基础语法(2)

顺序语句

默认情况下, Python 的代码执行顺序是按照从上到下的顺序, 依次执行的.

```
print("1")
print("2")
print("3")
```

执行结果一定为 "123", 而不会出现 "321" 或者 "132" 等. 这种按照顺序执行的代码, 我们称为 **顺序语句**.

> 这个顺序是很关键的. 编程是一件明确无歧义的事情. 安排好任务的顺序, 计算机才能够正确的进行执行.
>
> 就好像人生的顺序, 是上学, 工作, 结婚, 生子. 一旦这里的顺序乱了, 就很麻烦.

条件语句

什么是条件语句

条件语句能够表达 "如果 ... 否则 ..." 这样的语义. 这构成了计算机中基础的 **逻辑判定**.

条件语句 也叫做 **分支语句**, 表示了接下来的逻辑可能有几种走向.

> 一个典型的例子:
>
> 如果丘处机没有路过牛家村,
>
> 1. 那么金兵不会死在郭, 杨两家手上
> 2. 郭, 杨两家就不会流亡北方
> 3. 郭夫人就不会去到大漠, 完颜洪烈就不会遇到包惜弱
> 4. 郭靖就不会和江南七怪救了铁木真
> 5. 蒙古就不会统一
> 6. 蒙古铁骑就不会西征
> 7. 欧洲就不会出现火药, 也就不会出现文艺复兴, 大航海.
> 8. 大炮就不会从欧洲传到日本, 日本得不到统一
> 9. 完颜洪烈就不会全力战, 金国内乱
> 10. 宋朝不会灭亡, 并诞生出资本主义. 中国成为最发达的国家.

```
如果  我认真敲代码
      我就很容易找到工作
否则
      我就容易毕业就失业
```

其中 "我认真敲代码" 称为 **条件**. 如果条件成立(条件为真), 则会出现 "我就很容易找到工作" 这个情况. 如果条件不成立(条件为假), 则会出现 "我就容易毕业就失业".

当然, 同样的逻辑, 还可以反着表达.

```
如果  我选择躺平摆烂
    我就容易毕业就失业
否则
    我就很容易找到工作
```

虽然结构变了, 但是整体表达的语义是等价的.

> PS: 亲爱的同学们, 你们是选择认真敲代码, 还是躺平摆烂呢?

语法格式

Python 中使用 if else 关键字表示条件语句.

(1) `if`

```
if expression:
    do_something1
    do_something2
next_something
```

如果 expression 值为 True, 则执行 do_something1, do_something2, next_something

如果 expression 值为 False, 则只执行 next_something, 不执行 do_something1, do_something2

(2) `if - else`

```
if expression:
    do_something1
else:
    do_something2
```

如果 expression 值为 True, 则执行 do_something1

如果 expression 值为 False, 则执行 do_something2

(3) `if - elif - else`

```
if expression1:
    do_something1
elif expression2:
    do_something2
else:
    do_something3
```

如果 expression1 值为 True, 则执行 do_something1

如果 expression1 值为 False, 并且 expression2 为 True 则执行 do_something2

如果 expression1 值为 False, 并且 expression2 为 False 则执行 do_something3

注意: Python中的条件语句写法, 和很多编程语言不太一样.

- if 后面的条件表达式, 没有 ( ), 使用 : 作为结尾.
- if / else 命中条件后要执行的 "语句块", 使用 **缩进** (通常是 4 个空格或者 1 个 tab)来表示, 而不是 { }
- 对于多条件分支, 不是写作 else if, 而是 elif (合体了).

示例: 输入 1 表示愿意认真学习, 输入 2 表示躺平摆烂.

```
choice = input("输入 1 表示认真学习, 输入 2 表示躺平摆烂: ")

if choice == "1":
    print("你会找到好工作!")
elif choice == "2":
    print("你可能毕业就失业了!")
else:
    print("你的输入有误!")
```

缩进和代码块

代码块 指的是一组放在一起执行的代码.

