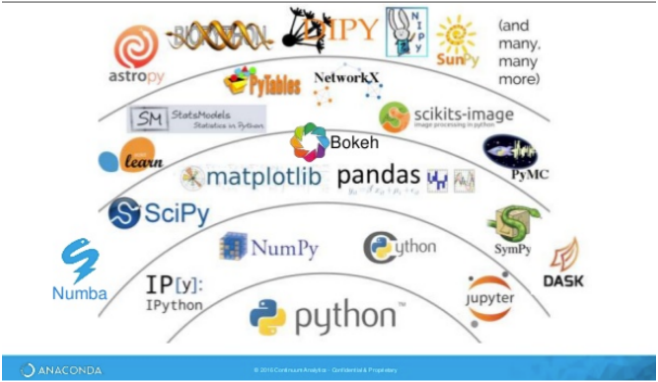
BSU SIAM Student Chapter Python Workshop

Boise State University

June 17, 2021

Python Uses and Applications



Python Uses and Applications

Basics Python

- Data types and variable declaration
- Data structures
- Loops and conditionals

Common Data Types

type	Python syntax
integer	int()
float 32 bit and 64 bit	float()
string	str()
complex	complex()

Conditionals statements

- The **if** keyword is used to execute a statement or a block if, and only if, a condition is fulfilled
- The basic code structure looks like this:

Loops

- Loops repeat a statement a certain number of times, or while a condition is fulfilled
- Types of loops:
  - **while**
    - Repeats a code body while the condition/expression is true
    - When the condition is no longer true, the loop ends
  - **for**
    - It is designed to iterate a number of times
    - Loop repeats while the condition is true

Input/Output

- print()
- input()
- file I/O: open()
    - read/write/append (rwa)

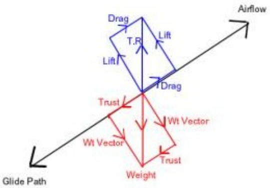
Python Practice

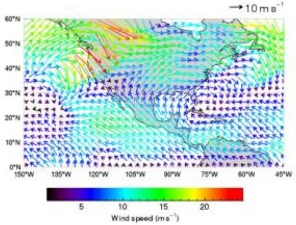
- Build a program with three floating point variables (a_i, b_i, c_i), two integers (i,j) , and allocate memory for three $N = 10$ element data structures (a, b, c) (1D arrays).
- Build a program that fills the lists (a,b,c) with the respective values (a_i, b_i, c_i) .
- Build a program that elements of the lists to a file.

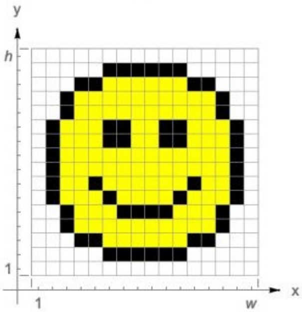
Matrices and Vectors

- vector-vector operations
- matrix-matrix operations:
- matrix-vector operations
- solving linear systems (direct or iterative methods):

Matrices and vectors







Vector operations

- vector arithmetic

$$\sum_{i=1}^{2}(a_i + b_i) = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \end{bmatrix} \tag{1}$$

- dot product

$$\sum_{i=1}^{2} a_i b_i = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = a_1 b_1 + a_2 b_2 \tag{2}$$

Matrix operations

- matrix arithmetic

$$\sum_{i=0}^{2}\sum_{j=0}^{2}(a_{ij}+b_{ij}) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{bmatrix} \tag{3}$$

- matrix products

$$\sum_{i=1}^{2}(a_{ij}x_j) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{bmatrix} \tag{4}$$

Solving Linear Systems of Equations

download.png

Figure: General form of a system of linear equations in two variables

We can solve this system using matrices

- The coefficients of variables x and y form a column each in the 2d matrix
- The constants on the right hand side form another vector

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

$$A\ \boldsymbol{x} = \boldsymbol{c}$$

Practice Problems

1. Build a program that performs a vector addition and subtraction, dot product, and normalize a vector
2. Build a program that performs a matrix addition and subtraction, matrix-matrix product
3. Build a program that performs a matrix-vector product

Comp. Mathematics Applications

- Numerical methods
- Numerical differentiation and integration
- Solving differential equations

Numerical Methods Examples

- Finding the square root of a number (i.e. Newton's Method)

$$x_{k+1} = x_k + f(x_k)/f'(x_k) \tag{5}$$

- exponential functions

$$e^x = \lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n \tag{6}$$

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{7}$$

- data/ function reconstruction (interpolation or least squares)
- numerical differentiation

$$u''(x) = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + O(h^2) \tag{8}$$

- solving differential equations numerically (iterative methods (Jacobi))

$$u''(x) = f(x) \in \Omega \tag{9}$$

$$u(0) = u(1) = 0 \; on \; \partial\Omega \tag{10}$$

Practice Problems

- Compute number e accurate to 8 decimal places by using the following formulae:

$$a_n = \frac{x^n}{n!} = \frac{x}{n} a_{n-1}$$

$$s_n = \sum_{i=1}^{n} a_n = s_{n-1} + a_n$$

where s_n represent the value e computed by summation of n terms of a_n.

Practice Problems

The combinations function C(n, k) determines the number of ways you can choose k values from a set of n elements, ignoring the order of the elements:

$$C(n,k) = \frac{n!}{(n-k)!k!} \tag{11}$$

If the order of the value matters then the function is denoted as P(n, k):

$$P(n,k) = \frac{n!}{(n-k)!} \tag{12}$$

- Build functions combinations(n, k) and permutations(n, k) that computes the C(n, k) and P(n, k) function without calling the factorial function.
- Build the drive code to compute C(47, 3) and P(47, 3) respectively.

- Build a program that numerically computes the integrals and derivatives of the following functions
  $f(x) = x, x^2, x\sin(x), \sin(x)\cos(x^2), \exp\left(x + \sin(x)\right)$.
- Build a program that reads in the data file **temperature.dat** and interpolates the data at $4N$ evaluation points.
- Build a program solves the following differential equations
  - $u_{xx}(x) = 1$
  - $u_{xx}(x) = -\sin x$
  - $u_t(t, x) = u_{xx}(t, x)$

  , $u_{xx}(x) = -\sin(x)$ with the following boundary conditions
  $u(0) = u(1) = 0$.