学习目标

1. 掌握在 if 单分支语法;
2. 掌握 and,or,not 逻辑运算符;
3. 掌握 if 多分支语法;

目录

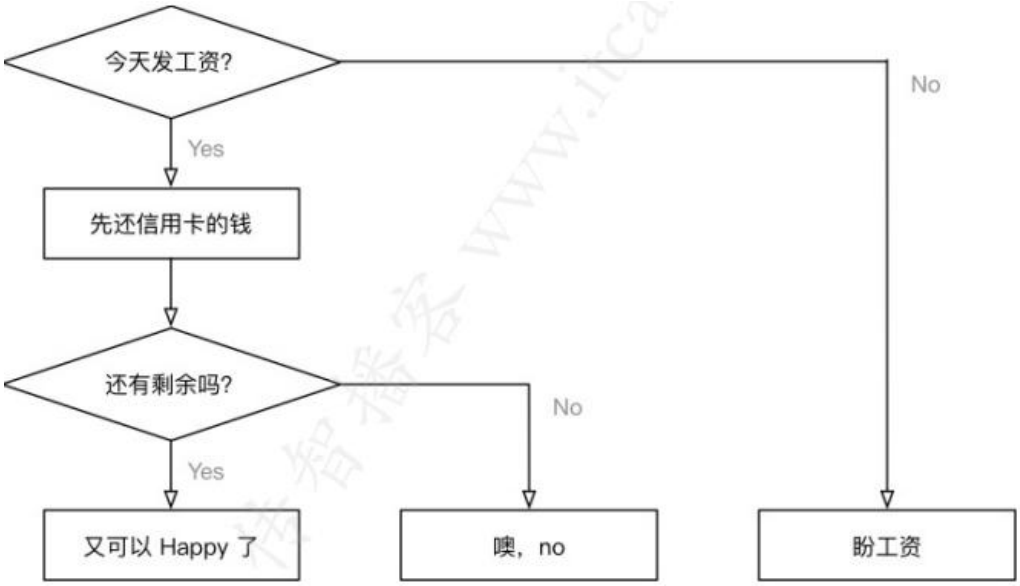
第 1 章 分支结构------基本说明..... 3
第 2 章 分支结构------if 单分支.....5
第 3 章 分支结构------逻辑运算符.....10
第 4 章 分支结构------if 多分支.....12
第 5 章 分支结构------运算符总结.....16
第 6 章 分支结构------综合应用.....18

第 1 章 分支结构------基本说明

一、条件判断

生活中的判断几乎是无所不在的，我们每天都在做各种各样的选择，如果这样？如果那样？

二、程序中的判断



if 今天发工资:

　　先还信用卡的钱

if 有剩余:

　　又可以 happy 了，O(∩_∩)O 哈哈~

else:

　　噢，no。。。还的等 30 天

else:

　　盼着发工资

三、判断的定义

- 如果条件满足，才能做某件事情，
- 如果条件不满足，就做另外一件事情，或者什么也不做 。

正是因为有了判断，才使得程序世界丰富多彩，充满变化！

判断语句又被称为“分支语句”，正是因为有了判断，才让程序有了很多的分支

第 2 章 分支结构------if 单分支

1. 比较运算符

比较运算符	说明
>	大于
>=	大于等于
<	小于
<=	小于等于
==	等于
!=	不等于

2. if 判断语句基本语法

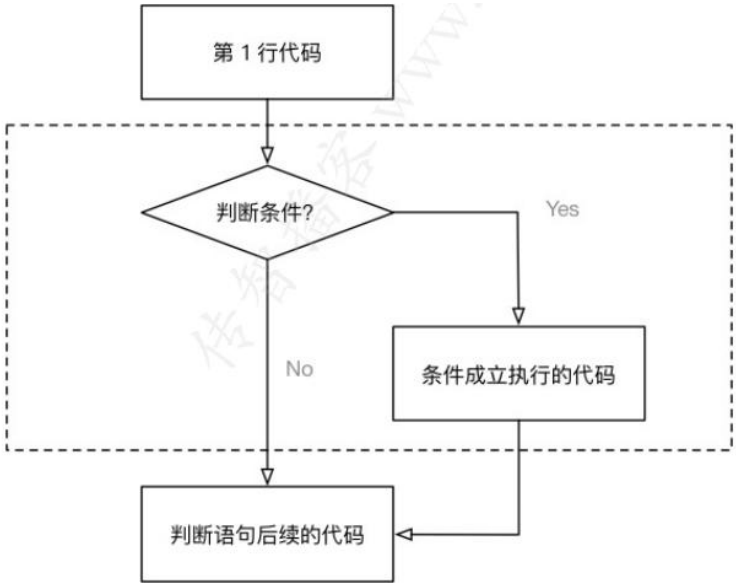
在 Python 中，if 语句 就是用来进行判断的，格式如下：

```
if 要判断的条件:
    条件成立时，要做的事情
……
```

