## Python语言常用的49个基本概念及含义

1. **列表（list）**：内置类型，可变（或不可哈希），其中可以包含任意类型的数据，支持使用下标和切片访问其中的某个或某些元素，常用方法有append()、insert()、remove()、pop()、sort()、reverse()、count()、index()，支持运算符+、+=、\*、\*=。可以使用[]直接定义列表，也可以使用list()把其他类型的可迭代对象转换为列表，列表推导式也可以用来创建列表，若干标准库函数、内置类型方法以及扩展库函数或方法也会返回列表。列表不能作为字典的“键”，也不能作为集合的元素。
2. **元组（tuple）**：内置类型，不可变（或可哈希），其中可以包含任意类型的数据，如果元组中只有一个元素，必须加一个逗号，例如(3,)。元组支持使用下标和切片访问其中的某个或某些元素，支持运算符+、\*。可以使用()直接定义元组，也可以使用tuple()把其他可迭代对象转换为元组，若干标准库函数、内置类型方法以及扩展库方法或方法也会返回元组。元组可以作为字典的“键”或者集合的元素，但是如果元组中包含列表、字典、集合或其他可变对象，就不能作为字典的“键”和集合的元素了。
3. **字典（dict）**：内置类型，常用于表示特定的映射关系或对应关系，可变（不可哈希），元素形式为“键:值”，其中“键”必须是可哈希类型的数据且不重复。如果创建字典时指定的“键”有重复，只保留最后一个，例如执行语句x = {'a': 96, 'b': 98, 'c': 99, 'a': 97}后x的值为{'a': 97, 'b': 98, 'c': 99}。
4. **集合（set）**：内置类型，可变（不可哈希），其中每个元素都必须可哈希且不会重复。
5. **字符串（str）**：内置类型，可哈希（不可变），可以是空字符串或包含任意多个任意字符的对象，使用单引号、双引号、三单引号、三双引号作为定界符，不同定界符之间可以嵌套。在字符串前面加字母r或R表示原始字符串，加字母f或F表示对其中的占位符进行格式化，可以在一个字符串前面同时加字母r和f（不区分大小写）。
6. **下标（subscript）**：对于列表、元组、字符串和range对象，可以使用整数作为下标来访问指定位置或序号的元素，例如x[0]。第一个元素的下标是0，第二个元素的下标是1，以此类推；如果使用负整数作下标的话，最后一个元素的下标为-1，倒数第二个元素的下标为-2，以此类推。对于字典，可以使用“键”作下标，返回对应元素的“值”。
7. **切片（slice）**：用来访问列表、元组、字符串和range中部分元素的语法，完整形式为[start:stop:step]，其中start、stop、step的含义与range()函数的参数相同。例如，'abcdefg'[:3]的结果为'abc'。
8. **表达式（expression）**：单个常量、变量以及若干常量、变量使用运算符或函数调用组成的式子都是合法表达式。表达式作为内置函数bool()的参数时如果返回True，那么这样的表达式作为条件表达式时表示条件成立。
9. **动态类型（dynamic type）**：在Python中，不需要声明变量的类型，第一次给某个变量赋值的语句会创建变量，每次重新赋值时会根据等号右侧表达式值的类型来动态改变变量的类型。
10. **解释型语言（interpreted language）**：Python程序不需要编译和链接为可执行程序，源代码就可以由Python解释器直接解释执行。
11. **伪编译（pseudo compilation）**：Python源程序可以通过多种方式伪编译为.pyc格式的字节码文件，Python解释器也可以直接解释和执行字节码文件。
12. **迭代器对象（iterator）**：同时具有特殊方法\_\_next\_\_()和\_\_iter\_\_()的对象，这类对象具有惰性求值特点，不能直接查看其中的内容，也不支持使用下标和切片访问其中的元素，可以把迭代器对象转换为列表、元组、集合，也可以使用for循环直接遍历其中的元素，或者使用内置函数next()获取迭代器对象中的下一个元素。不论使用哪种方式，每个元素只能使用一次。map对象、zip对象、enumerate对象、filter对象、reversed对象、生成器对象都属于迭代器对象。
13. **可迭代对象（iterable）**：具有特殊方法\_\_iter\_\_()的对象，可以使用for循环遍历其中的元素。列表、元组、字典、集合、字符串、以及各种迭代器对象都属于可迭代对象。
14. **可哈希对象（hashable object）**：可以计算哈希值的对象，概念等价于不可变对象。列表、字典、集合这样可以增加元素、删除元素、修改元素的对象属于不可哈希对象，元组、字符串这样的不可变对象属于可哈希对象。可以使用内置函数hash()计算一个对象的哈希值，如果试图计算不可哈希对象的哈希值会抛出异常。
15. **列表推导式（list comprehension）**：语法形式为[expr for var in iterable if condition]，计算结果为一个列表，可用于对iterable中的元素进行计算或过滤。
16. **生成器表达式（generator expression）**：语法形式为(expr for var in iterable if condition)，计算结果为一个生成器对象，生成器对象属于迭代器对象，具有惰性求值特点，不支持下标、切片，只能从前向后逐个访问其中的元素，且其中每个元素只能使用一次。
17. **字典推导式（dict comprehension）**：形如{key:value for key, value in iterable}这样的推导式，其中iterable中每个元素为包含两个元素的元组，并且每个元组的第一个元素为可哈希对象。字典推导式的结果为字典。
