

问答题

- 1) 列出你所了解到的 10 个内置函数，请写出它的功能，输入和输出
- 2) 观看拓展资料，总结视频中涉及的知识点

解答：

(1)

1. abs()函数

```
1 abs(-1) # 返回实数的绝对值
2 abs(-1+2j) # 返回虚数的模
```

输出：

```
>>> abs(-1)
1
>>> abs(-1+2j)
2.23606797749979
```

2. complex()函数

```
1 complex(1, 2) # 返回复数。其中第一个值为实部，第二个值为虚部
```

输出结果：

```
>>> complex(1, 2)
(1+2j)
```

3. print()函数

```
1 print() # 输出函数，可以设置输出的间隔方式，输出结束时的结束方式
2 print('123' * 3) # 默认输出
3 print('123', end='\n换行! \n') # 设置结束方式
4 print('123', a, sep='%') # 设置输出间隔方式
```

运行结果：

```
123123123
123
换行!
123%1
```

4. input()函数

```
1 input() # 接受键盘输入，并返回为字符串类型。里面可以接受提示信息的输入。
2 str = input('随便输入点什么吧')
3 print(type(str))
```

运行结果：

```
>>> str = input('随便输入点什么吧')
... print(type(str))
...
随便输入点什么吧>? 233
<class 'str'>
```

5. reversed()函数

```
1 reversed() # 把列表元素逆置后，返回一个可迭代对象的地址
2 b = [1, 2, 3, 4]
3 b_re = reversed(b)
4 print(b_re)
```

运行结果：

```
>>> b = [1, 2, 3, 4]
... b_re = reversed(b)
... print(b_re)
...
<list_reverseiterator object at 0x00000217329C5508>
```

6. round()函数

```
1 PI = 3.1415726535
2 round(PI, 3)
3 # 以四舍五入的方法控制小数输出位数，其中，接受的第一个参数为浮点数对象，第二个参数为指定的小
  数位数，若不指定，默认输出整数。
```

运行结果：

```
>>> PI = 3.1415726535
>>> round(PI, 3)
3.142
```

7. sum()函数

```
1 sum(b) # 返回序列x中所有元素之和
2 sum(b, 2)
```

运行结果:

```
>>> sum(b)
10
>>> sum(b, 2)
12
```

8. max(), min()函数

```
1 # 返回括号内元素的最大值或者最小值
2 max(b)
3 min(b)
```

运行结果:

```
>>> max(b)
4
>>> min(b)
1
```

9. range()函数

```
1 range(0, 10, 1) # 返回range对象，其中包含左闭右开的区间[a, b)内，以步长为c的整数。
2 print(list(range(0, 10, 1)))
3 print(list(range(0, 10, 2)))
```

运行结果:

```
>>> range(0, 10, 1) # 返回range对象
... print(list(range(0, 10, 1)))
... print(list(range(0, 10, 2)))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8]
```

10. len()函数

```
1 # 返回对象内包含的元素个数，可以用于列表、元组、集合、字典、字符串和range对象。
2 len(b)
3 len(range(0, 10, 1))
```

运行结果:

```
>>> len(b)
4
>>> len(range(0, 10, 1))
10
```

(2)

1. python函数中，值的传递是通过指针来进行的。当变量以形参的形式传入函数中时，形参和变量都指向同一块内存空间。当函数运行完成返回时，会将返回的值所在的地址返回给主函数中的变量。
2. 函数运行完成后，函数中的局部变量会被删除。

上机题

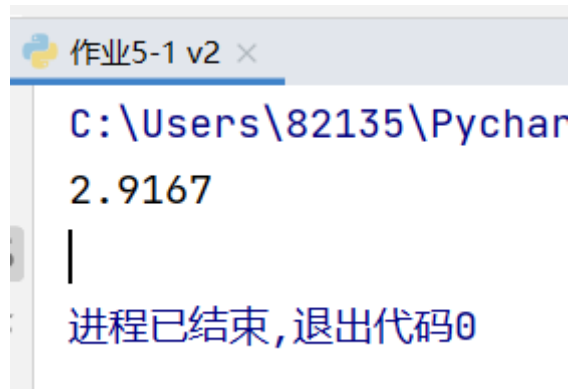
1. 课堂上机实践

源代码：

```
1  """
2  最后更新: 2022/3/29
3  作者: 林子皓
4  版本: 2.0
5  内容: 嵌套函数
6  """
7
8
9  def var(*nums): # 求方差
10     n = len(nums)
11
12     def mean(): # 求均值的平方
13
14         def sum() -> float: # 求传入的数字和
15             s = 0.0
16             for i in range(n):
17                 s += nums[i]
18             return s
19         s1 = sum()
20         m = (s1/n) * (s1/n)
21         return m
22     M = mean()
23
24     def sums():
25         s = 0.0
26         for i in range(n):
27             s += nums[i] * nums[i]
28         return s
29
30     s2 = sums()
31     S = s2/n - M
32     return S
33
```

```
34
35 a = var(1, 2, 3, 4, 5, 6)
36 a= round(a, 4)
37 print(a)
38
```

