Capítulo 1: Programando com Fórmulas

Adaptado e traduzido para o português por **Alisson Brito**, Laboratório de Engenharia de Sistemas e Robótica (LASER), da Universidade Federal da Paraíba (UFPB)

Material original: **Hans Petter Langtangen**, Simula Research Laboratory and University of Oslo, Dept. of Informatics

Data: **Junho, 2020**

Por que Programação Científica?

- Programação é uma linguagem universal
  - O computador é um máquina de resolver problemas matemáticos
  - A linguagem de programação é o seu formalismo
  - Não há mal entendidos

> *Everybody in this country should learn how to program a computer... because it teaches you how to think*. Steve Jobs, 1955-2011.

Por que Python?

- Simples de instalar, usar e aprender
- Documentação vasta
- Muitas bibliotecas
  - Processamento de imagens, inteligência artificial, jogos, cálculos matemáticos
- Reuniu o que há de melhor em diversas linguagens
- Criada para ser uma linguagem científica:
  - Lida muito com bem equações, fórmulas, notações literais, gráficos
  - Trabalha com números grandes, números racionais, números complexos
  - Otimização em hardware para operações matemáticas

Capítulo 1 é sobre resolução de fórmulas

Por quê?

- Todos entendem o problema quando é modelado por equações
- Muitos conteitos fundamentais de programação serão introduzidos:
  - variáveis
  - expressões aritméticas
  - objetos
  - formatando e imprimindo números e texto

Resolvendo uma fórmula matemática

Altura de uma bola em movimento vertical

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

onde

- y é a altura (posição) da bola em função do tempo t
- v_0 é a velocidade inicial em $t = 0$
- g é a aceleração da gravidade

Tarefa: dados v_0, g e t, calcule y.

Não seria mais melhor usar logo uma calculadora?!

Por que escrever um programa?

- Usar programação pode ampliar as possibilidades:
  - Reutilizar os cálculos feitos em outras operações
    - criar funções, talvez uma biblioteca
  - Executar a mesma operação milhares (ou até milhões de vezes)
  - Exibir gráficos e análises mais complexas
  - É fácil compartilhar soluções se elas estiverem escritas em linguagem de programação (bem escritas!)
- Vamos aprender um pouco disto tudo neste curso!

Nosso primeiro programa:

Calcular $y(t) = v_0 t - \frac{1}{2} g t^2$ para $v_0 = 5$, $g = 9.81$ e $t = 0.6$:

$$y = 5 \cdot 0.6 - \frac{1}{2} \cdot 9.81 \cdot 0.6^2$$

O programa em Python:

In [36]:

```
y = 5*0.6 - 0.5*9.81*0.6**2
print y
```

```
4.2342
```

Como escrever e executar um programa

- Um programa é um *texto plano* escrito em editor simples
- Use algum editor ou IDE específico para programação: Sublime, Visual Studio, Gedit, Emacs, Vim, Spyder, or IDLE (*nunca* Microsoft Word!)
- Ou use o Jupyter para deixar seus resultados documentados (este que estou utilizando)

Passo 1. Escreva o programa no seu editor:

In [ ]:

```
print 5*0.6 - 0.5*9.81*0.6**2
```

Passo 2. Salve o programa em um arquivo, ex: `ball1.py` . (`.py` é a extensão para programas Python.)

Passo 3. Abra o terminal e vá para a pasta onde o seu programa foi salvo

Passo 4. Execute o programa:

```
Terminal> python ball1.py
```

Armazene os valores em variáveis para tornar o programa mais legível

Variáveis em programação são uma analogia às variáveis da matemática, por exemplo:

$$v_0 = 5, \quad g = 9.81, \quad t = 0.6, \quad y = v_0 t - \frac{1}{2} g t^2$$

Podemos usar variáveis em nosso programa para torná-lo mais simples de entender:

In [37]:

```
v0 = 5.0
g = 9.81
t = 0.6
y = v0*t - 0.5*g*t**2
print "y = ", y
```

```
y =  1.2342
```

