出是什么？

# $i = i + 1?$

$i = i + 1$

左边是变量，右边是值

=表示赋值

理解：将右边视为一个计算式，算出来结果然后再赋予i

# break 和continue关键词

```
for i in range(10):  
    if i==5:  
        break  
    print(i)
```

```
0  
1  
2  
3  
4  
[Fin
```

```
for i in range(10):  
    if i==5:  
        continue  
    print(i)
```

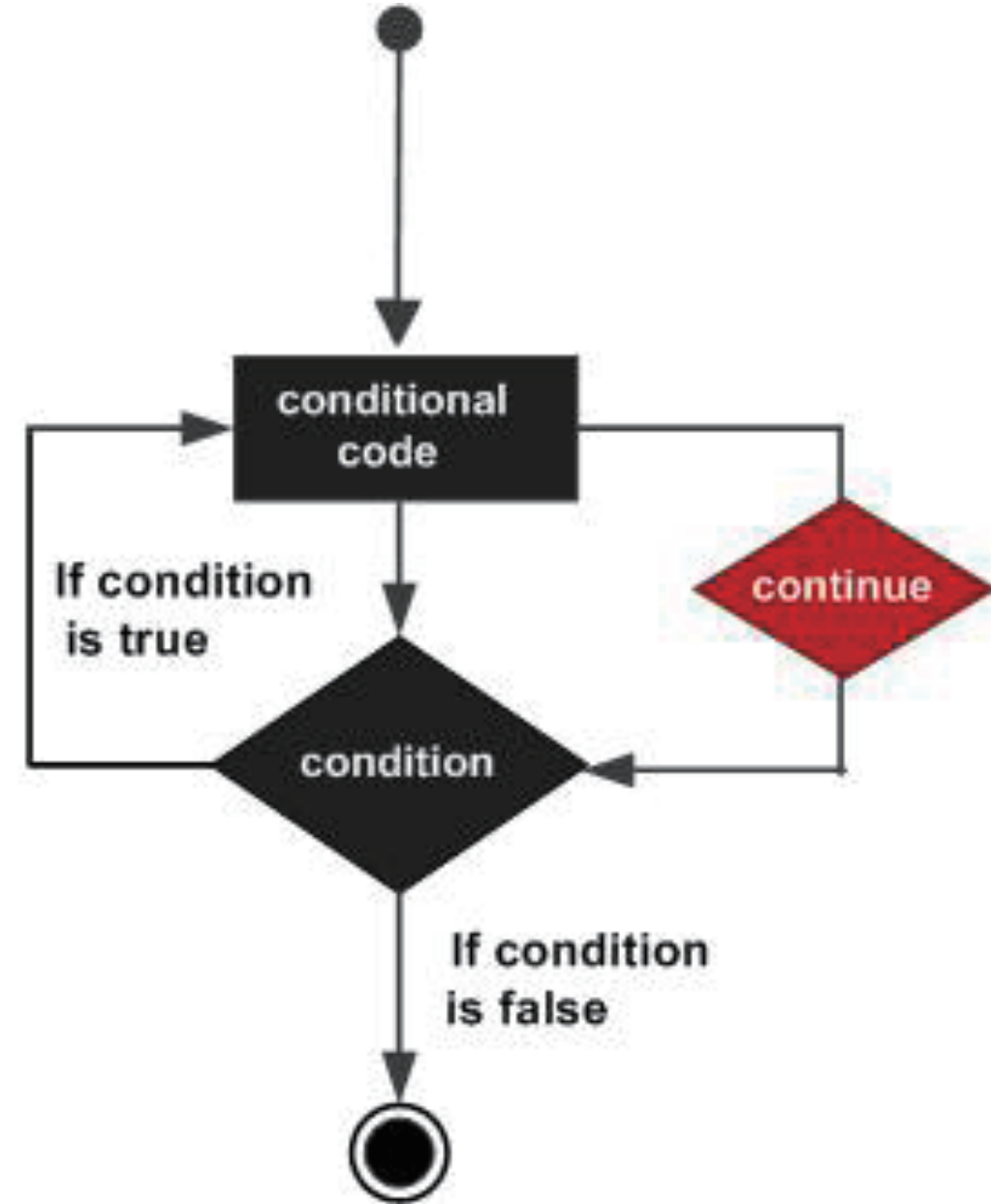
