python003

- Python条件语句是通过一条或多条语句的执行结果（True或者False）来决定执行的代码块。
- 分支结构
  - 单分支
    - 从流程图上我们可以看到，单分支就是只有一个判断，当条件成立的时候，去做操作，然后结束。否则直接结束；
    - a=5
    - if a<10:
    - 　　print('a is less than 10')
    - print (a)
  - **双分支**
    - 双分支就是当条件不成立的时候还多了一个操作，方便我们做逻辑处理；
    - a=5
    - if a<10:
    - 　print('a is less than 10')
    - else：
    - 　print('a is not less than 10')
    - print (a)
    - **双分支永远只有一个条件被执行；**
  -
  - **嵌套**
    - 嵌套即在if里面还可以有if,那有的时候我们可以把if逻辑合并，有的时候却要分开。判断的依据就是逻辑是否为一个整体；
    - a=5
    - if a<10:
    - 　print('a is less than 10')
    - else：
    - 　if < 20:
    - 　　print('10<=a<=20')
    - 　else:
    - 　　print('a is not less than 10')

        - print (a)
        - 通过结构嵌套实现多分支；
    -
    - 总结：分支结构永远只有1个或者0个分支会被执行；
    - 分支结构中的所有条件都是互斥的；
    - 条件只能是bool类型或者可以隐式转换为bool类型。
    -
- **if 语句**
    - 条件测试
        - 检查是否相等
        - 最简单的测试时检查变量的值是否相等；
            - car='bmw'
            - car=='bmw'
            - True
            -
            -
        - 检查是否相等时不考虑大小写
            - 在python中检查是否相等时区分大小写，2个大小写不同会被认为不相等
            - car='bmw'
            - car=='Bmw'
            - False
            - 如果大小写很重要，直接将变量转换为小写；
            - car='BMW'
            - car.lower()=='bmw'
            - 函数lower（）不会修改存储在变量car中的值；
        - 检查是否不相等使用
            - 可结合惊叹号和等号（！=)
        - 比较数字
            - 各种数字的比较：大于，小于，大于等于，小于等于；
            -
        - 检查多个条件
        - 1、使用and检查多个条件
        - 2、使用or检查多个条件
    - Python中if语句的一般形式如下所示：

- if condition_1: statement_block_1elif condition_2: statement_block_2else: statement_block_3
- 如果 "condition_1" 为 True 将执行 "statement_block_1" 块语句
- 如果 "condition_1" 为False，将判断 "condition_2"
- 如果"condition_2" 为 True 将执行 "statement_block_2" 块语句
- 如果 "condition_2" 为False，将执行"statement_block_3"块语句
- Python 中用 elif 代替了 elseif，所以if语句的关键字为：if – elif – else。
- 注意：
- 1、每个条件后面要使用冒号（:），表示接下来是满足条件后要执行的语句块。
- 2、使用缩进来划分语句块，相同缩进数的语句在一起组成一个语句块。
- 3、在Python中没有switch – case语句。
- 例如：计算狗的年龄
- age = int(input("请输入你家狗狗的年龄: "))
- print("")
- if age < 0:
- print("你是在逗我吧!")
- elif age == 1:
- print("相当于 14 岁的人。")
- elif age == 2:
- print("相当于 22 岁的人。")
- elif age > 2:
- human = 22 + (age -2)*5
- print("对应人类年龄: ", human)
- ### 退出提示
- input("点击 enter 键退出")

- if 嵌套
- 在嵌套 if 语句中，可以把 if...elif...else 结构放在另外一个if...elif...else 结构中。
- if 表达式1:
- 语句
- if 表达式2:
- 语句
- elif 表达式3:
- 语句
- else：
- 语句

- elif表达式4:
- 语句
- else:
- 语句
- 例：输入一个数是否能可以整除2或3
-

```
num=int(input("输入一个数字: "))
if num%2==0:
    if num%3==0:
        print ("你输入的数字可以整除 2 和 3")
    else:
        print ("你输入的数字可以整除 2，但不能整除 3")
else:
    if num%3==0:
        print ("你输入的数字可以整除 3，但不能整除 2")
    else:
        print ("你输入的数字不能整除 2 和 3")
```

-
- 循环语句
- Python中的循环语句有 for 和 while。
  - for循环用于针对集合中每个元素的一个代码块，而while循环不断的进行，直到指定条件终止；
  - 同样需要注意冒号和缩进。另外，在Python中没有do..while循环。
  - 例如计算 1 到 100 的总和：
  - n = 100
  - sum = 0
  - counter = 1
  - while counter <= n:
  - sum = sum + counter　counter += 1
  - print("1 到 %d 之和为: %d" % (n,sum))
  -
  - **无限循环**
  - 我们可以通过设置条件表达式永远不为 false 来实现无限循环，实例如下：
  - var = 1
  - while　var == 1 :
  - # 表达式永远为 true
  - num = int(input("输入一个数字 :"))
  - print ("你输入的数字是: ", num)

- print ("Good bye!")
-
- 你可以使用 CTRL+C 来退出当前的无限循环。
- 无限循环在服务器上客户端的实时请求非常有用。

-
- **while循环使用 else 语句**
- 在 while … else 在条件语句为 false 时执行 else 的语句块：
- count = 0
- while count < 5:
- print (count, " 小于 5")
- count = count + 1
- else:
- print (count, " 大于或等于 5")
-
-
- **for 语句**
- Python for循环可以遍历任何序列的项目，如一个列表或者一个字符串。
- for循环的一般格式如下：
- for <variable> in <sequence>:
- <statements>
- else:
- <statements>
- Python loop循环实例：
- languages = ["C", "C++", "Perl", "Python"] >>>
- for x in languages:
- print (x).
- Python的for循环的在for关键字里实现了迭代协议，通过迭代协议完成了可迭代对象的循环；
- 可迭代对象：能用next()函数进行下一个对象的访问；
- 注意：for in循环里永远不要修改可迭代对象
- python3中，range就是一个对象，是一个生成器对象，要通过for循环去迭代获取，而不能直接使用p[0]获取；
- python3中获取range(10)的列表，直接用list转化就可以了；list(range(10))
-
-
-

- **range()函数**
- 如果你需要遍历数字序列，可以使用内置range()函数。它会生成数列，例如:
- >>> for i in range(5):
- print(i)
-
- **break 语句：break 语句用于跳出当前循环体：提前终止**
- for i in range（0，,10):
- if i % 2!= 0:
- print(i)
- break
- print('break ')
- for 循环体内，遇到break关键字的时候，将会结束本层循环，注意不是本次循环;
-
-
- **break和continue语句及循环中的else子句**
- break 语句可以跳出 for 和 while 的循环体。
- 如果你从 for 或 while循环中终止，任何对应的循环 else 块将不执行。
-

```
for letter in 'Running':
    if letter == 'g':
        break
    print ('当前字母为 :', letter)
```

- continue语句被用来告诉Python跳过当前循环块中的剩余语句，然后继续进行下一轮循环。
-

```
for letter in 'Running':
    if letter == 'i':          # 字母为 i 时跳过输出
        continue
    print ('当前字母 :', letter)
```

-
-