Review of C.S. cont.

Crista Falk
3/15/17
A2

WHY STUDY ALGORITHMS

- experience = problem-solving techniques/dif algorithms to use in future problems
- Algorithms have dif. implementations, complexities, characteristics, tradeoffs
- "Intractable" - No algorithm to solve prob. in realistic amt of time

REVIEW of BASIC PYTHON

powerful set of built-in D.T. & control constructs

- easy-to-learn, modern object-oriented programming lang
- Interpreted language - most easily reviewed by looking @ it and describing interactive sessions (interpretter displays >>> prompt, then evaluates construct)

O.O.P paradigm considers data (objects) to be focal point of prob. solving process

GETTING STARTED w/DATA

- class - a description of what data look like & can do
- Data items in object oriented paradigms = objects (state) (behavior) - instance of a class
- Atomic data types:
  - 2 numeric classes = int (eg 7) float (eg 0.271)
  - Operations: aritmetic +, -, *, /, ** (exponent) also % & int division //
  - Booleans - True/False; operators: and, or, not
  - Logical operators ==, >=, <, etc
- Builtin collection D.T.

* Mutability - ability to modify; Immutable = unchangeable constant

*I x = [ ] • list - ordered collection of Ø or more ref. to Python data objects
  OPS: index [ ] concatenation +, rep *, in, len(), slicing [:]
  Methods: .append(), .pop(), del, .reverse(), etc

used to build d.s.

*I x = "" • string - sequential collection of Ø or more letters, #, symbols
  Methods: .split(schar), .center(), .count(), .lower(), etc

*I x = ( ) • Tuples - heterogeneous sequence of data (immutable list)

*M x = { } • Sets - OPS: in, len, |, &, -, <= (asks whether all elements of 1st set are in 2nd)

unordered collection of Ø or more immutable Python D. objects - Don't allow duplicates, comma delineated

FURTHER DEFINED ON DIF PAGE OF NOTES! along w/ dictionaries, etc

runtime errors = exceptions
try: ← trying
except: ← if error
can also raise err.()

INPUT/OUTPUT

- string prompt input ("x", "y")
- string formatting (, sep = "###") (, end = "###")
- % format operator ("%S" % (age)) CONVERSION SPECIFIER %S string. %d int, etc
- Format method as well

• ("x", y, "z") comma = inline