模式匹配

廖雪峰
资深软件开发工程师，业余马拉松选手。

当我们用 `if ... elif ... elif ... else ...` 判断时，会写很长一串代码，可读性较差。

如果要针对某个变量匹配若干种情况，可以使用 `match` 语句。

例如，某个学生的成绩只能是 `A` 、 `B` 、 `C` ，用 `if` 语句编写如下：

```
score = 'B'
if score == 'A':
    print('score is A.')
elif score == 'B':
    print('score is B.')
elif score == 'C':
    print('score is C.')
else:
    print('invalid score.')
```

如果用 `match` 语句改写，则改写如下：

```
score = 'B'

match score:
    case 'A':
        print('score is A.')
    case 'B':
        print('score is B.')
    case 'C':
        print('score is C.')
    case _: # _表示匹配到其他任何情况
        print('score is ???.')
```

使用 `match` 语句时，我们依次用 `case xxx` 匹配，并且可以在最后（且仅能在最后）加一个 `case _` 表示"任意值"，代码较 `if ... elif ... else ...` 更易读。

复杂匹配

`match` 语句除了可以匹配简单的单个值外，还可以匹配多个值、匹配一定范围，并且把匹配后的值绑定到变量：

```
age = 15

match age:
    case x if x < 10:
        print(f'< 10 years old: {x}')
    case 10:
        print('10 years old.')
    case 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18:
        print('11~18 years old.')
    case 19:
        print('19 years old.')
    case _:
        print('not sure.')
```

在上面这个示例中，第一个 `case x if x < 10` 表示当 `age < 10` 成立时匹配，且赋值给变量 `x` ，第二个 `case 10` 仅匹配单个值，第三个 `case 11|12|...|18` 能匹配多个值，用 `|` 分隔。

可见， `match` 语句的 `case` 匹配非常灵活。

匹配列表

`match` 语句还可以匹配列表，功能非常强大。

我们假设用户输入了一个命令，用 `args = ['gcc', 'hello.c']` 存储，下面的代码演示了如何用 `match` 匹配来解析这个列表：

```
args = ['gcc', 'hello.c', 'world.c']
# args = ['clean']
# args = ['gcc']

match args:
    # 如果仅出现gcc，报错：
    case ['gcc']:
        print('gcc: missing source file(s).')
    # 出现gcc，且至少指定了一个文件：
    case ['gcc', file1, *files]:
        print('gcc compile: ' + file1 + ', ' + ', '.join(files))
    # 仅出现clean：
    case ['clean']:
        print('clean')
    case _:
        print('invalid command.')
```

第一个 `case ['gcc']` 表示列表仅有 `'gcc'` 一个字符串，没有指定文件名，报错；

第二个 `case ['gcc', file1, *files]` 表示列表第一个字符串是 `'gcc'` ，第二个字符串绑定到变量 `file1` ，后面的任意个字符串绑定到 `*files` （符号 `*` 的作用将在函数的参数中讲解），它实际上表示至少指定一个文件；

第三个 `case ['clean']` 表示列表仅有 `'clean'` 一个字符串；

最后一个 `case _` 表示其他所有情况。

可见， `match` 语句的匹配规则非常灵活，可以写出非常简洁的代码。

参考源码

do_match.py

« 条件判断

循环 »

Comments

Comments loaded. To post a comment, please Sign In

时光过得好快 @ 2025/12/26 03:07:22

h,w = float(input('身高：')),float(input('体重：'))

bmi = w / （h**2)

match bmi:

```
case x if x <18.5: s = '过轻'

case x if x <25:   s = '正常'

case x if x <28:   s = '过重'

case x if x <32:   s = '肥胖'
```

PYTHON教程

```
case _:            s = '过度肥胖'
```

print(f'你的bmi是: {bmi:.2f},体重情况:{s}')

Thor. @ 2025/6/4 10:56:57

对于我这种纯小白0基础的朋友，经过AI的深度讨论，我将gcc（C语言相关）替换为python语言相关，然后以下是我的理解：

首先我们要知道，上述代码只是模拟一个简单的命令行工具，模拟命令行工具的参数解析和处理，根据输入的参数（比如python 、文件名、clean 等），输出不同的提示信息。上述代码可以作为学习命令行程序、参数解析、模式匹配等知识的入门模板。此代码并不会实际执行任何操作，如需进行操作需要进行扩展代码（添加执行代码）。已拓展的代码可以执行批量运行脚本或者清理环境等操作。

```
args = ['python', 'hello.py', 'world.py'] #把一个列表赋值给变量args，列表内容模拟cmd（power

args = ['clean'] #此两行为注释，提示可以输入的命令，同样没有拓展代码，实际也不会运行，仅做参考
args = ['python']

match args:
     case ['python']: #仅输入一个命令python，但未指定需要python运行的脚本文件，所以会提示缺少
```

