

第三周平时作业

朱士杭 231300027

2024 年 3 月 14 日

1 python 中 if-elif-else 语句工作原理

条件分支语，先判断 if 语句条件表达式是否成立是 True 还是 False，并且满足短路求值，即一旦前面部分的表达式已经决定了整个条件表达式的 True 或者 False 的时候就不再继续求表达式的值了；如果 if 满足则执行 if 下面的语句，执行完以后跳出，不再执行 elif 与 else；如果 if 不满足则执行 elif 下面的语句，执行完以后跳出，不再执行 else 语句；如果 if 与 elif 条件表达式都不满足，则执行 else 下的语句

2 while 循环

while 循环主要用于判断事件的循环

这里需要一个计数器 i=2 每执行完一次 while 循环计数器都需要 +2

```
29 sum=0
30 i=2
31 while i<=100:
32     sum+=i
33     i+=2
34 print("1到100以内所有偶数的和为:",sum)
```

图 1: 2 到 100 以内所有偶数之和

3 迭代器与生成器

3.1 迭代器

迭代器主要是对于一个可迭代对象进行遍历的一个工具，相当于一个指针指向这个可迭代对象，使用 `iter()` 函数获取该迭代器，通过 `next()` 函数获取迭代器下一个所指向的元素，并且将迭代器自动向下一个元素移动，直到移动到可迭代对象最后一个元素为止。

迭代器使用过一次就相当于失效了，需要生成一个新的迭代器

下图中由于已经遍历完整个迭代器对象，迭代器已经失效，再使用 `next()` 函数则会报错

```
62 list1=[2,4,5,9]
63 x=iter(list1)
64 print("迭代器为:",x)
65 print("迭代器指向第一个元素为:",next(x))
66 for i in x:
67     print(i,end=" ")
68 print("")
69 print("迭代器x指向下一个元素:",next(x))
```

运行 test

```
迭代器为: <list_iterator object at 0x0000028ACBCF4160>
迭代器指向第一个元素为: 2
4 5 9
Traceback (most recent call last):
  File "E:\大学课程\大一下\人工智能程序设计\作业集\第三次作业", line 69, in <module>
    print("迭代器x指向下一个元素:",next(x))
StopIteration
```

图 2: 迭代器

3.2 生成器

生成器相当于一个更加优雅的迭代器，在函数中通过与 `yield` 关键词语句搭配可以直接 `return` 内容的迭代器，当需要迭代器里面的元素的时候直接从 `yield` 下一行语句开始执行；而在列表解析的时候可以将 `[]` 替换为 `()` 从而获取该生成器，用 `next()` 函数的时候与普通迭代器用法一致

```
71 list1=[2,4,5,9]
72 x=(i*2 for i in list1)
73 print("生成器为:",x)
74 print("生成器指向第一个元素为:",next(x))
75 for i in x:
76     print(i,end=" ")
77 print("")
78 print("生成器x指向下一个元素:",next(x))
```

运行 test

Traceback (most recent call last):
File "E:\大学课程\太一下\人工智能程序设计\作业集", line 78, in <module>
 print("生成器x指向下一个元素:",next(x))
 ^^^^^^^
StopIteration

生成器为: <generator object <genexpr> at 0x0000000000000000>
生成器指向第一个元素为: 4
8 16 18

图 3: 用 `()` 列表解析产生生成器

```
1 个用法
82 def generator():
83     list1=[2,4,5,9]
84     for i in list1:
85         yield i**2
86     print("当对生成器进行迭代的时候直接从yield下一行语句开始执行")
87     print("返回生成器当中多加一个元素:",i**2)
88 x=generator()
89 for i in x:
90     print(i,end="\t")
```

运行 test

4 16 25 81
当对生成器进行迭代的时候直接从yield下一行语句开始执行
返回生成器当中多加一个元素: 4
16 25 81
当对生成器进行迭代的时候直接从yield下一行语句开始执行
返回生成器当中多加一个元素: 16
25 81
当对生成器进行迭代的时候直接从yield下一行语句开始执行
返回生成器当中多加一个元素: 25
81

图 4: 与 `yield` 关键字结合产生生成器

4 分析代码输出及其原因

代码 1: 输出 [1,4,9] 这里使用了列表解析的方式生成了一个列表

代码 2: 输出 [1,4,9] 这里用 () 生成了一个生成器对象 squares, 然后将该生成器进行强制类型转换成 list 列表。假如直接 print(squares) 则会输出 <generatorobject < genexpr > at0x0000025737DC0BA0 > 表明这是一个生成器对象

5 条件表达式

返回一个 bool 值 True 或 False 类型的函数, 相当于 C++ 里面的: 表达式? 值 1: 值 2 满足表达式返回值 1, 不满足返回值 2



```
93 num=int(input("请输入一个整数判断其正负性: "))
94 if num >0:
95     print("Positive")
96 elif num==0:
97     print("Zero")
98 else:
99     print("Negative")
```

运行 test x

E:\PythonANACONDA\python.exe E:\大学课程\大一下\人工智能程序设计\作业集\第三次作业

请输入一个整数判断其正负性: 0

Zero

图 5: 条件表达式判断正负

6 文件读写

```
35 import os
36 str_input=input("请输入要输入文件等等字符串:")
37 with open("file_input.txt","w+") as f:
38     f.write(str_input)
39 print("接下来进行读取并打印操作")
40 with open("file_input.txt","r") as f:
41     contents=f.readlines()
42     for line in contents:
43         print(line)
44 print("打印操作结束")
```

运行 test ×

E:\PythonANACONDA\python.exe E:\大学课程\大一下\...
请输入要输入文件等等字符串: asdfghjkl
接下来进行读取并打印操作
asdfghjkl
打印操作结束

图 6: 文件写入字符串并读取打印