

魔法方法(了解)

```
1 魔法方法:很魔幻
2 格式:__xxxx__(self)
3 xxxx有很多,如:
4 add 加法(两个对象相加时调用)
5 lt 小于(两个对象比较时调用)
6 str 字符串(打印对象时调用)
7 ....
8
9 class Maker():
10     def __add__(self,other):#self=m,other=m2
11         print(self.age)
12         print(other.kk)
13         print("当2个对象相加时,我会被调用")
14         return self.age+other.kk
15
16
17 m=Maker()
18 m.age=20
19 m2=Maker()
20 m2.kk=30
21 print(m+m2)#两个对象相加
22
23 class Maker():
24     def __lt__(self,other):#self=m,other=m2
25         print(self.age)
26         print(other.kk)
27         print("当2个对象小于比较时,我会被调用")
28         return self.age+other.kk
29
30
31 m=Maker()
32 m.age=20
33 m2=Maker()
34 m2.kk=30
35 print(m<m2)#两个对象比较
36
37 class Maker():
38     def __str__(self):
39         print("当直接打印对象时,会调用我")
40         return "aaa"
41
42
43 m=Maker()
44 print(m)
45
46 #案例:定义一个魔法方法,比较两个对象的大小
```

面向对象的三大特征介绍

- 1 1.封装,继承,多态
- 2 2.封装:把属性和方法封装在一起,并赋予权限
- 3 2.继承:子类继承父类,那么子类有父类的属性和方法
- 4 3.多态:同一操作作用于不同对象,可以有不同的解释,产生不同的执行结果
- 5

封装(重点)

```
1 1.封装的概念:把属性和方法封装在一起,并赋予权限
2 2.作用:保证内部的高内聚性和与外部的低耦合性
3 3.私有成员:
4     1.外部不能直接访问,内部可以直接访问
5     2.在属性或方法名前面加2个下划线就表示该成员是私有的
6
7 #!/usr/bin/env python
8 # -*- coding:utf-8 -*-
9 #====#====#====#====
10 #Author:
11 #CreatDate:
12 #Version:
13 #====#====#====#====
14
15 #封装是把属性和方法写到类中,并赋予权限
16
17 class Maker():
18     __age=10, #私有类属性
19
20     def getAge(self):
21         return self.__age
22
23     def setAge(self, age):
24         #判断修改的年龄是否合法
25         if age>=1 and age<=140:
26             self.__age=age
27
28 m=Maker()
29 # print(m.age)#当类属性为私有时,类外面不能直接操作类属性
30 # m.age=1000
31 print(m.getAge())
32 m.setAge(30)
33 print(m.getAge())
34
35
36
37
38
39 #案例:设计一个类,有私有对象属性name, age, 设计类方法,有r权限就可以读取这两个私有属性,
40 有w权限就可以修改这两个属性
```

继承介绍(重点)

- 1 1. 类与类之间才能继承
- 2 2. 作用: 简化代码, 代码复用
- 3 3. 名词解释: 父类(基类, 超类), 子类(派生类)
- 4 4. 子类的小括号中写了哪个类, 就表示继承了谁(认准做爹了)
- 5 5. 子类拥有父类的所有属性和方法(私有的除外)
- 6 6. 继承有单继承和多继承
- 7 7. object是所有类的父类, 如果一个类没有显式指明它的父类, 则默认为object
- 8

单继承(重点)

```
1 单继承, 就是子类继承一个父类(儿子只有一个爹)
2 格式:
3 #!/usr/bin/env python
4 # -*- coding:utf-8 -*-
5 #====#====#====#====
6 #Author:
7 #CreatDate:
8 #Version:
9 #====#====#====#====
10
11 #父类
12 class Father():
13     m="1千万",
14     __mytype="小三",
15
16     def mytest01(self):
17         print("有钱")
18
19     def mytest02(self):
20         print("有房")
21
22     def mytest03(self):
23         print("有颜值")
24
25 #子类
26 class Son(Father):
27     def myfunc(self):
28         print(self.m)
29         # print(self.__mytype)#报错, 不能继承父类私有成员
30
31 s=Son()
32 s.mytest01()
33 s.mytest02()
34 s.mytest03()
35 print(s.m)
36 # print(s.__type)
```

```

37 s.myfunc()
38
39
40
41 #案例:定义一个动物类,属性有name,age.方法有会走move,会叫spake.定义一个猫和狗类,继承动物类
42 #通过猫对象调用move和spake方法,分别是age岁的name猫在走,age岁的name猫在叫,狗也一样