Read More ▼

🥦📞🔴 @ 2025/7/1 00:44:15

优秀

林示 @ 2025/7/15 04:18:36

说的很专业，这是小白呜呜呜

... @ 2025/7/22 11:12:10

看你写的才明白了个大概，非常感谢

久恙～～ @ 2025/8/8 06:24:20

感谢大佬，没有你的解释又混过去了

---

sta @ 2025/8/17 23:45:48

感谢大佬解释

Magnolia @ 2025/9/3 11:37:04

感谢信息共享

▵→奥霍斯德尔萨拉多 @ 2025/12/19 01:44:22

感谢大佬的信息分享，差点稀里糊涂蒙混过去

Mooooses @ 2025/12/24 04:13:03

print里面的 ‘ , ’是什么?为什么要这样

▵→奥霍斯德尔萨拉多 @ 2025/12/18 06:33:44

```
#match模式匹配写法
h=1.75
w=80.5
bmi=w/h**2
```

```
match bmi:
    case bmi if bmi>32:
        print('严重肥胖')
    case bmi if bmi>28:
        print('肥胖')
    case bmi if bmi>25:
        print('过重')
    case bmi if bmi>18.5:
        print('正常')
    case _:
        print('过轻')
```

中国梧桐 @ 2025/12/19 03:12:18

args = ['gcc', 'hello.c', 'world.c']

args = ['clean']

args = ['gcc']

match args: # 濡傛灉浠呮寚瀹氫簡gcc锛屾姤閿�: case ['gcc']: print('gcc: missing source file(s).') # 鍑虹幇gcc锛屼笖鑷冲皯鎸囧畾浜嗕竴涓枃浠�: case ['gcc', file1, *files]: print('gcc compile: ' + file1 + ', ' + ', '.join(files)) # 浠呮寚瀹氫簡lean: case ['clean']: print('clean') case _: print('invalid command.')

中国梧桐 @ 2025/12/19 03:13:23

args = ['gcc', 'hello.c', 'world.c']

args = ['clean']

args = ['gcc']

match args:

```
case ['gcc']:
    print('gcc: missing source file(s).')

case ['gcc', file1, *files]:
    print('gcc compile: ' + file1 + ', ' + ', '.join(files))

case ['clean']:
    print('clean')
case _:
    print('invalid command.')
```

轨 迹 ^0^ @ 2025/12/10 22:35:54

```
height = 1.75
weight = 80.5

bmi = weight / height**2

match bmi:
    case bmi if bmi >= 32:
        print("严重肥胖")
    case bmi if bmi >= 28:
        print("肥胖")
    case bmi if bmi >= 25:
        print("过重")
    case bmi if bmi >= 18.5:
        print("正常")
    case _:
        print("过轻")
```

北方烟火🎇 @ 2025/12/10 02:12:03

上节课的作业拿这个match命令来做一下 下面是代码：

```
h=input('请输入身高（米）：')
h=float(h)
w=input('请输入体重（KG）：')
w=float(w)
bmi=w/(h*h)
match bmi:
    case bmi if bmi<=18.5:
        print(f'您的BMI={bmi:.2f}','结果：过轻')
    case bmi if bmi>18.5 and bmi<=25:
        print(f'您的BMI={bmi:.2f}','结果：正常')
    case bmi if bmi>25 and bmi<=28:
        print(f'您的BMI={bmi:.2f}','结果：过重')
    case bmi if bmi>28 and bmi<=32:
        print(f'您的BMI={bmi:.2f}','结果：肥胖')
```

Read More ▼

失眠的树 @ 2025/8/30 03:35:46

```
age = int(input('input age:'))
match age:
    case x if x < 18:#age < 18成立时匹配，且赋值给变量x
        print(f'<18, age is {x}')

这里的关键在于 x 在 case 语句中是一个捕获变量。
它的工作流程是这样的：
match age: 将 age 的值（例如 15）作为匹配的主体。
case x: 这是一个捕获模式。它的规则是：无条件匹配成功，并且立即将主体的值（即 age 的值 15）赋值
if x < 18: 这是守护项 (Guard)。只有在模式匹配成功后，才会执行守护项中的条件判断。此时，变量 x
因模式匹配成功且守护项为 True，所以执行这个 case 块下的代码。
简单来说，x 是在匹配过程中被创建和赋值的，它的值就是你要匹配的那个 age。
```