在 Python 中使用缩进表示代码块. 不同级别的缩进, 程序的执行效果是不同的.

```
# 代码1
a = input("请输入一个整数: ")
if a == "1":
    print("hello")
    print("world")
```

```
# 代码2
a = input("请输入一个整数: ")
if a == "1":
    print("hello")
print("world")
```

注意上述代码的区别.

在代码1 中, `print("world")` 有一级缩进, 这个语句属于 if 内的代码块, 意味着条件成立, 才执行, 条件不成立, 则不执行.

在代码2 中, `print("world")` 没有缩进, 这个语句是 if 外部的代码, 不属于 if 内部的代码块. 意味着条件无论是否成立, 都会执行.

另外, 代码块内部还可以嵌套代码块.

```
a = input("请输入第一个整数: ")
b = input("请输入第二个整数: ")

if a == "1":
    if b == "2":
        print("hello")
    print("world")
print("python")
```

在这个代码中,

- `print("hello")` 具有两级缩进, 属于 `if b == "2"` 条件成立的代码块.
- `print("world")` 具有一级缩进, 属于 `if a == "1"` 条件成立的代码块.
- `print("python")` 没有缩进, 无论上述两个条件是否成立, 该语句都会执行.

基于缩进的方式表示代码块, 带来的好处就是强制要求程序猿要写明确的缩进, 来明确代码之间的相对关系. 如果缩进书写的不对, 则直接报错.

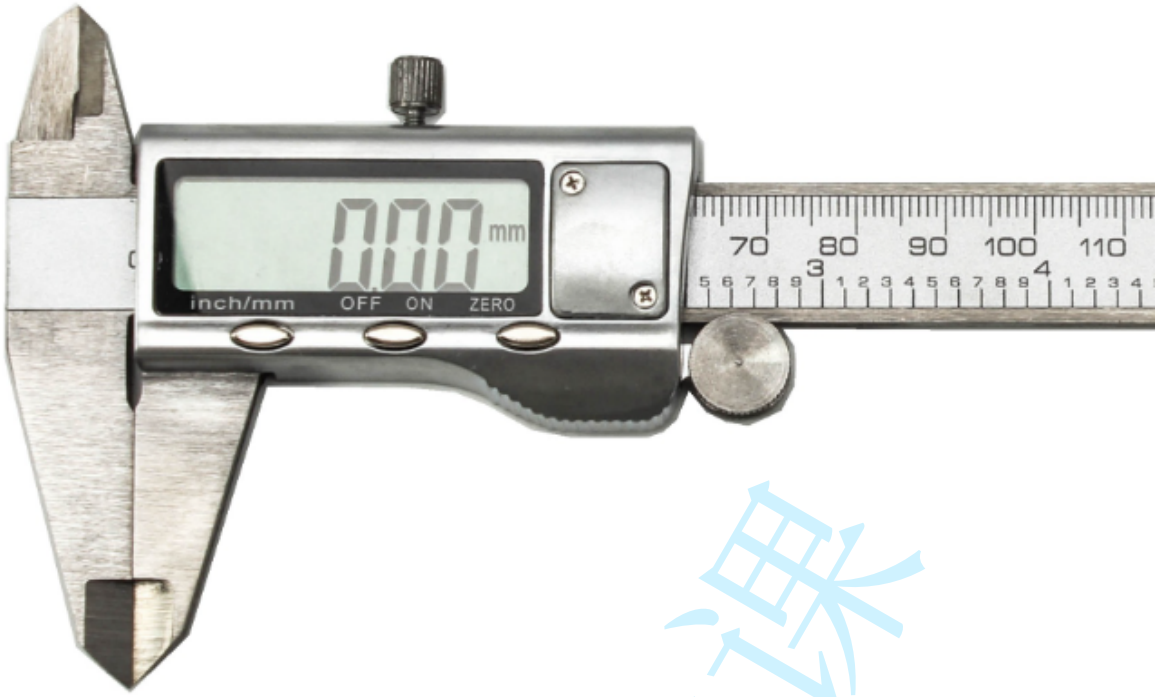
> 像 C++ / Java 这些语言, 即使完全不写缩进, 语法也不会报错. 代码可读性就比较差.

