Python

极夜酱

目录

Chapter 1 模块

1.1 模块导入

1.1.1 模块（Module）

模块是进行大型项目拆分组织的一种有效技术手段，它可以将一个庞大的代码分割成若干个小的组成单元，方便进行代码的开发与维护。利用模块的划分，在进行代码维护的时候，可以保证局部的更新不影响其它的程序的运行操作。

使用 import 关键字可以进行模块的导入，import 可以同时导入多个模块，但是从开发的角度来讲，强烈建议分开导入。

```
1 import package.module [as alias] [, ...]
```

模块导入

algorithm/search.py

```
1  def sequence_search(list, key):
2      """
3          顺序查找
4          Args:
5              list (list): 待查找数组
6              key (int): 关键字
7      """
8      for i in range(len(list)):
9          if list[i] == key:
10             return i
11     return -1
12
13 def binary_search(list, key):
14     """
15         二分查找
16         Args:
```

```
17              list (list): 待查找数组
18              key (int): 关键字
19      """
20      start = 0
21      end = len(list) - 1
22      while start <= end:
23          mid = (start + end) // 2
24          if list[mid] == key:
25              return mid
26          elif list[mid] < key:
27              start = mid + 1
28          else:
29              end = mid - 1
30      return -1
```

import_as.py

```
1  import algorithm.search as search
2
3  def main():
4      list = [40, 9, 20, 93, 7, 34, 85, 91]
5      key = 34
6      print("%d所在位置: %d" % (key, search.sequence_search(list, key)))
7
8  if __name__ == "__main__":
9      main()
```

运行结果

```
34所在位置: 5
```

在 Python 里有一个作者写的 Python 开发禅道（开发的 19 条哲学），如果想看到这个彩蛋的信息，可以在 Python 的交互模式输入 import this。

运行结果

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to
do it.
Although that way may not be obvious at first unless you're
Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

1.1.2 from-import 模块导入

使用 import 导入模块之后需要采用 module_name.func() 形式进行调用，每一次调用函数非常麻烦。from-import 导入语法可以简化调用语句。

```
from package.module import name [as alias] [, ...]
```

在实践中，from-import 不是良好的编程风格，因为如果导入的变量与作用域中现有变量同名，那么变量就会被悄悄覆盖掉。使用 import 语句的时候就不会发生这种问题，通过 module.var 或 module.func() 获取的变量或方法不会与现有作用域冲突。

from-import 导入

algorithm/search.py

```
def sequence_search(list, key):
    """
        顺序查找
        Args:
            list (list): 待查找数组
            key (int): 关键字
    """
    for i in range(len(list)):
        if list[i] == key:
            return i
    return -1

def binary_search(list, key):
    """
        二分查找
        Args:
            list (list): 待查找数组
            key (int): 关键字
    """
    start = 0
    end = len(list) - 1
    while start <= end:
        mid = (start + end) // 2
        if list[mid] == key:
            return mid
        elif list[mid] < key:
            start = mid + 1
```

```
28          else:
29              end = mid - 1
30      return -1
```

from_import.py

```
1  from algorithm.search import binary_search
2
3  def main():
4      list = [7, 9, 20, 34, 40, 85, 91, 93]
5      key = 34
6      print("%d所在位置: %d" % (key, binary_search(list, key)))
7
8  if __name__ == "__main__":
9      main()
```

运行结果

```
34所在位置: 3
```

1.2 math 模块

1.2.1 math 模块

现代计算机的基础学科就是数学，如果没有数学理论作为基础，计算机是无法的得到正常发展的。数学模块只提供了数学的基本计算功能，在很多的开发之中，有可能会需要使用到更加复杂的数学逻辑的时候就需要采用一些第三方模块进行数学计算了。

math 模块

```
1  import math
2
3  def main():
4      print("累加：%d" % math.fsum(range(101)))
5      print("阶乘：%d" % math.factorial(10))
6      print("乘方：%d" % math.pow(2, 10))
7      print("对数：%f" % math.log(10))
8      print("余数：%d" % math.fmod(22, 5))
9
10 if __name__ == "__main__":
11     main()
```

运行结果

```
累加：5050
阶乘：3628800
乘方：1024
对数：2.302585
余数：2
```

1.3 random 模块

1.3.1 random 模块

随机数可以在一个指定的范围之内随机地生成一些数字供使用。例如，手机验证码发送来的数字就是使用随机数的方式生成的。

方法	功能
random()	生成一个 0 到 1 的随机浮点数：$0.0 \leq n \leq 1.0$
uniform(x, y)	生成一个在指定范围内的随机浮点数
randint(x, y)	生成一个指定范围内的随机整数 $x \leq n \leq y$
choice(sequence)	从序列中随机抽取数据
shuffle(x [, random])	将一个列表中的元素打乱
sample(sequence, k)	从指定序列中随机获取指定序列分片

表 1.1: random 模块

random 模块

```
1  import random
2
3  def main():
4      lst = [random.randint(1, 100) for _ in range(10)]
5      print("初始序列：%s" % lst)
6      print("随机抽取：", end='')
7      for _ in range(5):
8          print(random.choice(lst), end=' ')
9
10 if __name__ == "__main__":
11     main()
```

运行结果

```
初始序列：[85, 83, 83, 29, 2, 93, 30, 65, 41, 54]
随机抽取：41 93 2 54 93
```

1.4 time 模块

1.4.1 time 模块

time 模块是 Python 内置的一个实现时间的操作模块，用于描述日期时间的数据类型分为三种：