NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 8, Lecture 5

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Python vs other languages

- Python is a good programming language to start with because
  - No declaration of names in advance
  - Indentation avoids punctuation — { }, ( ), ;
  - No explicit memory management
- Are there any down sides to this?

Debugging

- Declaring names helps debug code
  - "Simple" typos are caught by compiler
  - Mistyped name will be "undeclared"
- Static typing — assigning types to names
  - Again catch "simple" typos by type mismatch

Classes and objects

- Can only associate a type with a name by creating an object
- Empty tree, with name and type declarations
  - Declare `t` to be of type `Tree`
  - Empty tree — `t` has value `None`
- Instead, cumbersome convention with empty nodes to denote frontier etc

Classes and objects

* We want public interface, private implementation
* For a `Point p`, `p.x` and `p.y` should not be available directly outside the class
  * Stack implemented as a list has public methods `push()` and `pop()` but `s.append()` not ruled out
* Need to declare parts of implementation private
  * Only methods inside the class can access private names

Classes and objects

- Ideally, all internal names are private
- Special functions to access and update values
  - `p.getx()` gets x-coordinate
  - `p.setx(v)` sets x-coordinate
- x-coordinate is an “abstract” attribute
- Works even if internal representation is (r,θ)

Classes and objects

- Handle integrity of compound values
- Date is a tuple (day,month,year)
  - Range for day is 1—31, month is 1—12
  - Valid combinations depend on all three fields
    - 29 - 02 is valid only in a leap year
- `d.setdate(d,m,y)` vs separate `d.setd(d)`, `d.setm(m)`, `d.sety(y)`

Storage allocation

- Python needs to allocate space dynamically
  - Each assignment to a name could a new type
- Name declarations allow some static allocation
  - Still need dynamic allocation for lists, trees etc that grow at run time
  - Static arrays can optimize access time: base address plus offset

Dynamic storage

- What happens when we execute `del(x)`?
- Or when we delete a list node by bypassing it?
- Do these “dead” values continue to use memory?

Garbage collection

- Python, Java and other languages reclaim space using automatic "garbage collection"
  - Periodically mark all memory reachable from names in use in the program
  - Collect all unmarked memory locations as free space
  - Run time overhead to schedule garbage collector
- In C, need to explicitly ask for and return dynamic memory

Memory leaks

- Manual memory allocation is error prone
- Forgetting to return junk space to free list results in memory “leaking” out of the system
  - Performance suffers over time as space shrinks
- All modern languages use garbage collection
  - Run time overhead more than compensated by reduction of errors due to manual management

Functional programming

- Declarative vs imperative
- "What to compute" vs "how to compute it"
- Directly specify functions inductively

```
factorial :: Int -> Int   # Type

factorial 0 = 1
factorial n = n * factorial (n-1)
```

Functional programming

- List processing

```
sumlist :: [Int] -> Int

sumlist [] = 0
sumlist l = (head l) + sumlist (tail l)
```

Functional programming

- Many features of Python are modelled on functional programming
  - `map`, `filter` and other “higher order” functions
  - List comprehensions

Summary

- No programming language is “universally” the best
  - Otherwise why are there so many?
- Python’s simplicity makes it attractive to learn
  - But also results in some limitations
- Use the language that suits your task best
- Learn programming, not programming languages!