运行结果：



```
作业5-1 v2 x
C:\Users\82135\Pychar
2.9167
|
进程已结束,退出代码0
```

2. 运气游戏设计：

模拟一种骰子游戏，投掷两个六面骰子，骰子每个面上的点数分别为 1, 2, 3, 4, 5, 6。当骰子停下来时，计算两个朝上的面上的点数总和。如果第一次投掷的点数总和是 7 或 11，则游戏胜利。如果第一次投掷的点数总和为 2, 3 或 12（称为 craps），游戏失败。如果第一次投掷的点数总和是 4, 5, 6, 8, 9 或 10，那么这个总和就被称为 point。想要获胜，必须继续掷骰子直到再次投出 point。如果在得到 point 之前，出现了 7，则游戏失败。

提示：请设计两个函数来模拟该游戏

函数 roll_dice 用来模拟每次投掷两个骰子

函数 display_dice 展示骰子的求和结果。

以下是几种游戏结果展示示例，请程序最后按照此结果展示，否则扣分。

第一种：在第一次投掷时就获胜

```
Player rolled 2+5 = 7
Player wins
```

第二种：在第一次投掷时失败

```
Player rolled 6+6 = 12
Player loses
```

第三种：在后续投掷时失败

```
Player rolled 6+4 = 10
Point is 10
Player rolled 3+4 = 7
Player loses
```

第四种：在后续投掷时胜利

```
Player rolled 5+1 = 6
Point is 6
Player rolled 3+6 = 9
Player rolled 2+3 = 5
Player rolled 5+5 = 10
Player rolled 3+5 = 8
Player rolled 5+1 = 6
Player wins
```

源代码：

```
1  """
2  最后更新：2022/4/2
3  作者：林子皓
4  版本：1.0
5  内容：骰子游戏
6  """
7
8  from random import randint
9  import sys
10
11  d1 = 0
12  d2 = 0
13
14
15  def roll_dice():
16      # 随机生成两个骰子
17      dice_1 = randint(1, 6)
18      dice_2 = randint(1, 6)
19      # 赋予全局变量
20      global d1
21      global d2
22      # 记录两个骰子的值
23      d1 = dice_1
24      d2 = dice_2
25      return
26
27
28  def display_dice(dice_1, dice_2):
29      # 计算两次投骰子的和
```

```

30     total = dice_1 + dice_2
31     return total
32
33
34 roll_dice()
35 value = display_dice(d1, d2)
36 print('Player rolled ', d1, '+', d2, ' = ', value, sep='')
37
38 # win
39 if value == 7 or value == 11:
40     print('Player wins')
41     sys.exit(0)
42
43 # craps
44 elif value == 2 or value == 3 or value == 12:
45     print('Player loses')
46     sys.exit(0)
47
48 # point
49 else:
50     point = value
51     print('Point is', point)
52     while True:
53         roll_dice()
54         value = display_dice(d1, d2)
55         print('Player rolled ', d1, '+', d2, ' = ', value, sep='')
56         if value == 7:
57             print('Player loses')
58             sys.exit(0)
59         elif value == point:
60             print('Player wins')
61             sys.exit(0)

```

运行结果:

第一种情况: 在第一次投掷时就获胜

Player rolled 6+1 = 7	Player rolled 5+6 = 11
Player wins	Player wins

第二种情况: 在第一次投掷时失败

Player rolled 1+1 = 2	Player rolled 6+6 = 12
Player loses	Player loses

第三种情况: 在后续投掷时失败

Player rolled $4+6 = 10$
Point is 10
Player rolled $2+5 = 7$
Player loses

Player rolled $4+6 = 10$
Point is 10
Player rolled $6+5 = 11$
Player rolled $3+6 = 9$
Player rolled $2+2 = 4$
Player rolled $5+2 = 7$
Player loses

第四种情况：在后续投掷时胜利

Player rolled $6+3 = 9$
Point is 9
Player rolled $4+5 = 9$
Player wins
Player rolled $3+1 = 4$
Point is 4
Player rolled $6+4 = 10$
Player rolled $6+4 = 10$
Player rolled $4+5 = 9$
Player rolled $3+5 = 8$
Player rolled $4+4 = 8$
Player rolled $2+3 = 5$
Player rolled $2+2 = 4$
Player wins

