# Lecture 2

* Numpy
  + Array programming
  + Creating 1D arrays:
    - Create with list/ array: a = np.array([1,2,3]) (type(a) = class ‘numpy.ndarray’)
    - A = np.arange(start, end, step) (不包含end)
    - A = np.linspace(start, end, number) (包含end)
    - A = np.zeros(3)
    - A = np.ones(3)
    - A = np.eye(3) (identity matrix)
    - A = np.array(list, 参数（dtype=float32/ complex）)
  + Array attributes:
    - A.ndim: number of dimensions 看开头的括号数
    - A.shape: number of elements in each dimension 个数与dimension数符合
    - A.size: total number of elements
    - A.dtype: data type of element
  + Multi-dimensional arrays:
    - Create from list of lists
    - From pprint import pprint
  + Accessing arrays:
    - Slicing: a[start: stop: step] (not inclusive of stop)
    - A[1,0,1]比a[1][0][1]更好，更快：a[][]会一步一步的走
    - 省略的部分将会当成一整行/列
    - My\_slice = (slice(2, -3, 2)); print(a[my\_slice])
    - A.reshape(3, 3) 个数要与原矩阵相同,是视图
    - A.shape = (3,3)直接赋值，要符合个数,是视图
    - B = np.resize(a, (3, 3)) 会自动循环补齐或裁剪
      * B.base is a --- False: b是a的独立副本(copy)，不是视图(view)
    - Use negative steps to revert an array
  + Array copy:
    - b = a.copy() 避免指向同一个id
    - s = a[2:3, 1:3],当s改变时，a也同样改变，这是一个view视图
  + vectorization:
    - element wise multiplication: a\*b（每个对应元素相乘）
    - vector/ matrix multiplication: a.dot(b.T)
    - y = f(x) x是数组，y也是数组
  + manipulating array:
    - concatenate((x1, x2), axis=0) 竖着拼接；axis=1 横着拼接
  + fancy indexing:
    - a[rows, columns] --- rows = [list], columns = [list]
    - 是独立副本，而不像slicing是视图
  + Logical expressions: m[m<0]
  + Random number:
    - a = np.random.ranf(10) 浮点数 0-1
    - y = np.random.random(5) 浮点数 0-1
    - a = np.random.randint(0,high=5,size=25) 不包括end
    - normal distribution(mean=0, standard deviation=1): s = np.random.standard\_normal((5,5))；np.random.randn(10000)
  + file operation:
    - 保存数据到文件：np.savetxt(‘文件名’)
    - Reload data to an array: p = np.loadtxt(' 文件名'')
    - p = np.genfromtxt('savedata.txt', skip\_header=2, skip\_footer=1)
  + numpy.linalg:
    - 矩阵相乘：np.dot(A,B)
    - 范数：la.norm(A)
    - 行列式：la.det(A)
    - 求解Ax=B: la.solve(A,B)
    - 矩阵的逆：la.inv(A) 不推荐
    - 特征值：eival, eivec = la.eig(A)
  + Performance：
    - %timeit -n <iterations>（每次里面循环几次） -r <repeats>（运行几次） <code\_snippet>
* Scipy:
  + Linear algebra:
    - Import scipy.linalg as sla
    - 计算正交基：sla.orth(A)
    - Schur分解：T, Z = sla.schur(A) 其中A=Z @ T @ Z^t
  + Integration:
    - import scipy.integrate as sint
    - 计算连续函数的定积分即曲线下的面积：sint.quad(f(x), 0, 4)
    - 辛普森法则对离散数据进行数值积分: sint.simpson()
  + Interpolation:
    - import scipy.interpolate as sitp
    - 拉格朗日插值法拟合一组离散数据点: lp=sitp.lagrange(x, y) #返回一个拉格朗日多项式
    - 多项式拟合：poly\_coeffs=np.polyfit(x, y, 3)#3次多项式； p3 = np.poly1d(poly\_coeffs)
  + Optimization:
    - import scipy.optimize as sopt
    - 找一次函数最小值：result = sopt.minimize\_scalar(f)； x\_min = result.x
    - 找二次函数最小值：sopt.minimize()
* Matplotlib:
  + Import matplotlib.pyplot as plt
  + 画图：
    - Fig = plt.figure()
    - Plt.xlim(left, right)
    - Plt.xlabel(‘x’)
    - Plt.title(‘title’)
    - Plt.plot(x,y, label=’sinx’)
    - Plt.legend()
    - 注释：plt.annotate()
    - Plt.show()
    - 直方图：plt.hist(x,50)(50个等分区间)
    - 保存图像：plt.savefig("sin.png", dpi=200)
    - Subplot:
      * fig, axes = plt.subplots(row, column, figsize=(10,7))
      * axes[0,1].plot(x, y)
      * axes[1,2].set\_xlabel(‘x’)
      * plt.show()
      * 切换当前操作的字图：plt.sca(axes[0,0]) #set current axes
      * plt.subplot2grid()