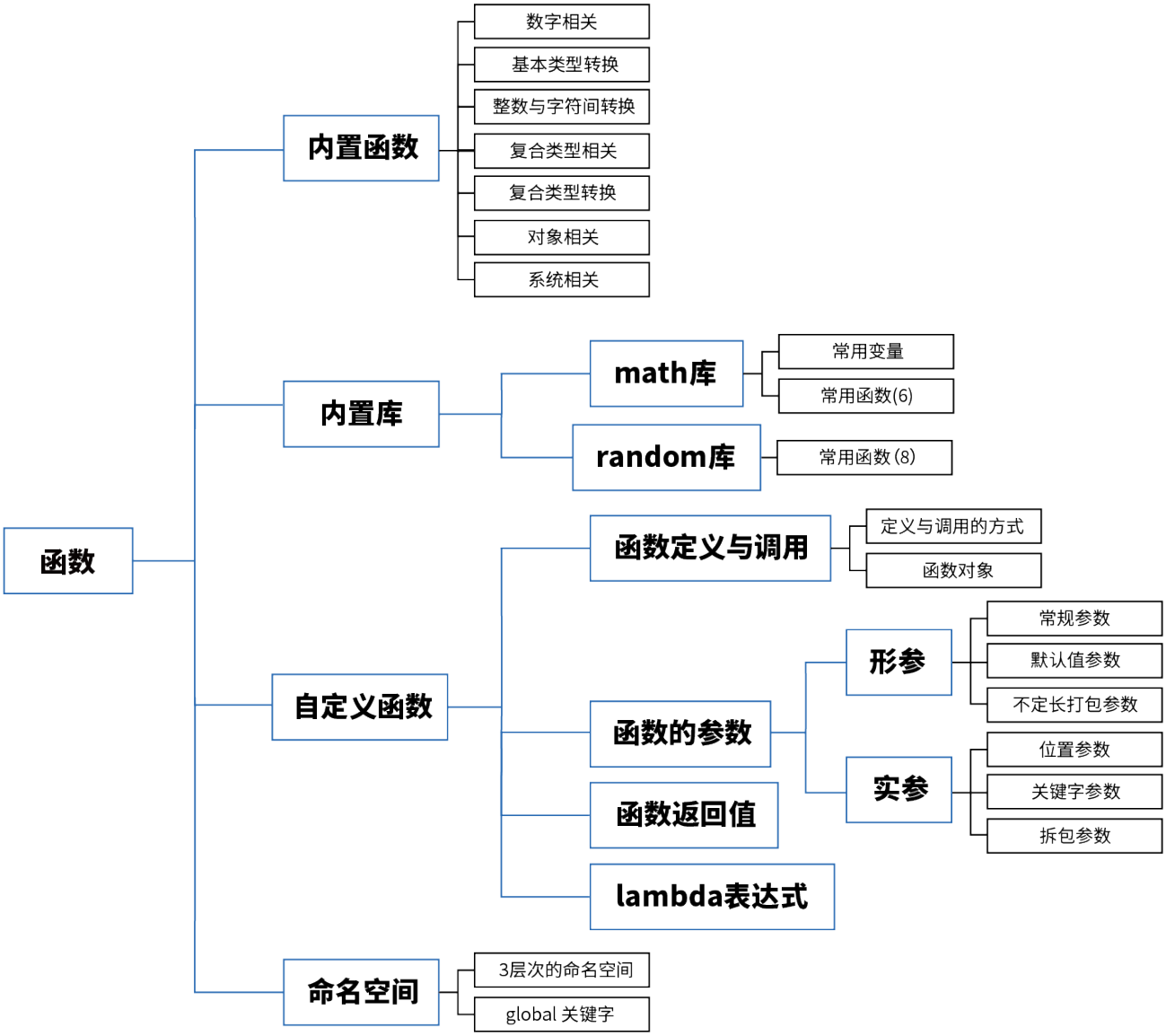
第 4 讲　函数

知识脉络



一　常用内置函数

1. 对象相关函数

① `id(obj)`　　# 显示对象 obj 在内存中的地址（赋值会改变标识符指向的对象，但不改变对象的地址）

② `type(obj)`　　# 显示对象 obj 的类型，格式为 <class '类型名称'>

2. 系统相关函数

① `eval(expression)`　　# 将字符串描述的表达式转换成 python 表达式，并计算返回值

② `exec(expression)`　　# 将字符串描述的 python 代码转换成 python 代码，并运行

· 以上两个函数中的字符串若被转换成代码后引起错误，就会直接报错

二　两个 python 内置库

1. math 库

```
import math
```

常用变量

```
math.pi     # 返回常量 3.1415926…
```

常用函数

```
math.sqrt(x)          # 返回 x 的开方（浮点数）
math.log(x[,base])    # 返回 x 以 base 为底的对数（默认为 2）
math.log10(x)         # 返回 x 以 10 为底的对数
math.pow(x, y)        # 返回 x^y
math.sin(x)           # 返回 sin x
math.cos(x)           # 返回 cos x
```

2. random 库

```
import random
```

常用函数

```
random.random()                       # 返回一个介于区间[0.0, 1.0)的随机浮点数
random.uniform(a, b)                  # 返回一个介于区间[a, b]的随机浮点数
random.randint(a, b)                  # 返回一个介于区间[a, b]的随机整数
random.randrange(start, stop, step)   # 从 range(start, stop, step)序列中随机返回一个数
random.choice(sequence)               # 随机返回序列 sequence 中的一个元素
random.shuffle(sequence)              # 返回将序列 sequence 中的元素打乱后重组的序列
random.sample(sequence, k)            # 返回序列中随机 k 个元素乱序组成的列表
