- Python
  - 内存管理
    - 引用计数
      - 当一个 Python 对象被引用时其引用计数增加 1， 当其不再被一个变量引用时则计数减 1. 当引用计数等于 0 时对象被释放
    - 垃圾收集（GC）
      - 1、引用计数
      - 2、标记-清除
        - 主要针对循环引用。如果两个对象的引用计数都为 1，但是仅仅存在他们之间的循环引用，那么这两个对象都是需要被回收的，也就是说，它们的引用计数虽然表现为非 0，但实际上有效的引用计数为 0。所以先将循环引用摘掉，就会得出这两个对象的有效计数
      - 3、分代回收
        - 所有内存块根据其存活时间划分为不同的集合，每一个集合就成为一个 代”
        - 垃圾收集的频率随着 代”的存活时间的增大而减小。如果一个对象经过的垃圾收集次数越多，则说明该对象存活事件越长
    - 内存池
      - Python 的内存机制呈现金字塔形状
        - 第 3 层是最上层，也就是我们对 Python 对象的直接操作；
        - 第 1 层和第 2 层是内存池，有 Python 的接口函数 PyMem_Malloc 函数实现，当对象小于256K 时有该层直接分配内存；
        - 第 0 层是 C 中的 malloc，free 等内存分配和释放函数进行操作；
        - -1，-2 层主要有操作系统进行操作
      - Python 内部默认的小块内存与大块内存的分界点定在 256 个字节，当申请的内存小于 256 字节时，PyObject_Malloc 会在内存池中申请内存；当申请的内存大于 256 字节时，PyObject_Malloc 的行为将蜕化为 malloc 的行为。当然，通过修改 Python 源代码，我们可以改变这个默认值，从而改变 Python 的默认内存管理行为
  - 协议描述符
    - __set__
    - __get__
      - 实例使用".xx"来访问属性时会触发此方法
    - __delete__
  - pyc
    - 字节码文件，python解释器使用。Python 解释器将源码转换为字节码，然后再由解释器来执行这些字节码
  - 内置属性
    - __name__
      - 返回绑定的类名与地址
    - __class__
      - 返回所在的类
    - __mro__
      - 返回继承链
  - 继承
    - 单继承
      - 子类不能访问父类的私有方法
      - 子类继承父类后，可以重写父类方法
    - 多继承
      - 继承顺序：先从左到右广度优先，再深度优先，深度优先时也按照之前一级从左到右的顺序
  - 私有属性或方法
    - 名称前面加两个下划线
  - 类
    - 类创建的不同实例对象，有各自不同的内存空间。因此，每个实例调用的方法都绑定在各自的实例的空间下
  - 闭包与装饰器
    - 闭包
      - 外部函数发现，自己的临时变量会在将来的内部函数中用到，自己在结束的时候，返回内函数的同时，会把外函数的临时变量送给内函数绑定在一起。所以外函数已经结束了，调用内函数的时候仍然能够使用外函数的临时变量
      - 使用 nonlocal 可以修改外部函数的值
    - 装饰器
      - 子主题
        - 外函数返回内函数的引用，并且调用了一下内函数
      - property
  - 上下文管理器
    - with处理的对象必须有__enter__()和__exit__()
      - __enter__()在语句块执行前先做的事
      - __exit__()在语句块执行完后做的事，通常用来捕获__enter__产生的异常。注意，只捕获__enter__产生的异常
      - 接收exc_type, exc_val, exc_tb三个参数
    - 在一些访问资源的场景下，需要不管是否发生异常都进行清理操作，释放资源
    - with语句会开辟出一块独立环境来执行文件的访问，类似沙盒机制
  - 元类
    - 动态语言和静态语言最大的不同，就是函数和类的定义，不是编译时定义的，而是运行时动态创建的。
    - class是由type()函数创建出来的，Python解释器遇到class定义时，仅仅是扫描一下class定义的语法，然后调用type()函数创建出class。
  - 异常处理
    - 如果有多个excep，那么它们会按照顺序进行捕获
      - 异常捕获
  - 协程
    - 自带CPU上线，比线程更小的执行单元
    - 调度完全由程序员控制
  - 线程
    - 系统最小可调度的单位
    - 每个进程都有一个主线程
    - 通过队列
      - 线程间通讯
  - 进程
    - 系统最小分配资源的单位
    - 可通过队列
      - 进程间通信
    - 多进程
  - lambda
    - 例如：filter(lambda x: x % 3 == 0, [1, 2, 3])指定将列表[1,2,3]中能够被3整除的元素过滤出来
    - lambda本质是一种匿名函数。通常将lambda赋给一个变量，然后再通过这个变量来调用这个lambda函数
  - 返回值
    - python中return，会根据接受者的数量，决定怎么样返回数据。如果要返回多个数据，但只有一个接受者，则会将这些数据打包成一个tuple返回
  - copy与deep copy方法
    - 与赋值行为相同
      - 不可变对象
    - 可变对象每次创建都会申请内存空间，copy出来的不受原对象的影响
      - 可变对象
  - 赋值拷贝
    - 相同值的不可变对象不会申请内存空间，会使用同一个内存地址
      - 不可变对象
    - 赋值方式创建出来的对象与原对象地址相同，两者互相影响
      - 可变对象
  - 数据类型
    - 不可变对象
      - string
      - tuples
      - int、float
    - 可变对象
      - list
      - dict
      - set
  - frozenset
    - 一旦创建就不能做任何改变