Há regras para os nomes das variáveis

- O nome da variável pode conter letras a-z, A-Z, underscore `_` e os digitos 0-9, mas não pode iniciar números
- Variáveis são case-sentive, ou seja, `a` é diferente de `A`)

In [ ]:

```
initial_velocity = 5
accel_of_gravity = 9.81
TIME = 0.6
VerticalPositionOfBall = initial_velocity*TIME - \
                         0.5*accel_of_gravity*TIME**2
print VerticalPositionOfBall
```

(Observe: a barra invertida () permite a instrução continuar na linha seguinte

Bons nomes de variáveis ajudam na compreensão do programa

Algumas palavras são reservadas em Python

Por exemplo: `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` , e `yield` .

Comentários são importantes para deixar observações e esclarecimentos

Programa com comentários:

In [38]:

```
# program for computing the height of a ball
# in vertical position
v0 = 5    # initial velocity
g = 9.81  # acceleration of gravity
t = 0.6   # time
y = v0*t - 0.5*g*t**2  # vertical position
print y
```

```
1.2342
```

A sintaxe do printf traz muita flexibilidade na formatação dos textos e números

É muito comum precisarmos imprimir os resultados juntos com textos, por exemplo:

```
Em t=0.6 seg., y é 1.23m.
```

In [41]:

```
# program for computing the height of a ball
# in vertical prática
v0 = 5    # initial velocity
g = 9.81  # acceleration of gravity
t = 0.6   # time
y = v0*t - 0.5*g*t**2  # vertical position

print 'Em t=%g segundos, y é %.2f metros' % (t, y)
```

```
Em t=0.6 segundos, y é 1.23 metros
```

Na sintaxe printf colocarmos o local dos valores e seus formatos e depois as variáveis na sequência: `%g` ← `t` , `%.2f` ← `y`

Exemplos de diferentes formatos

```
%g          most compact formatting of a real number
%f          decimal notation (-34.674)
%10.3f      decimal notation, 3 decimals, field width 10
%.3f        decimal notation, 3 decimals, minimum width
%e or %E    scientific notation (1.42e-02 or 1.42E-02)
%9.2e       scientific notation, 2 decimals, field width 9
%d          integer
%5d         integer in a field of width 5 characters
%s          string (text)
%-20s       string, field width 20, left-adjusted
%%          the percentage sign % itself
```

Imprimindo textos mais longos

Aspas 'triplas' (""") podem ser usadas para imprimir textos maiores em várias linhas, e podemos combinar isso com o formato printf:

In [45]:

```
v0 = 0.4123123
g = 9.81
t = 0.6
y = v0*t - 0.5*g*t**2

print """
Em t=%.1f seg., uma bola com
velocidade inicial v0=%.3E m/s
esta localizada na altura %.2f m.
""" % (t, v0, y)
```

```
Em t=0.6 seg., uma bola com
velocidade inicial v0=4.123E-01 m/s
esta localizada na altura -1.52 m.
```

Expressões aritméticas são avaliadas da mesma forma que na matemática

- Exemplo: $\frac{5}{9} + 2a^4/2$, em Python se escreve como `5.0/9 + 2*a**4/2`
- Devemos usar parentesis para determinar explicitamente uma ordem diferente de avaliação: `(5.0/9) + (2*(a**4))/2`

Algumas funções básicas das matemática são encontradas no módulo `math`

- Para computar $\sin x$, $\cos x$, $\ln x$, etc. usamos o módulo `math`
- Há diversos módulos disponíveis em Python. Alguns precisam ser instalados (veremos depois como isso pode ser feito) Computar $\sqrt{2}$ usando a função `sqrt` do módulo `math` :

In [ ]:

```
import math
r = math.sqrt(2)
# or
from math import sqrt
r = sqrt(2)
# or
from math import *   # import everything in math
r = sqrt(2)
```

Outro exemplo utilizando `math`

Calcule

$$Q = \sin x \cos x + 4 \log x$$

para $x = 1.2$.