random.seed(n)        # 设置随机数生成器的种子，设置之后每次调用同一随机函数的结果都是一致的
```

· 这里随机实际上是通过种子计算得到的，只要种子相同，结果就会相同

若不设置 seed，则 seed 将是生成时的系统时间，如此达成了伪随机

三　自定义函数

1.函数的定义与调用

① 定义函数

```
def 函数名(参数 1，参数 2，…):
    语句块
    return 返回值
```

· 程序在开始运行时会先处理这些函数定义，创建**函数对象**，并将函数名作为指向对象的标识符

函数对象和数字、字符串等对象一样，可以作为复合类型的元素，可以作为参数被传递

② 调用函数

```
函数名(实际参数 1，实际参数 2)
```

· 程序**跳转**到函数名指向的函数对象，执行语句块后将返回值作为整个函数调用表达式的结果值

· 注意：光有函数名并不是对函数的调用，**要调用函数必须要在函数名后加括号并提供参数**

2.函数的参数

① 函数调用时的四种实参形式

· **位置参数**

```
func(r1, r2)       # 调用时，这些实参会按照位置顺序分别复制给函数的参数
```

· **关键字参数**

```
func(arg1=r1, arg2=r2)   # 紫色是定义时的参数名，调用时可无视顺序传递给等号前面的参数
```

· **拆包序列**

```
func(*r1, *r2)     # 此时 r1 和 r2 为序列，内部的元素被拆出来变成多个位置参数
```

· **拆包键值对**

```
func(**d1, **d2)   # 此时 r1 和 r2 为字典，内部的键值对被拆出来变成多个关键字参数
```

参数顺序：**位置参数（包括 *）** → **关键字参数（包括 **）**

② 函数定义时的四种形参模式

· **常规参数**

```
def func(r1, r2):       # 调用该函数时，必须提供实参值给该参数
```

· **默认值参数**

```
def func(r1=value1):   # 调用该函数时，如果没有提供值给这个参数，那么该参数就会使用默认值
```

· **默认值只有在函数被创建时才会创建**，而不是调用时

因此若以可变对象作为默认值参数，该参数在调用时将是上次调用结束后的样子

（所以不建议以可变对象作为默认值参数，但考试总喜欢考这种）

· **不定长数目参数**（2 种）

```
def func(arg1, *args):     # 调用时，多出的位置实参会打包成元组给该参数
```

```
def func(arg1, **kwargs):  # 调用时，多出的关键字实参会打包成字典给该参数
```