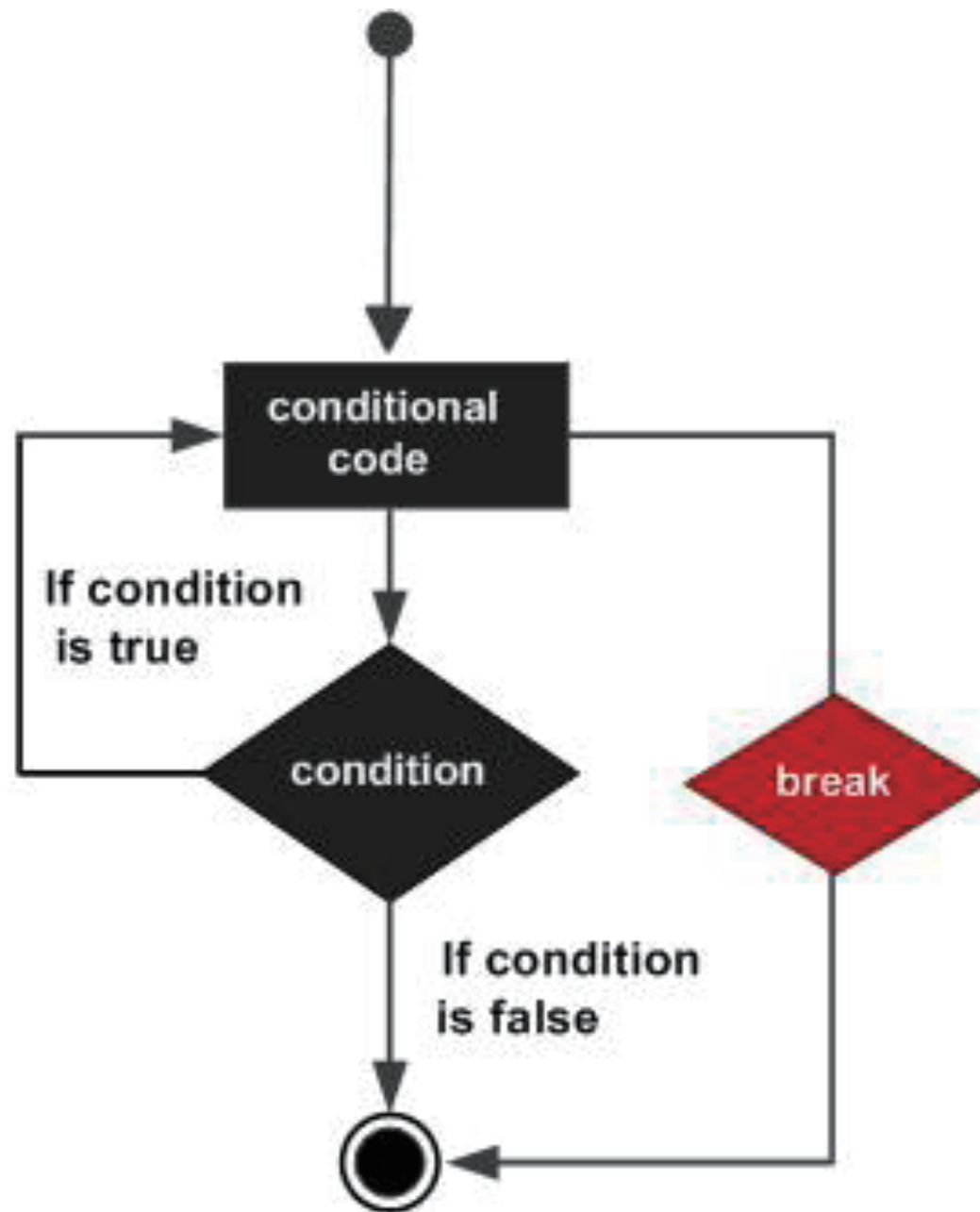
```
0  
1  
2  
3  
4  
6  
7  
8  
9  
[Fir
```

break跳出当前的循环语句，不执行剩下的循环

continue跳出当前这次的迭代，直接进入下一个迭代



# 图解break和continue



# 思考题

```
i = 0
while True:
    print(i)
    i = i + 1
    if i > 5:
        break
```

想一想输出是什么？

# pass 啥都不干，占位语句

```
while True:  
    pass
```

命令行或交互式解释器中强制退出死循环的程序

Ctrl+C (Mac OS: Control+C)

