Lecture 41: Python

CS 51G
Spring 2018
Kim Bruce

Announcements

- Test program 2
- What to review?

Slices

- if x = “abcde” then x[1] is “b”, x[-1] is e,
- Slices: x[1:3] is “bc”, x[3:] is “de”, x[:3] is “abc”,
- Works for lists as well as strings

Tying it all Together!

INEFFECTIVE SORTS

```
DEFINE HALFHEARTEDMERGESORT(LIST):
    IF LENGTH(LIST) < 2:
        RETURN LIST
    PIVOT = INT(LENGTH(LIST) / 2)
    A = HALFHEARTEDMERGESORT(LIST[:PIVOT])
    B = HALFHEARTEDMERGESORT(LIST[PIVOT:])
    // UMMMMM
    RETURN [A, B] // HERE. SORRY.
```

```
DEFINE FASTBOGOSORT(LIST):
    // AN OPTIMIZED BOGOSORT
    // RUNS IN O(N LOG N)
    FOR N FROM 1 TO LOG(LENGTH(LIST)):
        SHUFFLE(LIST):
        IF ISSORTED(LIST):
            RETURN LIST
    RETURN "KERNEL PAGE FAULT (ERROR CODE: 2)"
```

```
DEFINE JOBINTERVIEWQUICKSORT(LIST):
    OK SO YOU CHOOSE A PIVOT
    THEN DIVIDE THE LIST IN HALF
    FOR EACH HALF:
        CHECK TO SEE IF IT'S SORTED
            NO, WAIT, IT DOESN'T MATTER
        COMPARE EACH ELEMENT TO THE PIVOT
            THE BIGGER ONES GO IN A NEW LIST
            THE EQUAL ONES GO INTO, UH
            THE SECOND LIST FROM BEFORE
        HANG ON, LET ME NAME THE LISTS
            THIS IS LIST A
            THE NEW ONE IS LIST B
        PUT THE BIG ONES INTO LIST B
        NOW TAKE THE SECOND LIST
            CALL IT LIST, UH, A2
        WHICH ONE WAS THE PIVOT IN?
        SCRATCH ALL THAT
        IT JUST RECURSIVELY CALLS ITSELF
        UNTIL BOTH LISTS ARE EMPTY
            RIGHT?
        NOT EMPTY, BUT YOU KNOW WHAT I MEAN
    AM I ALLOWED TO USE THE STANDARD LIBRARIES?
```

```
DEFINE PANICSORT(LIST):
    IF ISSORTED(LIST):
        RETURN LIST
    FOR N FROM 1 TO 10000:
        PIVOT = RANDOM(0, LENGTH(LIST))
        LIST = LIST[PIVOT:] + LIST[:PIVOT]
        IF ISSORTED(LIST):
            RETURN LIST
    IF ISSORTED(LIST):
        RETURN LIST:
    IF ISSORTED(LIST):  //THIS CAN'T BE HAPPENING
        RETURN LIST
    IF ISSORTED(LIST): // COME ON COME ON
        RETURN LIST
    // OH JEEZ
    // I'M GONNA BE IN SO MUCH TROUBLE
    LIST = [ ]
    SYSTEM("SHUTDOWN -H +5")
    SYSTEM("RM -RF ./")
    SYSTEM("RM -RF ~/*")
    SYSTEM("RM -RF /")
    SYSTEM("RD /5 /Q C:\*")  //PORTABILITY
    RETURN [1, 2, 3, 4, 5]
```

Dictionaries in Python

- Dictionaries are collections that pair a key with a value.
- Example: Phone book pairs name with phone number
- Properties of colors are r, g, and b components
- In Python, pairs designated by ":" to join
  - Keys are unordered.
  - Keys must be immutable!!

More Dictionaries

```
city_population = {"New York City":8550405,
    "Los Angeles":3971883, "Toronto":2731571,
    “Chicago":2720546, "Houston":2296224,
    "Montreal":1704694, "Calgary":1239220,
    "Vancouver":631486, "Boston":667137}

print (city_population["New York City"])  # gives 8550405
print(city_population)        # comes out in different order

city_population["Claremont"] = 35000   # add new city
newDictionary = {}            # create new empty dictionary
```

Why Dictionaries

- Like unordered list where look up items by key rather than index.
- Useful in lots of applications
  - Grace has them as well...

Anonymous Functions

- In GraceL {x:Type -> code to execute}
  - Used to add listeners to GUI components
  - For loops, etc.
- In Python: lambda x,y: ...x...y...
  - Example: lambda x: x * x
  - Must represent expression, no side effects

Anonymous Functions

```
squaring = lambda x: x * x
lsquares = list(map(squaring,[1,2,3,4]))
print (lsquares)

evenFilter = lambda x: x % 2 == 0
levens = list(filter(evenFilter, [1,2,3,4]))
print (levens)
```

Results of map and filter are iterators, must cast to lists

Python vs Grace

- Similar syntax
  - Grace requires variable declarations
  - rtobjectdraw dialect requires type annotations
    - Provides better error messages
    - Python has no type annotation
  - Blocks in Grace with {}, in Python with :
- Both use lists
- Object-oriented programming more natural in Grace

Python vs Grace

- Python easier to write (no types)
  - but harder to read (no types)
  - & very hard to analyze
- Big asset for Python:
  - Great efficient libraries
  - In use in lots of contexts.
- Transition from Grace to Java much simpler than Python to Java.

Review for Final

- No labs using:
  - Inheritance
  - Sorting/Searching
- ... but they will definitely be on final.

Questions?