HF @ 2025/11/26 23:26:24

那在后面的match 10 中是将age的值与10做一个比较吗？

Meow @ 2025/11/12 02:03:00

a,b,c

a, b, c

','.join(files)

', '.join(files)

Y² @ 2025/11/4 02:57:34

match age: case x if x < 10: print(f'< 10 years old: {x}') 如何完成age赋值给x的？f是什么意思？浮点数？删了好像也没影响

lzh @ 2025/11/5 02:38:34

可以回看下5.2 f-string，是用来格式化字符串的

Y² @ 2025/11/6 03:44:59

楼上说的正确，今天看到6.1后知后觉了

炬火 @ 2025/11/5 06:28:07

打卡第四天

JOhN @ 2025/8/12 15:12:24

```
在 ','.join(files) 中，逗号是字符串连接符，用于指定列表中元素之间的分隔符。
具体来说：
join() 是字符串的方法，作用是将一个可迭代对象（这里是 files 列表）中的所有元素连接成一个新字符
调用方法的字符串（这里是 ','）会作为分隔符，插入到列表元素之间。
例如，当 files = ['world.c'] 时：
','.join(files) 的结果是 'world.c'（只有一个元素，不需要分隔符）
如果 files = ['a.c', 'b.c', 'c.c']：
','.join(files) 的结果是 'a.c,b.c,c.c'（元素之间用逗号分隔）
简单说，这里的逗号就是最终生成的字符串中，各个元素之间的 “间隔符号”。如果把 ',' 换成 ';'，结果
```

。。 @ 2025/9/12 09:55:32

excellent

大汪汪 @ 2025/9/26 09:35:41

thanks!

學不懂Fourier @ 2025/10/1 04:59:53

thanks

柯柯&神奇眼镜熊 @ 2025/10/10 04:12:08

感谢!

俞瑾迦 @ 2025/11/3 19:54:05

感谢

我的八见奈😜😜😜 @ 2025/11/4 03:49:07

谢谢老哥，看懂了

，.?！。 @ 2025/10/29 02:31:49

age = int(input('输入你的数字：'))

match age: case x if x < 7: #age < 18成立时匹配，且赋值给变量x #如果接收到“某值”，当“某值”<10时，就执行接下来的操作 print(f'小于七岁：{x}，不可游玩') case 8|9|10 : print(f'8-10岁，三折') case 11|12|13|14|15|16|17: #或case x if 11 <= x <=17 print(f"11-17岁未成年，半价") case x if 18 <= x <= 60: print(f"已成年，全价票") case _: print("年龄过大不建议购票")

爱爱爱爱爱爱爱 @ 2025/10/31 05:40:31

实际上这个就是其他语言里面的swchi语句 为什么要取这么一个名字呢

2333 @ 2025/9/25 09:54:09

顺带一提match可以嵌套进条件判断的else:后，而且match后还能嵌套if条件判断

```
a=int(input("Enter a number"))
if a>=100:
    print("a大于100")
else:
```

```
    match a:
        case 23:
            print(23)
        case x if 0<x<23:#这里的属于match结构的一部分
            if x<11:
                print("0<x<11")
            else:
                print("11<x<23")
        case _:
            print("nothing")
```

不知道有什么办法可以让elif也嵌套个match，那样一定很coooooool

大汪汪 @ 2025/9/26 07:24:46

你太牛了，我觉得你都不需要来学了

可燃乌龙茶 @ 2025/10/22 03:03:23

本身用match就是为了简单明了，如果套if，再套match就有点本末倒置了。

拼刀接下劈 @ 2025/10/19 11:54:45

我的卡组=["黑魔术少女","黑魔术师","羽翼栗子球","死者苏生"] match 我的卡组 : case ["黑魔术少女"] : print("第一个") case ["黑魔术少女",file1,*files]: print(file1+","+",".join(files)) case _: print("其他情况")

土土🐰 @ 2025/9/24 07:56:47

第2天打卡学习记录 因python低于3.10版本，所以选择用if改写match

```
# match匹配多个值并能把匹配值绑定到变量
# 因低于3.10版本不能使用match，以下#为if-elif-else用法
age = 15
match age:
    case x if x < 10:
#   if age < 10:
        print(f'< 10 years old: {x}')
#       print(f'< 10 years old: {age}')
    case 10:
#   elif age == 10:
        print('10 years old.')
#       print('10 years old.')
    case 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18:
#   elif age in (11,12,13,14,15,16,17,18):
```

Read More ▼

大汪汪 @ 2025/10/19 05:17:04

满分!