```

多继承

```

1 多继承,就是子类有多个父类(儿子有多个爹)
2 继承规则: Python允许多继承。调用顺序:从左到右,先深度再广度
3 不建议使用多继承,多继承会增加代码的复杂性
4 格式:
5 #!/usr/bin/env python
6 # -*- coding:utf-8 -*-
7 #====#====#====#====
8 #Author:
9 #CreateDate:
10 #Version:
11 #====#====#====#====
12 class Maker():
13     BMW="奔驰"
14
15 class Father(Maker):
16     name="Father",
17     # BMW="宝马1",
18
19     def mytest01(self):
20         print("我是Father的mytest01")
21
22
23
24 class Father2():
25     name2="Father2",
26     BMW = "宝马",
27
28     def mytest02(self):
29         print("我是Father2的mytest02")
30
31 class Son(Father,Father2):
32     pass
33
34 s=Son()
35 s.mytest01()
36 s.mytest02()
37 print(s.name)
38 print(s.name2)
39 print(s.BMW)#?这个宝马是哪个爹的?不清楚
40

```

继承构造函数和析构函数问题(重点)

```

1  1.子类不写构造函数,那么会默认调用从父类继承过来的构造函数
2  2.如果重写了__init__ 时,要继承父类的构造方法,可以在子类构造函数中使用 super 关键字或父类名
3  3.析构和构造一样
4
5  #!/usr/bin/env python
6  # -*- coding:utf-8 -*-
7  #====#====#====#====
8  #Author:
9  #CreatDate:
10 #Version:
11 #====#====#====#====
12 class Father():
13     def __init__(self,name):
14         print("我是Father的构造函数")
15
16
17 class Son(Father):
18     def __init__(self):
19         # # super(Son,self).__init__("kk")#调用从父类继承的构造函数
20         # Father.__init__(self,"kk")#调用从父类继承的构造函数
21         print("我是Son的构造函数")
22
23
24 s=Son()
25 #注意,当子类没有写自己的构造函数,那么就要调用从父类继承过来的构造函数,要注意父类的构造函数是否有参数
26
27
28
29 #案例:测试一下析构函数是怎么样的?

```

子类调用父类同名方法

```

1  1.当子类的函数和父类的函数同名时,在子类的同名函数中使用super()关键字
2  格式:
3  #!/usr/bin/env python
4  # -*- coding:utf-8 -*-
5  #====#====#====#====
6  #Author:
7  #CreatDate:
8  #Version:
9  #====#====#====#====
10
11 class Father():
12     def mytest(self):
13         print("Father")
14
15 class Son(Father):
16     def mytest(self):
17         super().mytest()#调用父类的mytest函数
18         print("Son")

```

```
19
20 s=Son()
21 #当父类和子类有同名函数时,子类对象,先调用自己的函数
22 s.mytest()
23
```

重写

```
1 1.什么叫重写,当子类定义了父类的同名函数时,就叫子类重写了父类的函数
2 2.有什么用:当从父类继承过来的方法功能不够用时,可以在子类中重写该方法
3
4 #!/usr/bin/env python
5 # -*- coding:utf-8 -*-
6 #====#====#====#====
7 #Author:
8 #CreatDate:
9 #Version:
10 #====#====#====#====
11
12 class Father():
13     def myadd(self,a,b):
14         return a+b
15
16
17 class Son(Father):
18     def myadd(self,a,b):
19         print("我是有新功能的")
20         return a+b
21
22 s=Son()
23 k=s.myadd(10,20)
24 print(k)
25
```

多态(重点)

```
1 1.多态:同一操作作用于不同对象,可以有不同的解释,产生不同的执行结果
2 2.鸭子类型:当看到一只鸟走起来像鸭子、游泳起来像鸭子、叫起来也像鸭子,那么这只鸟就可以被称为鸭子
3 3.多个类中的成员函数名字要一样
4
5 #!/usr/bin/env python
6 # -*- coding:utf-8 -*-
7 #====#====#====#====
8 #Author:
9 #CreatDate:
10 #Version:
11 #====#====#====#====
12
13 class Dog():
```

```

14     def spake(self):
15         print("狗在叫")
16
17 class Cat():
18     def spake(self):
19         print("猫在叫")
20
21 class tiger():
22     def spake(self):
23         print("老虎在叫")
24 class Maker():
25     def spake(self):
26         print("天在看,人在叫")
27 #同一个操作
28 def mytest(obj):
29     obj.spake()
30
31 # 1.多态:同一操作作用于不同对象,可以有不同的解释,产生不同的执行结果
32 d=Dog()
33 c=Cat()
34 t=tiger()
35 m=Maker()
36
37 #作用于不同对
38 mytest(d)
39 mytest(c)
40 mytest(t)
41 mytest(m)
42
43 #不动前面的代码的基础上,增加新的功能
44
45
46
47 #案例:写一个操作(函数),传入三个不同的对象,打印不同的颜色
48