注意：代码的缩进为一个 tab 键，或者 4 个空格 —— 建议使用空格

在 Python 开发中，Tab 和空格不要混用！

可以把整个 if 语句看成一个完整的代码块



3. pass 占位符

当 if 语句内容为空的时候, 如果没有任何代码程序会报错, 此时可以使用 pass 做为占位符

```
if 要判断的条件:
    pass
```

4. 判断语句演练 —— 判断年龄

- 需求

1. 定义一个变量 age 记录年龄 ,用 input 输入年龄

2. 判断是否满 18 岁 （>=）

3. 如果满 18 岁，允许进网吧嗨皮

```
# 定义变量age 存放年龄
age = int(input("请输入年龄"))
# 判断年龄是否大于18
if age >= 18:
    print("可以进网吧嗨皮")  # 缩进表示print 和if 是同一个代码块

print("程序执行完毕")
# 没有缩进，和if 语句不是一个代码块，无论if 判断如何，都会执行
```

注意:

if 语句以及缩进部分是一个完整的代码块

5. 课堂练习---

判断除数是否为 0

num1 = 通过 input 函数输入的任意数字

num2 = 通过 input 函数输入的任意数字

如果 **num2** 不等于 **0**，计算 **num1** 除以 **num2** 的结果。

6. 课堂练习----

计算器

num1 = 通过 input 函数输入的任意数字

num2 = 通过 input 函数输入的任意数字

a = 通过 input 函数输入 + - * / 中的任意一个字符

如果 a 的值为+， 那么显示 num1 和 num2 相加的结果

如果 a 的值为-， 那么显示 num1 和 num2 相减的结果

如果 a 的值为*， 那么显示 num1 和 num2 相乘的结果

如果 a 的值为/， 那么显示 num1 和 num2 相除的结果

7. if 双分支

- 思考

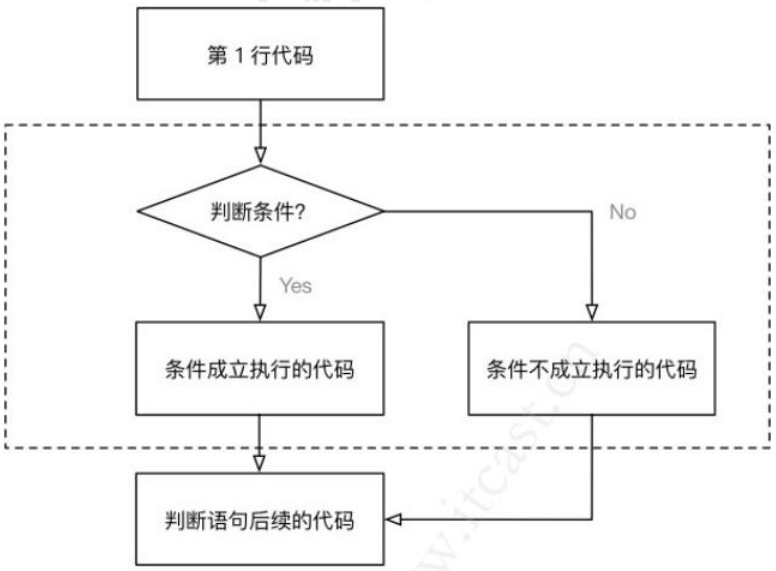
在使用 if 判断时，只能做到满足条件时要做的事情。那如果需要在不满足条件的时候，做某些事情，该如何做呢?

