Introducción a la programación

Usando Python

G. Sebastián Pedersen

Instituto de Industria
Universidad Nacional de General Sarmiento

Matemática para Economistas III, 1er. cuat. 2019

`https://sebasped.github.io/pythonungs/`

Máximas de la programación que vimos hasta ahora

- Reutilización de valores:

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.
- Duplicación de información:

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.
- Duplicación de información: siempre que sea posible *no duplicar información* en el código.

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.
- Duplicación de información: siempre que sea posible *no duplicar información* en el código.
  - Guardar el valor que se repite en una variable, y utilizar la variable en lugar del valor.

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.
- Duplicación de información: siempre que sea posible *no duplicar información* en el código.
  - Guardar el valor que se repite en una variable, y utilizar la variable en lugar del valor.
  - Con las líneas de código que se repiten crear una función, y utilizarla en lugar de andar repitiendo el código.

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.
- Duplicación de información: siempre que sea posible *no duplicar información* en el código.
  - Guardar el valor que se repite en una variable, y utilizar la variable en lugar del valor.
  - Con las líneas de código que se repiten crear una función, y utilizarla en lugar de andar repitiendo el código.
- Particionar y encapsular el problema: siempre que sea posible *particionar* el problema y *encapsularlo* en problemas menos complejos.

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.
- Duplicación de información: siempre que sea posible *no duplicar información* en el código.
  - Guardar el valor que se repite en una variable, y utilizar la variable en lugar del valor.
  - Con las líneas de código que se repiten crear una función, y utilizarla en lugar de andar repitiendo el código.
- Particionar y encapsular el problema: siempre que sea posible *particionar* el problema y *encapsularlo* en problemas menos complejos.
  - Mediante funciones.

Máximas de la programación que vimos hasta ahora

- Reutilización de valores: cualquier *valor que quiera ser reutilizado* (para un cálculo posterior, para una salida, etc.) *debe ser almacenado previamente en una variable*.
- Duplicación de información: siempre que sea posible *no duplicar información* en el código.
  - Guardar el valor que se repite en una variable, y utilizar la variable en lugar del valor.
  - Con las líneas de código que se repiten crear una función, y utilizarla en lugar de andar repitiendo el código.
- Particionar y encapsular el problema: siempre que sea posible *particionar* el problema y *encapsularlo* en problemas menos complejos.
  - Mediante funciones.
  - Mediante scripts.

¡Nuestro primer programa para resolver ODEs! I


```
#para resolver ecuaciones diferenciales
from sympy import *
# para generar una lista de valores equiespaciados
from numpy import linspace

# defino los nombres de la variable y la función
t = symbols('t') # la variable independiente
y = symbols('y', cls=Function) # la función

# defino la ecuación diferencial
# Eq es la función de Python para definidir ecuaciones diferenciales. La coma e
# y(t).diff(t) es la derivada primera
# y(t).diff(t,t) es la derivada segunda
# defino  y'' + 6y' + 9y = 45  (cambiar a gusto)
ecDif = Eq( y(t).diff(t,t) + 6*y(t).diff(t) + 9*y(t), 45 )

# defino condiciones iniciales y(0)=1 e y'(0)=2  (cambiar a gusto)
condInic = {y(0): 1, y(t).diff(t).subs(t, 0): 2}
```

¡Nuestro primer programa para resolver ODEs! II

```
# resuelvo la ecuación diferencial
# dsolve es la función de Python para calcular la ec. dif.
# ecDif es la ecuación a calcular
# y(t) son la variable y función a resolver
# ics son las condiciones iniciales
sol = dsolve( ecDif, y(t), ics=condInic )

#Muestro la ecuación y el resultado
print 'La ecuación diferencial:'
print ecDif
print condInic
print "\n" #agrega un renglón en blanco
print 'La solución:'
print sol
print "\n" #agrega un renglón en blanco
```

```
La ecuación diferencial:
Eq(9*y(t) + 6*Derivative(y(t), t) + Derivative(y(t), (t, 2)), 45)
{y(0): 1, Subs(Derivative(y(t), t), (t,), (0,)): 2}


La solución:
Eq(y(t), (-10*t - 4)*exp(-3*t) + 5)
```

