20.180:Python

Introduction to the Python Tutorial

The goal of this introduction is to get you familiar with basic terms and concepts, so that the tutorial is easier to follow. Also, we hope that this will allow you to spend more time implementing each example that is discussed in the tutorial in order to get a feel for programming.

Python is an object-oriented, interpreted programming language. We will not go into object-oriented programming too much in this basic tutorial, and for now we will just think of everything in Python as an object. In Python, we write programs to do things with "stuff", where "stuff" are objects.

Python programs can be decomposed into modules, statements, and objects:

1. Programs are composed of modules
2. Modules contain statements
3. Statements create and process objects

Python has various data types, including numbers (normal integers, floating-points), strings ("Laura"), lists, dictionaries, and files. A specific instance of one of these is an object of that class.

NUMBERS:

Tools for processing number objects:

1. expression operators (+, *, >>, etc.)
    1. A combination of numbers (or other objects) and operators that computes a value when executed by Python. Expressions are written using the usual mathematical notation and operator symbols.
    2. Example: Add two numbers X and Y; the result is another number object that is the sum of X and Y. If Z=X+Y, the variable Z now has the value of the sum.
2. built-in mathematical functions (pow, abs, etc.)
3. utility modules (rand, math, etc.)

STRINGS: ordered collection of characters used to store and represent text-based information

Common string operations:

1. concatenation (combining strings)
2. slicing
3. indexing
    1. Access characters of strings by their position
    2. Python offset starts at zero and ends at one less than the length of the string
4. length

Note: uses the same operators that were called addition and multiplication when we were working with numbers. Cannot mix numbers and strings using these operations.

LISTS: most flexible ordered collection object type; can contain any sort of object: numbers, strings, even other lists - can treat them as a group

Common list operations: (similar sequence operations we put to work on strings earlier)

1. concatenation (combining lists)
2. slicing
3. indexing - fetch a component object out of a list
4. length
5. method calls: append (adding items to the end of the list), sort, reverse
6. deletion of an item in the list or a section of the list
7. index assignments - replace an item in a list
8. nesting - list within a list

Note: cannot concatenate strings and lists.

DICTIONARIES: unordered collection of objects that maps keys to values; keys can only be string objects or numbers, but values can contain objects of any type (including lists and dictionaries)

Common dictionary operations:

1. nesting - dictionary within a dictionary, or list within a dictionary
2. indexing by key - items are sorted and fetched in dictionaries by keys, not offset, so no particular order of items (keys provide the symbolic - not physical - location of items in a dictionary)
3. methods: membership test (does a dictionary contain a certain key), keys list, values list
4. length - number stored entries
5. adding/changing, deleting entries in the dictionary

Note: does not support sequence operations that work on strings and lists because those operations depend on fixed order

Note: no append method is needed because whenever a new dictionary key is assigned you create a new entry in the dictionary

Summary: Three data type (and operation) catagories in Python -

1. numbers support addition, multiplication, etc.
2. sequences (strings and lists) support indexing, slicing, concatenation, etc.
3. mapping (dictionaries) supports indexing by key, etc.

There are two ways to use Python: the interactive interpreter command line, which erases your code once you close Python, or save code permanently in a module file (text file with Python statements) and then excute the file at the Python command line. Most of the time you will be creating module files using the Pico editor, but the interactive mode is helpful when testing out new code or debugging.

Now you are ready to get started with the Unix and Python Tutorial.