else 实现双分支 ， 格式如下:

```
if 要判断的条件:
    条件成立时，要做的事情
else:
    条件不成立时，要做的事情
```

注意:

if 和 else 语句以及各自的缩进部分共同是一个完整的代码块



8. 判断语句演练 —— 判断年龄改进

- 需求

1. 定义一个变量 age 记录年龄 ,用 input 输入年龄

2. 判断是否满 18 岁 （>=）

3. 如果满 18 岁，允许进网吧嗨皮

4. 如果不满 18 岁，回家写作业

```
# 定义变量 age 存放年龄
age = int(input("请输入年龄"))
# 判断年龄是否大于 18
if age >= 18:
    print("可以进网吧嗨皮")  # 缩进表示 print 和 if 是同一个代码块
else:  # age 小于 18
    print("回家写作业")

print("程序执行完毕")
# 没有缩进，和 if 语句不是一个代码块，无论 if 判断如何，都会执行
```

9. 课堂练习---

判断偶数

num1 的值为通过 input 函数输入的任意整数

判断 num1 是偶数还是奇数

如果为偶数，print 显示"偶数"，如果为奇数 print 显示"奇数"

10. 课堂练习---

判断正负数

num1 的值为通过 input 函数输入的任意整数

判断 num1 是正数还是负数(假设 0 为正数)

如果为正数，print 显示"正数"，如果为负数 print 显示"负数"

第 3 章 分支结构------逻辑运算符

在程序开发中，通常在判断条件时，会需要同时判断多个条件 。

只有多个条件都满足，才能够执行后续代码，这个时候需要使用到逻辑运算符。

逻辑运算符可以把多个条件按照逻辑进行连接，变成更复杂的条件。

Python 中的逻辑运算符包括：and 与 / or 或 / not 非 三种 。

一、and（与）

条件 1 and **条件 2**

- 两个条件同时满足，返回 True
- 只要有一个不满足，就返回 False

1. 案例--只有姓名为”小明”并且年龄大于 20，才能通过

```
name = "小明"
age = 25
if name == "小明" and age >= 20:
```

```
    print("通过了")
else:
    print("不能通过")
```

二、or（或）

条件 1 or **条件 2**

- 两个条件只要有一个满足，返回 True
- 两个条件都不满足，返回 False

2. 案例--只要姓名为”小明”或者年龄大于 20，都能通过

```
name = "小明"
age = 25
if name == "小明" or age >= 20:
    print("通过了")
else:
    print("不能通过")
```

三、not（非）

not **条件**

3. 案例--只有姓名不叫”小明”才能通过

```
name = "小明"
if not name == "小明":
    print("通过了")
else:
    print("不能通过")
```

四、逻辑运算课堂练习

1. 课堂练习---

通过 input 输入一个数 ， 编写代码判断该数是否在 0 到 120 之间 。

2. 课堂练习---

name 的值为通过 input 函数输入的登录账号

passwd 的值为通过 input 函数输入的登录密码

只有 name 的值为"itcast",并且 passwd 的值为"123456",显示“通过登录”，

否则显示“登录失败”

第 4 章 分支结构------if 多分支

一、 elif 语句

在开发中，使用 if 可以判断条件

使用 else 可以处理 条件不成立的情况

但是，如果希望再增加一些条件，条件不同，需要执行的代码也不同 时，就可以使用 elif

语法格式如下：

```
if 条件 1:
    条件 1 成立时，要做的事情
elif 条件 2:
    条件 2 成立时，要做的事情
elif 条件 3:
    条件 2 成立时，要做的事情
else:
    所有条件不成立时，要做的事情
```

注意

1. elif 和 else 都必须和 if 联合使用，而不能单独使用

2. 可以将 if 、 elif 和 else 以及各自缩进的代码，看成一个完整的代码块

二、elif 演练 —— 女友的节日

- 需求

1. 如果是情人节应该买玫瑰 ；
2. 如果是平安夜应该吃大餐；
3. 如果是生日应该买蛋糕 ；
4. 其他的日子该上班了 。

```
hday = input("请输入")
if hday == "情人节":
    print("买玫瑰")
elif hday == "平安夜":
    print("吃大餐")
elif hday == "生日":
    print("吃蛋糕")
else:
    print("该上班了")
```