```

错误和异常

- 1 错误:通常是指代码中的语法错误,一般初级程序员很常见或者说很容易犯。
- 2 异常:即便Python程序的语法是正确的,在运行它的时候,也有可能发生错误。运行期间检测到的错误被称为异常
- 3 pycharm中错误常有箭头指示出来,异常没有

异常的捕获

```

1 1.捕获异常的格式:
2 try:#有可能出现异常的代码写在try里
3     print('1111111')
4     # print(a)
5     print(1 / 0)
6     # print(1 + 'a')

```

```

7     print('2222222')
8 except Exception as e: #捕获到了异常,执行except里的代码
9     print(e)
10    print("I'm except")
11 说明:Exception常规错误的基类,把异常的基本信息存储到e中
12
13 2.异常处理-except分支else:没有异常执行else下面的代码
14
15 3.finally语句:有没有异常都要执行这下面的代码
16 #!/usr/bin/env python
17 # -*- coding:utf-8 -*-
18 #====#====#====#====
19 #Author:
20 #CreatDate:
21 #Version:
22 #====#====#====#====
23 def mytest(a,b):
24     return a/b
25
26 try:
27     mytest(10,0)
28 except Exception as e:
29     print(e)
30     # raise #抛出异常
31 else:
32     print("没有异常就执行这")
33 finally:
34     print("有没有异常都要执行这里的代码")
35
36
37
38
39
40 4.抛出异常:
41     1.raise关键字
42
43 5.异常要处理,不管是谁
44 #!/usr/bin/env python
45 # -*- coding:utf-8 -*-
46 #====#====#====#====
47 #Author:
48 #CreatDate:
49 #Version:
50 #====#====#====#====
51 def mytest(a,b):
52     return a/b
53 def myfunc():
54     try:
55         mytest(10,0)
56     except Exception as e:
57         # print(e)
58         raise #抛出异常
59     else:

```



```

60         print("没有异常就执行这")
61     finally:
62         print("有没有异常都要执行这里的代码")
63
64     #这里处理myfunc函数抛出的异常
65     try:
66         myfunc()
67     except Exception as ee:
68         print(ee)
69
70     6. 自定义异常:
71         1. 要继承Exception常规错误的基类
72
73     #!/usr/bin/env python
74     # -*- coding:utf-8 -*-
75     #====#====#====#====
76     #Author:
77     #CreatDate:
78     #Version:
79     #====#====#====#====
80
81     class ShowInputOut(Exception):
82         def __init__(self, len, flag):
83             super().__init__()
84             self.len=len
85             self.flag=flag
86
87
88
89
90     def main():
91         try:
92             s=input("请输入-->")
93             if len(s)<3:
94                 raise ShowInputOut(len(s),3)
95         except ShowInputOut as e:
96             print("你输入长度为%d,长度必须为%d"%(e.len,e.flag))
97         else:
98             print("没有发生异常")
99
100     main()

```

OS模块(重点)

```

1  import sys
2  print('===Python import mode===')
3  print ('命令行参数为:')
4  for i in sys.argv:
5      print (i)
6  print ('\n python 路径为',sys.path)

```

```

7
8
9 from sys import argv,path # 导入特定的成员
10 print('===python from import===')
11 print('path:',path) # 因为已经导入path成员,所以此处引用时不需要加sys.path
12
13 import os
14     # path = os.getcwd()#获取当前执行文件的绝对路径•
15     # path = os.path.abspath('.')#获取当前路径•
16     #     # path = os.path.abspath('..')#获取当前的父路径
17 #!/usr/bin/env python
18 # -*- coding:utf-8 -*-
19 #====#====#====#====
20 #Author:
21 #CreateDate:
22 #Version:
23 #====#====#====#====
24 import os
25 print(os.getcwd())#E:\cn202305\10day
26
27 print(os.path.abspath(".."))#E:\cn202305
28 #获取当前路径下所有文件名
29 mylist=os.listdir(os.getcwd())
30 print(mylist)
31 #修改文件名
32 os.rename(os.getcwd()+'/1.txt',os.getcwd()+'/2.txt')
33
34 案例:批量修改文件名
35 import os
36 funFlag = 1 # 1表示添加标志,2表示删除标志
37 folderName = './renameDir/'
38 #获取指定路径下所有文件的名字
39 dirList = os.listdir(folderName)
40 #遍历输出所有文件的名字
41 for name in dirList:
42     print(name)
43     if funFlag==1:
44         newName='[山哥出品]-'+name
45     elif funFlag==2:
46         num=len('[山哥出品]-')
47         newName=name[num:]
48     print(newName)
49     os.rename(folderName+name,folderName+newName)

```

;