Computational Structures in Data Science

Lecture #2: Algorithmic Structures

January 26, 2018


https://inst.eecs.berkeley.edu/~cs88

Requirements for CS61b and CS Major

• Data8+CS88 qualify you for CS61b
• Only CS majors: Need to take CS47a any time after CS88 to fulfill requirements.

Computational Concepts today

• Algorithm, Code, Data, Information
• Data Types, Simple Data Structures
• Function Definition Statement
• Conditional Statement
• Iteration

Algorithm

• An algorithm (pronounced AL-go-rith-um) is a procedure or formula to solve a problem.
• An algorithm is a sequence of instructions to change the state of a system. For example: A computer’s memory, your brain (math), or the ingredients to prepare food (cooking recipe).

Algorithm: Properties

• An algorithm is a description that can be expressed within a finite amount of space and time.
• Executing the algorithm may take infinite space and/or time, e.g. “calculate all prime numbers”.
• In CS and math, we prefer to use well-defined formal languages for defining an algorithm.

6 ÷ 2(1+2) = ?

1 or 9

Algorithm: Well-definition
Algorithms early in life

```
+ 1 1 0 0
  1 0 1 0
  + 1 0 1 0
  -----------
  1 0 1 0 1
```

Carry (MSD)  Least significant digit of result
operands

More Terminology (intuitive)

- **Code**
  A sequence of symbols used for communication between systems (brains, computers, brain-to-computer)

- **Data**
  Collection of facts and inference (for reference or analysis)

- **Information**
  Reduction of uncertainty about a fact (usually measured in bits)

Experiment

**Code vs Data vs Information**

Data or Code?

```
00000000 10000000 01000001 10000000 00100000 00000000 10000001
01000001 10000000 00010000 00000000 10000002 01000000 10000000
00100000 00000000 10000003 01000000 10000002 00100000 00000000
10022133 01000001 10022133 00010000 00000000 10000000 01000000
20000000 10000000 00000000 10000000 00000000 00000000 10000000
00000000 00000000 00000000 00000000 00000000 10000000 00121212
01000000 10001212 00010000 00000000 00000000 00000000 01000000
00100000 00000000 10000002 01000001 10000002 00100000 00000000
00000000 10100101 10000000 00010000 00000000 10000001 01000000
10000001 00010000 00000000 10111111 01011000 10000000 00010000
00000000 10100020 01010001 10011101 00110000 00000000 10000000
```

Data or Code?
Data or Code?

Here is some information!

```
00000000 10000000 01000001 00000000 01000001 00000001
01000000 00100000 00100000 00100000 00100000 00100000
00100000 00000000 00000000 00000000 00000000 00000000
10000000 00000000 00000000 00000000 00000000 00000000
10000000 00000000 00000000 00000000 00000000 00000000
10000000 00000000 00000000 00000000 00000000 00000000
```

Integer

String

Language Structures (Python)

- **Variables and literals**
  - with some internal representation, e.g., Integers, Floats, Booleans, Strings, ...
    - In Python: Implicit data types!
- **Operations** on variable and literals of a type
  - e.g., +, *, /, //, **
  - ==, <, >, <=, >=
- **Expressions** are valid well-defined sets of operations on variables and literals that produce a value of a type.
  - x = 4 * 3

More Language Structures (Python)

- **Data type**: values, literals, operations, e.g., int, float, string
  - **Expression**: 3.1 * 2.6
  - **Call expression**: `max(0, x)`
  - **Variables**
  - **Assignment Statement**: `x = <expression>`
  - **Control Statement**: `if ... (see later)`
- **Sequences**: tuple, list
  - `numpy.array(<object>)`
- **Data structures**
  - `numpy.array`, Table
- **Tuple assignment**
  - `x, y = <expression>`

Call Expressions

- Evaluate a function on some arguments

What would be some useful functions?
- Built-in functions
  - `https://docs.python.org/3/library/functions.html`
  - `str`, `max`, `min`
  - `https://docs.python.org/3/library/
  - `import math; help(math)`

Defining Functions

- Generalizes an expression or set of statements to apply to lots of instances of the problem
- A function should *do one thing well*
Conditional statement

- Do some statements, conditional on a predicate expression

```python
if <predicate>:
    <true statements>
else:
    <false statements>
```

for statement – iteration control

- Repeat a block of statements for a structured sequence of variable bindings

```python
<initialization statements>
for <variables> in <sequence expression>:
    <body statements>
<rest of the program>
```

while statement – iteration control

- Repeat a block of statements until a predicate expression is satisfied

```python
<initialization statements>
while <predicate expression>:
    <body statements>
<rest of the program>
```

Data-driven iteration

- Describe an expression to perform on each item in a sequence
- Let the data dictate the control

```python
[ <expr with loop var> for <loop var> in <sequence expr> ]
```

By the Way...

- Could we build a computer that has no instructions, only data?

Yes! The One Instruction Set Computer.

Check it out:
https://en.wikipedia.org/wiki/One_instruction_set_computer