In [47]:

```
from math import sin, cos, log
x = 1.2
Q = sin(x)*cos(x) + 4*log(x)   # log is ln (base e)
print Q
```

```
1.06701781745
```

Computadores podem gerar resultados inexatos devido a erros de arredondamento

Vamos calcular $1/49 \cdot 49$ e $1/51 \cdot 51$:

In [49]:

```
v1 = 1/49.0 * 49

v2 = 1/51.0 * 51
print '%.17f %.17f' % (v1, v2)
```

```
0.99999999999999989 1.00000000000000000
```

Obs:

- Muitos números reais são representados de forma inexata pelo computador (17 digitos)
- Nem 1/49, nem 1/51 são representados de forma exata, o erro típico está na casa de 10^{-16}
- As vezes pequenos erros se propagam até o resultado final.
- As vezes o erro é acumulado e replicado a diversas operações
- Lição aprendida: resultados calculados por computadores envolvendo números reais são apenas aproximações

Teste este cálculo

O que foi impresso?

In [51]:

```
a = 0.1 ; b = 0.2;
computed = a + b
expected = 0.3
correct = (computed == expected)
print 'Correct:', correct
```

```
Correct: False
```

Agora mude para `a = 0.1` e `b = 0.2` (`expected = 0.3`). O que foi impresso? Por quê?

Igualdades devem ser usadas com tolerância!

Vejamos em mais detalhes:

In [52]:

```
a = 0.1; b = 0.2; expected = 0.3
a + b == expected
```

Out[52]:

```
False
```

In [53]:

```
print '%.17f\n%.17f\n%.17f\n%.17f' % (0.1, 0.2, 0.1 + 0.2, 0.3)
```

```
0.10000000000000001
0.20000000000000001
0.30000000000000004
0.29999999999999999
```

Vejamos outro exemplo utilizando funções matemáticas

A função $\sinh x$ é definida como

$$\sinh(x) = \frac{1}{2}\left(e^x - e^{-x}\right)$$

Nós podemos computar esta função de três formas:

1. `math.sinh`
2. combinação de duas exponenciais: `math.exp`
3. combinação de duas potências: `math.e`

In [54]:

```
from math import sinh, exp, e, pi
x = 2*pi
r1 = sinh(x)
r2 = 0.5*(exp(x) - exp(-x))
r3 = 0.5*(e**x - e**(-x))
print '%.16f %.16f %.16f' % (r1,r2,r3)
```

```
267.7448940410164369 267.7448940410164369 267.7448940410163232
```

Resultado: `r1` = 267.7448940410164369, `r2` = 267.7448940410164369, `r3` = 267.7448940410163232 (!)

Exercício

Uma bola é lançada com velocidade v_0 em um ângulo θ com o eixo horizontal, a partir do ponto $(x, y = y_0)$. A trajetória da bola é uma parábola (desconsiderando a resitência do ar):

$$y = x \tan\theta - \frac{1}{2v_0}\frac{gx^2}{\cos^2\theta} + y_0$$

- Faça um programa que retorne a posição horizontal (y) da bola, considerando:
  - v_0 = 15 km/h, $theta$ = 60 graus, x = 0.5m, y_0 = 1m
  - Temos x, y e y_0 em m, $g = 9.81\text{m/s}^2$, v_0 em km/h e θ em graus. Será necessária a conversão de v_0 para m/s e θ para radiano

Solução

Programa:

In [ ]:

```
g = 9.81     # m/s**2
v0 = 15      # km/h
theta = 60   # degrees
x = 0.5      # m
y0 = 1       # m

print """v0    = %.1f km/h
theta = %d degrees
y0    = %.1f m
x     = %.1f m""" % (v0, theta, y0, x)

# convert v0 to m/s and theta to radians:
v0 = v0/3.6
from math import pi, tan, cos
theta = theta*pi/180

y = x*tan(theta) - 1/(2*v0)*g*x**2/((cos(theta))**2) + y0

print 'y     = %.1f m' % y
```

In [ ]:

In [ ]:

In [ ]: