W4260: Modeling the Universe

Greg Bryan
Uses of Computers in Astronomy and Astrophysics

• Data analysis
  – E.g. image reduction, detectors, statistical analysis (correlations)

• Simulation
  – E.g. stellar evolution, galaxy collision, cosmological structure formation
An overview of computing languages

- **Hardware**
  - "assembly" language
- **“assembly” language**
- **General programming languages**
  - QBasic, C, Pascal
- **Specialized analysis packages**
  - Maple, Mathematica, IDL, MATLAB

Increasing complexity
What is a program?
What is a program?
What is a program?
What is a program?

1. lather
2. rinse
3. if necessary repeat
What is a program?

C program

```c
main()
{
    printf(“hello world\n”);
}
```

PYTHON program

```python
print “hello world”
```
What is a program?

C program

```c
main()
{
    float x, y;
    x=0.1;
    while (x < 2.0) {
        y = x*x;
        printf("%f %f\n", x, y);
        x = x + 0.1;
    }
}
```

PYTHON program

```python
x=0.1
While x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```
1. Data storage

- variables store data
- different types (integer, real, text, …)

```c
float x, y;
x = 0.1;
while (x < 2.0) {
y = x*x;
    printf("%f %f\n", x, y);
    x = x + 0.1;
}
```

```python
x = 0.1
While x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```
2. Flow of control: Loops

- allows for repeated actions

While (this is true)
[do this]

C program

```c
main()
{
    float x, y;

    x=0.1;
    while (x < 2.0) {
        y = x*x;
        printf("%f %f\n", x, y);
        x = x + 0.1;
    }
}
```

PYTHON program

```python
x=0.1
while x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```
3. (Arithmetic) Operations

- elementary maths (+, -, multiply, divide)
- other operations (more later)

```
x = 0.1
square
y = 0.01
```

**C program**

```c
main()
{
    float x, y;

    x=0.1;
    while (x < 2.0) {
        y = x*x;
        printf("%f %f\n", x, y);
        x = x + 0.1;
    }
}
```

**PYTHON program**

```python
x=0.1
while x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```
4. Output (& Input)

- write out results to screen (or file)

C program

```c
main()
{
    float x, y;
    x=0.1;
    while (x < 2.0) {
        y = x*x;
        printf("%f %fn", x, y);
        x = x + 0.1;
    }
}
```

PYTHON program

```python
x=0.1
while x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```

result:

```
0.1  0.01
```
Another arithmetic operation

C program

```c
main()
{
    float x, y;
    x=0.1;
    while (x < 2.0) {
        y = x*x;
        printf("%f %f\n", x, y);
        x = x + 0.1;
    }
}
```

PYTHON program

```python
x=0.1
while x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```

Read: x becomes x+0.1
5. Relational operator

while (x < 2.0)
while (0.2 < 2.0)
while (true)

So loop continues

C program

```c
main()
{
    float x, y;
    x=0.1;
    while (x < 2.0) {
        y = x*x;
        printf("%f %f\n", x, y);
        x = x + 0.1;
    }
}
```

PYTHON program

```python
x=0.1
while x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```
Final result

C program

```c
main()
{
    float x, y;

    x=0.1;
    while (x < 2.0) {
        y = x*x;
        printf("%f %f\n", x, y);
        x = x + 0.1;
    }
}
```

PYTHON program

```python
x=0.1
While x < 2.0:
    y = x*x
    print x, y
    x = x + 0.1
```

result:

```
0.1 0.01
0.2 0.04
0.3 0.09
0.4 0.16
...```
Introduction to Programming

Part I
Overview

• Storing Data
  – Basic types
  – Arrays

• Controlling the flow of execution
  – Loops (for, while)
  – If…then…else
Variables

• Name
  – e.g. x, y, index, NSync, Nigela6

• Types
  – Integer (-2147483648 to +2147483647)
  – Float (6 digits, 10^-38 to 10^+38)
  – Double (15 digits, 10^-308 to 10^+308)
  – Boolean (true or false)
  – Character or string
Variables

- Integer (e.g. 2, -1056)
- Float (e.g. 2.997925e10)
- Double (e.g. 4.8372393473732e-128)
- Boolean (true or false)
- Character or string
- Others
  - complex, collections, dictionaries, etc
Variables

- Integer (e.g. 2, -1056)
- Float (e.g. 2.997925e10)
- Double (e.g. 4.8372393473732e-128)
- Boolean (true=1 or false=0)
- Character or string
- Others
  - complex, collections, dictionaries, etc

In C/Python, no true Boolean type: instead the integer type is used
Variables

• Integer (e.g. 2, -1056)
• Float (e.g. 2.997925e10)
• Double (e.g. 4.8372393473732e-128)
• Boolean (true or false)
• Character or string
• Others
  – complex, collections, objects

Won’t cover this very much, at least to start with.
Data representation: integers

- All numbers represented in base\(_2\) (binary)
- Base 10: 0-9
  
  - \(438 = 4 \times 100 + 3 \times 10 + 8\)
  
  10\(^2\) digit ↑ ↓ 1 digit

- Base 2: 0-1
  
  - \(10 = 1 \times 2 + 0\) (= 2)
  
  2\(^4\) digit ↑ ↓ 1 digit
  
  - \(10111 = 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 1\) (=23)
Representing real numbers

• How to represent numbers like $3.06 \times 10^{26}$?
• Answer: binary fractions
• Base 10:
  – $3.14 = 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2}$
• Base 2:
  – $1.01 = 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$
  – $\quad = 1 \times 1 + 0/2 + 1/4 = (1.25 \text{ in base}_{10})$
• Numbers have mantissa and exponent
Representing real numbers

• Limited precision (float: 32 bits, double: 64 bits)
• Some decimal numbers cannot be represented exactly

C program

```c
main()
{
    double x, y;
    x = 1.25;
    y = 1.26;
    printf("x=%.20f\n", x);
    printf("y=%.20f\n", y);
}
```

Python program

```python
x = 1.25
y = 1.26
print "x=%.20f" % x
print "y=%.20f" % y
```

result

|x = 1.25000000000000000000 |
|y = 1.26000000000000000888 |

Problems start near the 16th digit
Limited precision: calculating $\pi$

- This limited precision has an impact on algorithms

\[
\pi = 2\sqrt{3}\left(1 - \frac{1}{3 \times 3} + \frac{1}{5 \times 3^2} - \frac{1}{7 \times 3^3} + \ldots\right)
\]

C program

```c
#include <math.h>

main()
{
    double sum, pi;
    int i;
    sum = 1.0;
    for (i = 1; i < 30; i = i + 1)
        sum = sum+1.0/((i*2+1)*pow(-3.0,i));
    pi = 2.0*sqrt(3.0)*sum;
    printf("%.20f\n", 2.0*sqrt(3.0)*sum);
}
```

Python program

```python
from math import *

sum = 1.0
for i in range(1,30):
    sum = sum+1.0/((i*2+1)*(-3.0)**i)
pi = 2.0*sqrt(3.0)*sum
print "%.20f\n" % pi
```

result: $\approx 3.1415926535897936009$  Wrong!
Data representation blues

- Operations with “real” numbers on computers are (usually) inaccurate
  - Algorithmic changes
  - (e.g. Financial calculations)