1. 课堂练习---

```
age 的值为通过 input 函数输入的任意整数
```

如果 age 小于 10，显示“小孩”

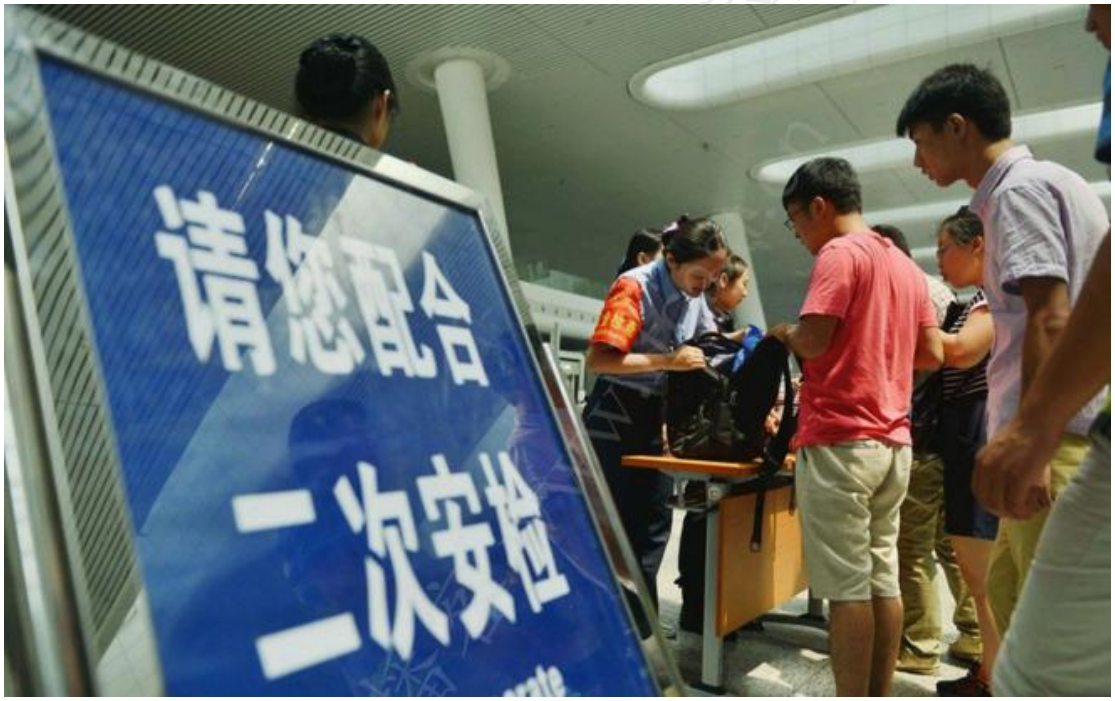
如果 age 在 10 到 20 之间，显示“小朋友”

如果 age 在 20 到 30 之间，显示“年轻人”

如果 age 在 30 到 50 之间，显示“中年人”

如果 age 大于 50，显示“老年人”

三、if 嵌套



elif 的应用场景是：同时判断多个条件，所有的条件是平级的 。

在开发中，使用 if 进行条件判断，如果希望在条件成立的执行语句中再增加条件判断，就可以使用 if 的嵌套。

if 的嵌套的应用场景就是：在之前条件满足的前提下，再增加额外的判断。

if 的嵌套的语法格式，除了缩进之外和之前的没有区别 。

语法格式如下：

```
if 条件 1:
    条件 1 满足执行的代码
    ……
    if 条件 1 基础上的条件 2:
        条件 2 满足时，执行的代码
        ……
    # 条件 2 不满足的处理
    else:
        条件 2 不满足时，执行的代码
# 条件 1 不满足的处理
else:
    条件 1 不满足时，执行的代码
    ……
```

四、if 嵌套演练 —— 判断 0 到 100 以内的任意一个数字是否能被 3 整除

```
num1 = int(input("请输入"))
if num1 > 0 and num1 < 100:
    if num1 % 3 == 0:
        print("能整除")
    else:
        print("不能整除")
else:
    print("num1 不在范围之内")
```

1. 课堂练习---

```
name 的值为通过 input 函数输入的字符串

如果 name 的值为"tom"

    通过 input 函数输入 age 的值，
```