同时, 带来的坏处就是, 如果缩进层次比较多, 就容易分不清楚某个语句属于哪个层级.

```
if a == 1:
    if b == 2:
        if c == 3:
            if d == 4:
                if e == 5:
                    if f == 6:
                        if g == 7:
                            print("hello")
                    print("1")
        print("2")
```

> 请问, 上述代码中的 `print("1")` 和 `print("2")` 属于哪一级缩进?

因此, 就有了 "写 Python 需要自备游标卡尺" 这个梗.



练习

(1) 输入一个整数, 判定是否是奇数

```
a = int(intput("请输入一个整数: "))
if a % 2 == 0:
    print("偶数")
else:
    print("奇数")
```

(2) 输入一个整数, 判定是正数还是负数

```
a = int(input("请输入一个整数: "))
if a > 0:
    print("正数")
elif a < 0:
    print("负数")
else:
    print("为 0")
```

(3) 判定年份是否是闰年

```
year = int(input("请输入年份: "))

if year % 100 == 0:
    # 判定世纪闰年
    if year % 400 == 0:
        print("闰年")
    else:
```

```
        print("平年")
else:
    # 判定普通闰年
    if year % 4 == 0:
        print("闰年")
    else:
        print("平年")
```

```
year = int(input("请输入年份: "))

if (year % 100 != 0 and year % 4 == 0) or year % 400:
    print("闰年")
else:
    print("平年")
```

空语句 pass

代码示例: 输入一个数字, 如果数字为 1, 则打印 hello.

```
a = int(input("请输入一个整数:"))
if a == 1:
    print("hello")
```

这个代码也可以等价写成

```
a = int(input("请输入一个整数:"))
if a != 1:
    pass
else:
    print("hello")
```

其中 `pass` 表示 **空语句**, 并不会对程序的执行有任何影响, 只是占个位置, 保持 Python 语法格式符合要求.

> 如果代码写作
>
> ```
> a = int(input("请输入一个整数:"))
> if a != 1:
>
> else:
>  print("hello")
> ```
>
> 程序是不符合 Python 语法的, 会直接报错.

循环语句

有些操作是需要反复执行的. 这种就需要使用循环.

while 循环

基本语法格式

```
while 条件:
    循环体
```

- 条件为真, 则执行循环体代码.
- 条件为假, 则结束循环.

代码示例: 打印 1-10 的整数

```python
num = 1
while num <= 10:
    print(num)
    num += 1
```

代码示例: 计算 1-100 的和

```python
sum = 0
num = 1
while num <= 100:
    sum += num
    num += 1
print(sum)
```

代码示例: 计算 5 的阶乘

```python
result = 1
n = 1
while n <= 5:
    result *= n
    n += 1
print(result)
```

代码示例: 求 1! + 2! + 3! + 4! + 5!

```
num = 1
sum = 0
while num <= 5:
    factorResult = 1
    i = 1
    while i <= num:
        factorResult *= i
        i += 1
    sum += factorResult
    num += 1
print(sum)
```

这个程序用到了两重循环.

也就是在循环语句中也可以套循环.

for 循环

基本语法格式

```
for 循环变量 in 可迭代对象:
    循环体
```

注意:

- python 的 for 和其他语言不同, 没有 "初始化语句", "循环条件判定语句", "循环变量更新语句", 而是更加简单
- 所谓的 "可迭代对象", 指的是 "内部包含多个元素, 能一个一个把元素取出来的特殊变量"

代码示例: 打印 1-10

```
for i in range(1, 11):
    print(i)
```

- 使用 range 函数, 能够生成一个可迭代对象. 生成的范围是 [1, 11), 也就是 [1, 10]

代码示例: 打印 2, 4, 6, 8, 10

```
for i in range(2, 12, 2):
    print(i)
```

- 通过 range 的第三个参数, 可以指定迭代时候的 "步长". 也就是一次让循环变量加几.

代码示例: 打印 10-1

```
for i in range(10, 0, -1):
    print(i)
```

- range 的 步长 也可以设定成负数.

