Python

Being a brief introduction to the python language for computer scientists with a familiarity with another programming language.

Python

- python is an interpreted language
  - like lisp
  - with text processing capabilities and a more free for all approach
    - it can be object oriented, but doesn't have to be.
    - it can be entered in a script file, or at the interpreter etc.
    - can have methods/functions, but can execute simple statements as well.
    - dynamic typing.
      - vars are the type of the value you assign to them, no need to declare before assigning value.
  - part of the growing trend of new languages looking to be the “next thing” along with ruby and the like.

Hello World

- Hello world in python
  - mystring = 'hello world'
  - print mystring
- thats it
  - not bad
  - Note no { or } or ;
  - white space is used for separators and scoping.

Running python

- On eagle/csdev01
  - you'll get accounts
  - type python at command line to bring you to the interactive prompt.
  - type
    - python <file>
    - with <file> replaced by your python file to run a python program from your file.
    - python programs traditionally have the suffix .py on them.

Python Output

- print statement
  - note statement not function
  - can print a plain string (using “string” or 'string')
  - or a formatted string
- format operator %
  - syntax:
    - <format string> % (<arg1>, <arg2>, ...)
    - take the same format commands at c printf
    - %s string, %d integer, %f floating point etc.

program input.

- easiest way to get command line input is to use the function raw_input
- syntax
  - raw_input('prompt')
    - where prompt is replaced by any prompt string.
    - the function returns a string.
    - you can convert a string to a number by using the int() method
  - inputstr = raw_input('gimme input:')
  - inputstr is now available for the program to use.

comments

- # is the line comment character
  - everything from # to the end of the line is a comment
- documentation comments also possible
  - when declaring a function (to be discussed) if the first thing is a string, it is considered a comment.

Math operators

- Standard operators with a couple of new things
  - +, -, *, /
    - subtraction, addition, multiplication and floating point division
  - //
    - 'floor division' returns an integer w/out remainder
  - ** exponent : 5**3 is 125

logical operators

- the usual logical operators for c/java like languages are available
  - >, <, >=, <=, ==, !=
- also legacy <>
  - pascal style not equal, being phased out.

functions in python

- functions need to be declared before used.
- declare using
  - def <functionName>([arguments]) :
    - “optional doc string”
    - <function body.>
  - where functionName is a valid identifier for the function
  - arguments is an optional list of arguments (thus the brackets)
  - notice the indenting in the lines following the def line – that matters!!

functions in python II

- no need to specify return type
  - to return a value,
    - use return keyword followed by a value
  - if you don't explicitly return a value
    - **none** is returned
    - none is the python keyword equivalent to null in c++/Java
- example
  - def getInput(prompt):
  - inputstr = raw_input(prompt)
  - return inputstr

call functions

- No static typing even on parameters,
  - pass what you want, run time error rather than syntax error if you pass a value the function can't handle.
- myInput = getInput(“tell me what you want : ”)

Identifiers

- Same rules as in Java/C++
- any letters, numbers and _
- number can't come first
- case matters.

variables

- dynamically typed language, define and type variables when you initialize it with a value.
  - str = getInput("show me show me show me: ")
  - print str
  - str = 3
  - print str
  - output
    - show me show me show me: this and that
    - this and that
    - 3

Variable assignment and updating

- use = for variable assignment
  - like c/java
  - augmented assignment available
  - n=10
  - n = n*10 is same as n*= 10
  - but no ++ and -- – like in c/java
    - unary operators – --n is same as -(-n) aka n

Strings

- string:
  - sequence of characters inside of ' ' or “ “
  - triple quotes ''' ''' ore “”” “”” are for strings with special characters in them
  - var = ''' this is a
    string with a newline in it '''
  - str = “string”
- use len() function to find number of chars in string
  - len(str) will return 6

string operators

- two most common operators
  - index [] and slice [:] operators
  - want a character from a string use the index (like an array in c/java)
    - str = 'string'
    - str[1] will return t
  - strings and other collections zero based in python
  - slice [begin:end] (if either is omitted goes to the end)
    - from beginning upto but not including end
    - str[1:4] returns tri
    - str[3:] returns ing
    - str[:3] returns str

conditional

- conditional in python, like others is if
  - syntax:
    - if *expression*:
      - *if_block*
  - if the expression evaluates to True or non-zero, if_block will be executed. if expression evaluates to False or 0, then if_block will not be executed.
  - if_block is a series of statements indented one indent greater than the if expression.
  - optional else: after if_block or elif
  - see example in two slides

String membership

- Want to check to see if there is a substring in an input string
  - use in operator
  - str='example'
  - `'am' in str` returns True
- `not in` also available
  - 'good' not in 'evil' returns true.

putting some together

- a function with conditionals and strings
  - def really(input):
  - if 'mother' in input:
  - print "tell me about your mother"
  - else:
  - print "oh really"
- calling that function
  - chat = getInput("tell me about whats bothering you:")
  - really(chat)

indefinite loops

- for indefinite loops
  - while same as c/java
  - syntax
    - while *expression* :
      while_body
  - this will execute all the lines of *while_body* until expression evaluates to 0 or False
- for loops exist, but are different in python than in java/c

Very basic syntax

- Thats the most basic python syntax
- now lets learn a little about classes and some common support functions and libraries