电子死宅爹 @ 2025/8/28 21:48:18

```
厨房=['厨师','西红柿','鸡蛋','盐']
match 厨房:
    case['厨师']:
        print('厨师:缺少食材')
    case['厨师',主料,*辅料]:
        print(f'厨师用{主料}和{','.join(辅料)}做菜')
    case['清洁工']:
        print('打扫厨房卫生')
    case _:
        print('无效指令')
```

大汪汪 @ 2025/9/26 23:25:17

我有个问题请教一下：如果西红柿和鸡蛋都算作主料，怎样把'主料'指向西红柿和鸡蛋这两样东西呢?

@残梦one @ 2025/9/28 10:12:43

很棒的问题,问了ai,需要在匹配栏中增添一个匹配项case['厨师',主料1,主料2,*辅料],在匹配结果中用新变量接收这两个数据,主料=(主料1,主料2),打印结果print(f'厨师用{主料}和{','.join(辅料)}做菜'),不知道有没有其他方法

多量子阱 @ 2025/10/10 08:46:15

@大汪汪

厨房=['厨师','西红柿','鸡蛋','盐'] match 厨房: case['厨师']: print('厨师:缺少食材') case['厨师',*主料,辅料]: print(f"厨师用{','.join(主料)}和{辅料}做菜") case['清洁工']: print('打扫厨房卫生') case _: print('无效指令')

这样主料就是西红柿和鸡蛋了

大汪汪 @ 2025/10/19 03:08:57

谢谢 @残梦one 和 多量子阱，你们的方案确实都可行！多量子阱 的用"*主料"一个变量就绑定了n个主料，比 @残梦one 的更方便。 那么我想到第二个问题 :-)：如果主料和辅料都不是连续出现的怎么办？-->厨房=['厨师','西红柿','葱','鸡蛋','盐'] 只有西红柿和鸡蛋是主料，这下怎么办 是按照 @残梦one 的分别定义：case['厨师',主料1,辅料1,主料2,*辅料] 还是得在开头硬性规定主料和辅料必须分别连续：厨房=['厨师','西红柿','鸡蛋','葱','盐']？ 谢谢了！

哈力 @ 2025/9/19 02:27:02

继续学习中。 调查一下，大伙学这个是什么原因？是有明确的目标吗？

明日 @ 2025/9/20 22:24:36

为了加薪

土土🐰 @ 2025/9/24 22:13:34

为了多一门技术成为第二职业

爱 贝里屋 @ 2025/9/25 03:29:20

好玩

2333 @ 2025/9/25 09:26:14

好玩

狼图腾-崛起 @ 2025/9/28 06:53:31

为了当前项目能更完善，一开始是用AI写一些项目 然后发现 不懂编程 行不通 浪费了大多数时间

2333 @ 2025/9/25 09:27:17

查了下，.join命令是让括号内的列表去掉......算了和我一样第一时间没懂的你看一下结果就懂了

```
args = ['gcc', 'hello.c', 'world.c']
print("".join(args))
```

结果：gcchello.cworld.c

```
args = ['gcc', 'hello.c', 'world.c']
print(args)
```

结果：['gcc', 'hello.c', 'world.c']，##所以大伙输出列表的时候记得使用.join，直接输出可能得不到你预想中的结果

大汪汪 @ 2025/9/26 07:27:36

没看懂。你第二个例子就是要输出列表的，但没用.join，不需要输出列表反而用了.join（第一个例子），你是不是说反了？

土土🐰 @ 2025/9/24 12:34:55

第2天打卡学习记录

```
# match匹配列表解析

args = ['gcc', 'hello.c', 'world.c'] #包含1个字符串和2个文件
#  args = ['clean']  #此列表中只有clean
#  args = ['gcc']    #此列表中只有gcc
match args:
    case ['gcc']: #当列表中只有gcc
        print('gcc: missing source file(s).')
       #只有gcc时，打印：缺少源文件
    case ['gcc',file1, *files]:
   #列表中有3个元素：
   # 1.gcc (字符串)
   # 2.file1 (此变量指向'hello.c')
   # 3.*files (指的是把剩下所有文件打包成一个,指向'world.c')
```

Read More ▼

蘅芜君 @ 2025/7/31 07:38:49

age = 15

match age: case x if x < 10: print(f'< 10 years old: {x}') case 10: print('10 years old.') case 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18: print('11~18 years old.') case 19: print('19 years old.') case _: print('not sure.')

这段代码第三行没看懂，为啥有两个x?

Δ @ 2025/8/1 10:58:32

先case x 匹配一个变量，再用if给这个变量限制条件

JOhN @ 2025/8/12 15:03:03

```
case x if x < 10:
如果接收到“某值”，当“某值”<10时，就执行接下来的操作
```

🌲 @ 2025/8/7 00:55:45

day4