代码示例: 求 1 - 100 的和

```
sum = 0
for i in range(1, 101):
    sum += i
print(sum)
```

continue

continue 表示结束这次循环, 进入下次循环.

代码示例: 模拟吃包子. 吃第 3 个包子的时候吃出了一只虫.

```
for i in range(1, 6):
    if i == 3:
        continue
    print(f"吃完第 {i} 个包子")
```

break

break 表示结束整个循环

代码示例: 模拟吃包子. 吃第 3 个包子的时候吃出了半只虫.

```
for i in range(1, 6):
    if i == 3:
        break
    print(f"吃完第 {i} 个包子")
```

代码示例: 输入若干个数字, 求平均值. 使用 "分号" 作为结尾.

```
sum = 0
count = 0
while True:
    num = input("请输入数字:")
    if num == ';':
        break
    num = float(num)
    sum += num
    count += 1
print(sum / count)
```

综合案例

实现 "人生重开模拟器"

> 这是一款之前很火的文字类小游戏. 玩家输入角色的初始属性之后, 就可以开启不同的人生经历.
>
> 大家可以在网上搜索 "人生重开模拟器", 就可以玩到这个游戏的各种版本.

完整的程序代码较多, 此处我们**只实现其中的一部分逻辑**.

1. 设置初始属性

在游戏中我们设定四个属性.

- 颜值 (face)
- 体质 (strong)
- 智力 (iq)
- 家境 (home)

我们约定每个属性的范围为 [1, 10], 并且总和不能超过 20.

> 如果玩家输入的初始属性不合理, 就提示输入有误, 重新输入.

```
print("+-------------------------------------------------------------------------+")
print("|                                                                         |")
print("|                        花有重开日，人无再少年                           |")
print("|                                                                         |")
print("|                        欢迎来到，人生重开模拟器                         |")
print("|                                                                         |")
print("+-------------------------------------------------------------------------+")

# 设置初始属性
while True:
    print("请设定初始属性(可用总点数 20)")
    face = int(input("设定 颜值(1-10):"))
    strong = int(input("设定 体质(1-10):"))
    iq = int(input("设定 智力(1-10):"))
    home = int(input("设定 家境(1-10):"))

    if face < 1 or face > 10:
        print("颜值设置有误!")
        continue
    if strong < 1 or strong > 10:
        print("体质设置有误!")
        continue
    if iq < 1 or iq > 10:
        print("智力设置有误!")
        continue
    if home < 1 or home > 10:
        print("家境设置有误!")
        continue
```

```
    if face + strong + iq + home > 20:
        print("总点数超过了 20!")
        continue
    print("初始属性设定完成!")
    break
```

2. 设置性别

通过 `random.randint(1, 6)` 生成一个 [1, 6] 的随机整数, 类似于掷色子.

- 如果是单数, 则性别设为男孩
- 如果是双数, 则性别设为女孩.

男孩和女孩会遇到不同的事件.

```
point = random.randint(1, 6) # 掷色子
if point % 2 == 1:
    gender = 'boy'
    print("你是个男孩")
else:
    gender = 'girl'
    print("你是个女孩")
```

3. 设置出生点

首先按照家境(home), 分成四个档位.

- 10 是第一档. 加成最高
- [7, 9] 是第二档. 也有一些加成
- [4, 6] 是第三档. 加成较少
- [1, 3] 是第四档. 会扣掉属性.

再扔一次色子, 生成 [1, 3] 的随机数, 用来表示每一种细分情况.

> 这里的代码主要就是各种 if else 构成.

```

point = random.randint(1, 3) # 掷色子
if home == 10:
    print('你出生在帝都, 你的父母是高官政要')
    home += 1
    iq += 1
    face += 1
elif 7 <= home <= 9:
    if point == 1:
        print('你出生在大城市, 你的父母是公务员')
        face += 2
    elif point == 2:
        print('你出生在大城市, 你的父母是大企业高管')
        home += 2
    else:
```

```
        print('你出生在大城市，你的父母是大学教授')
        iq += 2
elif 4 <= home <= 6:
    if point == 1:
        print('你出生在三线城市，你的父母是教师')
        iq += 1
    elif point == 2:
        print('你出生在镇上，你的父母是医生')
        strong += 1
    else:
        print("你出生在镇上，你的父母是个体户")
        home += 1
else:
    if 1 <= point <= 2:
        print('你出生在村里，你的父母是辛苦劳作的农民')
        strong += 1
        face -= 2
    elif 3 <= point <= 4:
        print('你出生在穷乡僻壤，你的父母是无业游民')
        home -= 1
    else:
        print('你出生在镇上，你父母感情不和')
        strong -= 1
```