Pequeña mejora al mostrar la EDO y la solución

```
#Muestro la ecuación y el resultado de forma más legible

#para resolver ecuaciones diferenciales
from sympy import *
#para imprimir fórmulas matemáticas
init_printing()
from IPython.display import display

#muestro la ode y la solución
print 'La ecuación diferencial:'
display(ecDif)
display(condInic)
print "\n" #agrega un renglón en blanco
print 'La solución:'
display(sol)
```

$$9y(t) + 6\frac{d}{dt}y(t) + \frac{d^2}{dt^2}y(t) = 45$$

$$\left\{y(0) : 1, \quad \frac{d}{dt}y(t)\Big|_{t=0} : 2\right\}$$

La solución:

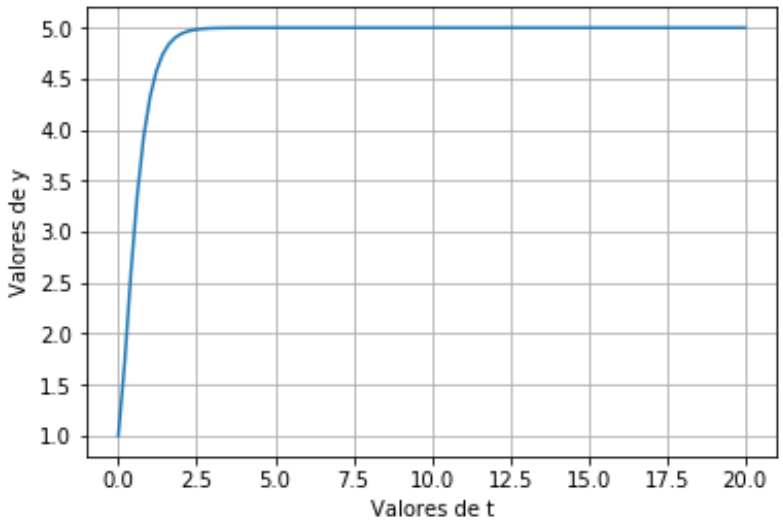
$$y(t) = (-10t - 4)\, e^{-3t} + 5$$

También graficando la solución

```
# para graficar
from matplotlib import pyplot as plt

f = sol.args[1] # me quedo solamente con la fórmula de la solución
lam = lambdify(t, f, modules=['numpy']) #hago la fórmula evaluable
t_vals = linspace(0, 20, 100) #valores de t a usar para evaluar: inicio, fin, p
y_vals = lam(t_vals) #evalúo la fórmula en los valores de t

#grafico la solución
plt.plot(t_vals, y_vals) #hago el gráfico
plt.grid() #que me ponga un grillado en el gráfico
plt.xlabel('Valores de t') #leyende del eje horizontal
plt.ylabel('Valores de y') #leyenda del eje vertical
plt.show() #muestro el gráfico
```



To code or not to code...

Tener presente las máximas de la programación (ver diapo "Máximas de la programación que vimos hasta ahora").

Ejercicios:

1. *Utilizar* nuestro primer programa que resuelve EDOs para resolver algunas ecuaciones diferenciales de la práctica.
2. *Modificar* nuestro primer programa que resuelve EDOs para muestre la ecuación y la solución de forma más legible (ver diapo "Pequeña mejora al mostrar la EDO y la solución").
3. *Modificar* el programa del punto 2. para que también grafique la solución de la EDO (ver diapo "También graficando la solución").
4. ¡Bonus Hacker! 😎: *Modificar* el primer programa que resuelve EDOs para que resuelva sistemas con 2 ecuaciones diferenciales (aunque sea algún caso básico).