2.18: Introduction to Programming for Geoscientists

Lecture 1 - Computing with Formulas

October 2015

Mathematical Formulas

- Programs are sequences of instructions given to the computer.
- Can solve mathematical formulae.
- E.g. Given an initial velocity $v_0 = 5 \text{ms}^{-1}$ and the acceleration due to gravity $g = 9.8 \text{ms}^{-2}$, compute the position of a ball in vertical motion at time $t = 0.6$s using $y(t) = v_0 t - \frac{1}{2} g t^2$
- $y = 5 \cdot 0.6 - \frac{1}{2} \cdot 9.81 \cdot 0.6^2$
- ```
  y = 5*0.6 - 0.5*9.81*0.6**2
  print y
  ```
  ```

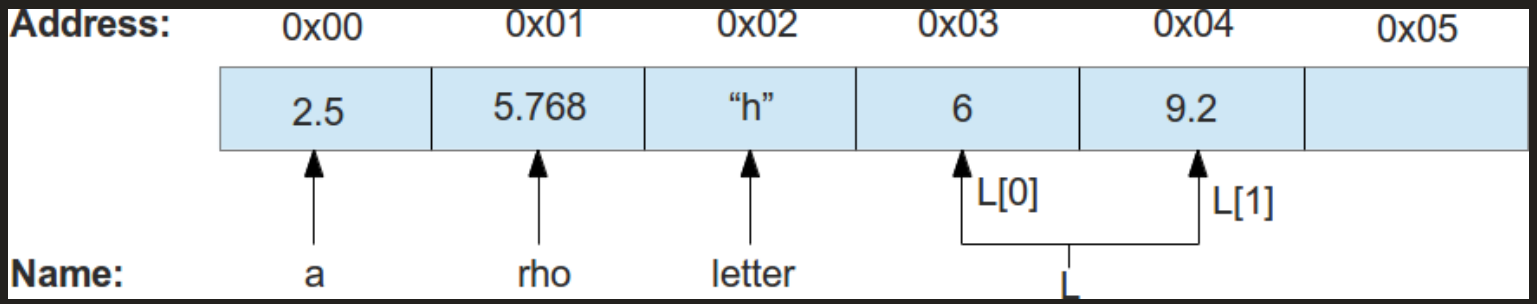
Mathematical Formulas

- Programming languages simpler than natural languages...
- ...but more pedantic.

Variables

- Variable: a place in the computer's memory which holds a value.
  - Memory address + name
  - You define the name in your Python program.
  - e.g. If variable `a` does not already exist, the statement `a = 5` stores the value 5 in an un-used block of memory.
  - The value can then be referenced (i.e. accessed) using the name, e.g. `print a`.

A simplified view of variables:



Variables

- Make sure variables are defined before trying to use them! The following will not work:

```
b = 5
c = a*b
a = 10
```

- Variable names:
  - are case sensitive.
  - cannot start with a digit.
  - cannot be a Python keyword: `and`, `as`, `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `in`, `is`, `lambda`, `not`, `or`, `pass`, `print`, `raise`, `return`, `try`, `with`, `while`, `yield`.

Comments

- Always comment code - for your benefit as well as others'.
- Describe what the key statements are doing.
- e.g. `y = v0*t - 0.5*g*t**2 # Calculate the vertical position`
- Ignored by the Python interpreter.

Printing

- Data held in variables can be printed to the screen using

```
b = 5.67560
print b
```

- Or, to present data in a nicer way, use printf style formatting:

```
print "The data held in variable b is: %.2f" % (b)
```

- The format specifier `%.2f` acts like a placeholder. When printing to the screen, Python substitutes this for the data in `b` and formats it accordingly:
  - `%.2f` prints out the data in `b` to 2 decimal places (i.e. `5.68`).
  - `%d` prints out the data in `b` as an integer (i.e. `5`).
  - `%g` prints out the data in `b` to the minimum number of significant figures (i.e. `5.6756`).
- If you see numbers like `5e-2`, this is 5×10^{-2}. Nothing to do with the mathematical constant e ≈ 2.71828.

Integer Division

- Dividing an integer by another integer will result in another integer.
- Python computes the result, and drops the decimal point and everything after it. e.g. 9/5 = 1.8 will evaluate to 1
- If in doubt, just make the numerator or denominator (or both) floating-point numbers. e.g. $3 \rightarrow 3.0$

Operator precedence

- Expressions like `2.0 + 3.0/5.0` are evaluated in a particular order, determined by operator precedence.
- Division has a higher precedence than addition, so `3.0/5.0` is evaluated first, and `2.0` is then added on afterwards.
- If we wanted `2.0 + 3.0` to be evaluated first, then we need to use parentheses: `(2.0 + 3.0)/5.0`.
- BODMAS: Brackets, Order, Division, Multiplication, Addition, Subtraction.
- Note: Python groups certain operators together such that they have the same precedence, and then evaluates expressions from left to right. See `http://docs.python.org/2/reference/expressions.html`.

Importing modules

- Use modules to split your code up to make it more manageable, or make a piece of code available to other programs.
- Mathematical functions like sin(x), cos(x), log(x) are in the `math` module.
- Two ways of importing functions from modules:
  - `import math`: imports all functions in the math module, but keeps functions in their own separate namespace. That is, you must prepend `math.` to the function's name to use it, e.g. `x = 0.5; y = math.sin(x)`
  - `from math import *`: Python will import all the functions in the math module into the current namespace. That is, you can simply do `x = 0.5; y = sin(x)`. But: be careful that you do not have another function named `sin` in your program!