4. 针对每一岁，生成人生经历

按照年龄，把人生经历分成四个阶段：

- 幼年阶段 [1, 10]
- 青年阶段 [11, 20]
- 壮年阶段 [20, 50]
- 老年阶段 50 岁以上.

每个阶段都会有不同的精力和事件发生.

- 幼年阶段可塑性强，体质，颜值，智力都会有较快变化
- 青年阶段主要是求学，同时父母一辈会有明显变化，智力和家境会有明显变化.
- 壮年阶段相对平稳，属性变化不大，主要是一些随机事件影响到属性.
- 老年阶段体质，颜值，智力都会显著退化，并且随着年龄的上升，疾病/死亡的风险逐渐升高.

此处我们以幼年为例，简单实现一下这里的处理逻辑.

- 使用 for 循环，按照年龄循环起来.
- 针对每一年，先掷一次 [1, 3] 的色子，根据不同的随机数值，来触发不同的事件.
- 根据性别，年龄，各种属性，来触发不同的事件. 这里都使用 `if - else` 的方式来组织.
- 不同的事件可能会对属性有正面/负面的影响.
- 在每一年的最后，打印这一年遇到的事情.
- 如果夭折，则直接 `sys.exit(0)` 退出程序.
- 使用 `time.sleep(1)` 使程序暂停执行 1s，方便观察程序结果.

```
for age in range(1, 11):
    info = f'你今年 {age} 岁, '
    point = random.randint(1, 3)
    # 性别触发事件
    if gender == 'girl' and home <= 3 and point == 1:
        info += '你家里人重男轻女思想非常严重, 你被遗弃了!'
        print(info)
        print("游戏结束!")
        sys.exit(0)
    # 体质触发的事件
    elif strong < 6 and point != 3:
        info += '你生了一场病, '
        if home >= 5:
            info += '在父母的精心照料下恢复了健康'
            strong += 1
            home -= 1
        else:
            info += '你的父母没精力管你, 你的身体状况更糟糕了'
            strong -= 1
    # 颜值触发的事件
    elif face < 4 and age >= 7:
        info += '你因为长的太丑, 别的小朋友不喜欢你, '
        if iq > 5:
            info += '你决定用学习填充自己'
            iq += 1
        else:
            if gender == 'boy':
                info += '你和别的小朋友经常打架'
                iq -= 1
                strong += 1
            else:
                info += '你经常被别的小朋友欺负'
                strong -= 1
    # 智商触发的事件
    elif iq < 5:
        info += '你看起来傻傻的, '
        if home >= 8 and age >= 6:
            info += '你的父母给你送到更好的学校学习'
        elif 4 <= home <= 7:
            if gender == 'boy':
                info += '你的父母鼓励你多运动, 加强身体素质'
                strong += 1
            else:
                info += '你的父母鼓励你多打扮自己'
                face += 1
        else:
            info += '你的父母为此经常吵架'
            if point == 1:
                strong -= 1
            elif point == 2:
                iq -= 1
    # 健康成长
    else:
        info += '你健康成长, '
        if point == 1:
```

```
            info += '看起来更聪明了'
            iq += 1
        elif point == 2:
            info += '看起来更好看了'
            face += 1
        else:
            info += '看起来更结实了'
            strong += 1

    print('---------------------------------------------')
    print(info)
    print(f'strong={strong}, face={face}, iq={iq}, home={home}')
    time.sleep(1)
```

更多的逻辑, 此处就不再实现了. 大家可以按照类似的方式, 设计更多的事件, 完成 青年, 壮年, 老年 的相关逻辑.