# 例子：判断用户输入的内容

```
user_gender = input("请输入您的性别(F/M): ")

if user_gender == 'F':
    print("你是萌妹子")
elif user_gender == 'M':
    print("你是糙汉子")
else:
    print("输入不正确, 请输入F或M")
```

当用户没有正确输入时如何重新获取用户输入？

# 例子：判断用户输入的内容

```
user_gender = input("请输入您的性别(F/M): ")
user_is_student = input("您是学生吗? (Y/N): ")

if user_gender == 'F':
    print("你是萌妹子学生")
elif user_gender == 'M':
    print("你是糙汉子")
else:
    print("输入不正确, 请输入F或M")
    user_gender = input("请输入您的性别(F/M): ")
    if user_gender == 'F':
        print("你是萌妹子学生")
    elif user_gender == 'M':
        print("你是糙汉子")
    else:
        print("输入不正确, 请输入F或M")
```

用户输错一次可以，几次都输错呢？



# while循环

```
user_answer_correct = False

while not user_answer_correct:
    user_gender = input("请输入您的性别(F/M): ")
    if user_gender == 'F':
        print("你是萌妹子学生")
        user_answer_correct = True
    elif user_gender == 'M':
        print("你是糙汉子")
        user_answer_correct = True
    else:
        print("输入不正确, 请输入F或M")
```

**while循环:** 当满足条件时一直执行里面的代码块



# 作业

- 完成本ppt中的习题
- 阅读《Python Crash Course》第六章和第七章并完成章节练习

<b>6</b>	<b>95</b>
<b>DICTIONARIES</b>	
A Simple Dictionary .....	96
Working with Dictionaries .....	96
Accessing Values in a Dictionary .....	97
Adding New Key-Value Pairs .....	97
Starting with an Empty Dictionary .....	98
Modifying Values in a Dictionary .....	99
Removing Key-Value Pairs .....	100
A Dictionary of Similar Objects .....	100
Exercise 6-1: Person .....	102
Exercise 6-2: Favorite Numbers .....	102
Exercise 6-3: Glossary .....	102
Looping Through a Dictionary .....	102
Looping Through All Key-Value Pairs .....	103
Looping Through All the Keys in a Dictionary .....	104
Looping Through a Dictionary's Keys in Order .....	106
Looping Through All Values in a Dictionary .....	107
Exercise 6-4: Glossary 2 .....	108
Exercise 6-5: Rivers .....	108
Exercise 6-6: Polling .....	108
Nesting .....	109
A List of Dictionaries .....	109
A List in a Dictionary .....	111

<b>7</b>	<b>117</b>
<b>USER INPUT AND WHILE LOOPS</b>	
How the input() Function Works .....	118
Writing Clear Prompts .....	118
Using int() to Accept Numerical Input .....	119
The Modulo Operator .....	120
Accepting Input in Python 2.7 .....	121
Exercise 7-1: Rental Car .....	121
Exercise 7-2: Restaurant Seating .....	121
Exercise 7-3: Multiples of Ten .....	121
Introducing while Loops .....	122
The while Loop in Action .....	122
Letting the User Choose When to Quit .....	122
Using a Flag .....	124
Using break to Exit a Loop .....	125
Using continue in a Loop .....	126
Avoiding Infinite Loops .....	126
Exercise 7-4: Pizza Toppings .....	127
Exercise 7-5: Movie Tickets .....	127
Exercise 7-6: Three Exits .....	128
Exercise 7-7: Infinity .....	128
Using a while Loop with Lists and Dictionaries .....	128
Moving Items from One List to Another .....	128
Removing All Instances of Specific Values from a List .....	129
Filling a Dictionary with User Input .....	130
Exercise 7-8: Deli .....	131
Exercise 7-9: No Pastrami .....	131
Exercise 7-10: Dream Vacation .....	131
Summary .....	131