# Lecture 3

* Pandas:
  + 数据分析与处理
  + 一维series：s = pd.Series([1,2,3], index=[‘a’,’b’,’c’])
  + 二维data frame: data = {‘name’:[‘Bob’, ‘Alice’], ‘age’:[12, 13], ‘city’: [‘Paris’, ‘London’]}; df = pd.DataFrame(data)
  + For index, row in data.iterrows(): 逐行遍历dataframe,每行row都是一个series
  + Array.astype(dtype):转换数据的类型
  + Series\_or\_df.to\_numpy():把series/dataframe转换成数组
* Profiling:
  + Assert 条件：用于判断这个条件是否成立，成立则继续运行，否则报错
  + Wall time:
    - Start = time.time()
    - End = time.time()
    - Wall\_time = start – end
    - 指的是现实世界里“挂钟”经过的时间,包括CPU 执行时间、系统调度延迟、I/O 等待、等待其他线程或进程的时间，是**真实经过的时间**
    - **Cpu time + time outside code = wall time**
  + Cpu time:
    - Time.process\_time()
    - 有可能比wall time长
  + Timeit:
    - Measure wall time
    - %timeit -r 10 -n 100 1 + 2
    - 会自动设置run和loop的次数
    - 如果标准差太大，重新运行（让timeit来决定loop和run的次数），如果还是很大，可能是硬件问题（e.g.有其他程序在运行）
    - 多行代码：%%timeit; 代码
    - 存放每次重复运行的耗时列表: timeit\_object.timings
    - timeit\_object.average
    - timeit\_object.best
    - timeit\_object.worst
    - standard deviation：timeit\_object.stdev
  + cProfile:
    - 累计时间：%prun -l 8 -s cumulative 代码
    - Cumtime：从代码的第一行到最后一行到时间
    - Tottime：真正在这个函数里面的代码的运行时间，而不是call的其他函数
    - Line\_profiler: %lprun -f estimate\_pi estimate\_pi(10000)
* Algorithm:
  + Time complexity:
    - Big O / O(f(n)):
      * Upper bound: n趋于极限
      * N: the size of input
      * Common big O:
        + O(1): runtime doesn’t change with input
        + O(n): runtime grows linearly with the input
        + O(n^2)
        + O(log n): binary search二分查找
        + O(n log n)
        + O(2^n)
* Optimization：
  + 不同算法algorithm，选择更好的big O
  + Select better data structure，比如numpy
  + Caching
* Memory management:
  + Float64 --- 64 bits ---8 bytes
  + 字符串---category  
    float64 / int64 --- 减小内存