· 两种形参模式最多只能各有一个，名称不一定要是 args 和 kwargs

参数顺序：**常规参数 → 默认值参数 → **kwargs**，*args 不能在 **kwargs 后面

但*args 后面的参数将**只能通过关键字参数传递**，无法通过位置实参传递

③ **混合参数调用规则**

· 传递参数时，首先分配关键字参数，剩下的关键字参数若有**kwargs 打包成字典给它

位置参数再按照位置顺序从左到右分配给剩下来的形参，遇到*args 时，将截获剩下所有参数

④ **形参改动对实参的影响**

· 实参是不可变对象时，改变形参值不会影响实参；实参是可变对象时，改变形参值可能会影响实参

3. 函数返回值

· **如果函数没有用 return 语句返回，或 return 后面没有表达式，函数返回值为 None**

4. lambda 表达式

```
lambda x1, x2, … : expression
```

· 返回一个**函数对象**，这个函数对象以 x1，x2 等作为参数，以表达式的值作为返回值

表达式 expression 中包含 x1，x2，也可包含其它标识符，创建时标识符会变成其引用的对象值

· 这个函数对象可以赋值给标识符，通过标识符来调用，调用的方式于 def 定义的函数一模一样

四　命名空间和作用域

1. 命名空间

· 命名空间是内存中保存从标识符到对象的对应关系的地方

出现标识符时，程序会在对应的命名空间里查找该标识符，获得对应的对象

① **内置命名空间：**解释器启动时建立，记录所有标准常量名、标准函数名等

② **全局命名空间：**在**函数外定义的标识符**保存在这里

③ **局部命名空间：**在**函数内定义的标识符**保存在这里，每一个函数都有自己的命名空间

· 标识符查找顺序：**局部 → 全局 → 内置**

如 **局部标识符与全局标识符同名时，程序将优先选择局部标识符指向的对象，全局标识符被屏蔽**

在函数外创建一个标识符与内置函数同名时，该内置函数将无法被调用

2. global 关键字

```
global x
```

· 在函数内部使用，引入**全局空间**中的标识符，此后在函数内部调用 x，获得的是全局空间中指向的对象

典型例题

例 1 下面程序的输出是_____。

```
def func1():
    return 1
def func2():
    return 2
def func3():
    return 3
funclist = [func1, func2, func3]
print(sum([func() for func in funclist]))
```

解 最内层的列表推导式中，func 将遍历 func1，func2，func3 三个函数名

函数名指向函数对象，后面加()意味着调用该函数，获得返回值

因此列表将是[func1(), func2(), func3()]，返回值 [1, 2, 3]

因此函数 sum 的返回值为 1+2+3=6，程序输出 6

例 2 （判断）下面程序的输出是字符 a

```
c = "A"
print(c.lower)
```

解 虽然 lower 方法的功能确实是将字母变小写，但这里的 c.lower 只是函数对象

要使用功能必须要加()调用，因此输出不是字符 a，而是“str 类中一个方法”相关的描述

例 3 下面程序的输出是

A.6　　B.3　　C.1　　D.TypeError

```
def f1(a, b, c):
    print(a + b)
nums = (1,2,3)
f1(nums)
```

解 f1(nums)传递元组 nums 作为整体被 a 接收，b 和 c 没收到参数，又没有默认值，报错，选 D

例 4 下面程序的输出是

```
def func(x1, x2, x3, **x4):
    print(x1, x2, x3, x4)

func(x1=1, x2=22, x3=333, x4=4444)
```

解 前三个关键字参数被传递给对应形参，因此 x1=1，x2=22，x3=333

由于最后一个形参是**，且此时还剩下 x4=4444 没分配，因此 x4=4444 被打包成字典给 x4

也就是 x4 为{"x4": 4444}，因此程序输出 `1 22 333 {"x4": 4444}`

例 5 在一个函数中如局部变量和全局变量同名，则

A. 局部变量屏蔽全局变量　　B. 全局变量屏蔽局部变量

C. 全局变量和局部变量都不可用　　D. 程序将报错

解　A

例 6　判断：已知 x=3，则执行 x=7 后，id(x)的返回值与原来没有变化

解　原来 x 指向对象 3，id(x)返回的是 3 的地址，现在 x 指向对象 7，id(x)返回的是 7 的地址

这两个对象的地址显然是不同的，因此返回值肯定发生了变化，本题错误

例 7　python 语句 print(type(1J))的输出结果是

A.<class 'complex'>　　B.<class 'int'>

C.<class 'float'>　　D.<class 'dict'>

解　1J 是复数，因此类型名称为 complex，选 A