如果 age 大于等于 30 显示“大叔”

如果 age 小于 30 显示“小弟”

如果 name 的值不为"tom"

显示“姓名错误”

第 5 章 分支结构------运算符总结

一、算数运算符

是完成基本的算术运算使用的符号，用来处理四则运算。

运算符	描述	实例
+	加	10 + 20 = 30
-	减	10 - 20 = -10
*	乘	10 * 20 = 200
/	除	10 / 20 = 0.5
//	取整除	返回除法的整数部分（商） 9 // 2 输出结果 4
%	取余数	返回除法的余数 9 % 2 = 1
**	幂	又称次方、乘方，2 ** 3 = 8

在 Python 中 * 运算符还可以用于字符串，计算结果就是字符串重复指定次数的结果 。

二、比较（关系）运算符

运算符	描述
==	检查两个操作数的值是否 **相等**，如果是，则条件成立，返回 True
!=	检查两个操作数的值是否 **不相等**，如果是，则条件成立，返回 True
>	检查左操作数的值是否 **大于** 右操作数的值，如果是，则条件成立，返回 True
<	检查左操作数的值是否 **小于** 右操作数的值，如果是，则条件成立，返回 True
>=	检查左操作数的值是否 **大于或等于** 右操作数的值，如果是，则条件成立，返回 True
<=	检查左操作数的值是否 **小于或等于** 右操作数的值，如果是，则条件成立，返回 True

三、逻辑运算符

运算符	逻辑表达式	描述
and	x and y	只有 x 和 y 的值都为 True，才会返回 True 否则只要 x 或者 y 有一个值为 False，就返回 False
or	x or y	只要 x 或者 y 有一个值为 True，就返回 True 只有 x 和 y 的值都为 False，才会返回 False
not	not x	如果 x 为 True，返回 False 如果 x 为 False，返回 True

四、赋值运算符

在 Python 中，使用 = 可以给变量赋值 。

在算术运算时，为了简化代码的编写， Python 还提供了一系列的与算术运算符对应的赋值运算符 。

注意：赋值运算符中间不能使用空格

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c = a 等效于 c = c a
/=	除法赋值运算符	c /= a 等效于 c = c / a
//=	取整除赋值运算符	c //= a 等效于 c = c // a
%=	取 模 (余数)赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c = a 等效于 c = c a

五、运算符的优先级

以下表格的算数优先级由高到最低顺序排列

运算符	描述
**	幂 (最高优先级)
* / % //	乘、除、取余数、取整除
+ -	加法、减法
<= < > >=	比较运算符
== !=	等于运算符
= %= /= //= -= += = *=	赋值运算符
not or and	逻辑运算符

第 6 章 分支结构------综合应用

一、 随机数与猜拳游戏

1. 需求

- 1：代表石头 / 2：代表剪刀 / 3：代表布；
- 从控制台输入要出的拳，输入 1 或 2 或 3；
- 电脑随机出拳 ；
- 比较胜负。

序号	规则
1	石头 胜 剪刀
2	剪刀 胜 布
3	布 胜 石头

2. 随机数处理

在 Python 中，要使用随机数，首先需要导入随机数的模块。

import random

导入模块后，调用函数 randint 生成一个随机数。

random.randint(a, b) ，返回 [a, b] 之间的整数，包含 a 和 b

例如：

```
# 导入生成随机数的模块
import random
# 生成从 10 到 20 之间的一个随机整数
a = random.randint(10, 20)
print(a)
# 生成从 0 到 5 之间的一个随机整数
a = random.randint(0, 5)
print(a)
```

3. 猜拳游戏实现

```
# 导入生成随机数的模块
import random
# pc 为电脑，通过调用 randint 函数得到一个从 1 到 3 的随机数
pc = random.randint(1, 3)
# player 代表我，具体值从键盘输入

player = int(input("请输入"))

if (player == 1 and pc == 2) or (player == 2 and pc == 3) or (player
== 3 and pc == 1):

    print("电脑出的%d，我出的%d，我赢了" % (pc, player))

elif player == pc:
```

```
    print("平局")
else:
    print("电脑出的%d，我出的%d，我输了" % (pc, player))
```