18. **集合推导式（set comprehension）**：形如{item for item in iterable}这样的推导式，其中iterable中每个元素都是可哈希对象。集合推导式的结果为集合。
19. **生成器对象（generator object）**：可以使用生成器表达式和生成器函数得到生成器对象。
20. **生成器函数（generator function）**：包含yield语句的函数，这样的函数调用时不是返回一个值，而是返回生成器对象。
21. **修饰器（decorator）**：一种特殊的函数，接收一个函数作为参数，对其功能进行补充或增强或限制，返回一个新函数。
22. **可调用对象（callable object）**：可以像函数一样的调用的对象，包括函数、lambda表达式、类（实际是调用的构造方法）、类方法、静态方法、对象的成员方法、定义了特殊方法\_\_call\_\_()的类的对象。
23. **lambda表达式（lambda expression）**：一种常用来定义匿名函数（没有名字的函数）的语法，功能相当于函数，属于可调用对象，常用于内置函数max()、min()、sorted()、map()、filter()以及标准库functools的函数reduce()的参数。在功能上，lambda x: x+5相当于接收一个数字然后加5返回的函数。也可以给lambda表达式起名字定义具名函数（具有名字的函数），func = lambda x, y: x+y相当于def func(x, y): return x+y。
24. **位置参数（positional argument）**：调用函数时严格按位置和顺序进行传递的参数，例如sorted(data, key=str)中的参数data。
25. **关键参数（keyword argument）**：调用函数时明确说明哪个实参传递给哪个形参，例如sorted(data, key=str)中的参数key。
26. **可变长度参数**：有def func(\*p)和def func(\*\*p)两种形式，前者可以接收任意多个位置参数并放入元组p中，后者可以接收任意多个关键参数并放入字典p中，元组或字典中元素数量取决于实参的数量。
27. **全局变量（global variable）**：如果一个变量的第一次赋值语句不在任何函数内部，那么它是全局变量。另外，在函数内部可以使用关键字global直接声明一个变量为全局变量。
28. **局部变量（local variable）**：在函数内部创建且没有使用关键字global声明的变量。
29. **变量作用域（variable scope）**：变量起作用的代码范围。在Python中，变量自定义开始，直到当前函数或文件结束，都是可以使用的，除非被声明为全局变量或者被更小的作用域内同名变量暂时隐藏。
30. **闭包作用域（enclosing scope）**：在Python中允许嵌套定义函数，也就是一个函数的定义中可以再定义函数。在内层函数中可以直接使用父函数中局部变量的值，但是如果要在内层函数中修改父函数中局部变量的值，必须使用关键字nonlocal声明该变量绑定到距离最近的父函数中已经存在的局部变量。
31. **序列解包（sequence unpacking）**：同时给多个变量赋值的语法，要求等号左侧变量的数量和等号右侧值的数量或者可迭代对象中元素数量严格一致。
32. **星号表达式（star expression）**：也属于序列解包的用法，在可迭代对象前面加一个星号表示把其中的元素都取出来，常见于把可迭代对象中的全部元素作为函数的位置参数的场合，例如print(\*'abc')。
33. **类（class）**：使用关键字class定义，是对某些具有相似特征和行为的对象的抽象。如果在类中定义了\_\_call\_\_()特殊方法，那么该类的所有对象都是可调用对象，可以像函数一样调用。在类中重新实现\_\_add\_\_()等特殊方法，可以实现对运算符或内置函数的支持。
34. **方法（method）**：形式类似于函数，表示特定的行为或运算，必须通过类或对象来调用，后者用的更多一些。一般来说，方法直接作用在调用方法的对象上，函数必须指定要操作的对象；自定义类时，属于对象的成员方法的第一个参数（一般名为self）表示对象自己，属于类的方法第一个参数（一般名为cls）表示类自己，都不需要显式传递，是调用时隐式绑定和传递的。可以使用修饰器把成员方法声明为属性。
35. **模块（module）**：包含若干函数、类、常量的Python程序文件。
36. **包（package）**：包含若干Python程序文件的文件夹，且其中有一个文件名为\_\_init\_\_.py。
37. **内置模块（built-in module）**：随同Python安装包一起安装，封装在Python解释器中，不存在独立的Python程序文件。
38. **标准库（standard library）**：随同Python安装包一起安装的Python程序文件，需要导入之后才能使用其中的对象，所有标准库对应的Python程序文件位于Python安装目录中的Lib子文件夹。
39. **扩展库（extension package）**：不随Python安装包一起安装，可以根据需要使用pip安装特定的扩展库，所有扩展库对应的文件默认位于Python安装目录中的Lib\site-packages子文件夹。
40. **异常（exception）**：代码运行时由于代码错误或某个条件临时不满足导致代码运行失败。
41. **语法错误（syntax error）**：存在语法错误的程序无法运行，例如缩进错误、在if选择结构的条件表达式中误用=运算符、在变量后面误用++等。
42. **逻辑错误（logical error）**：程序可以运行但是结果不对。