Rudimentos de `turtle`
Módulo de Python

Héctor Manuel Mora Escobar

Universidad Nacional
Bogotá
hectormora@yahoo.com
www.hectormora.info

septiembre de 2014

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. El nombre proviene del grupo de cómicos ingleses “Monty Python”.

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. El nombre proviene del grupo de cómicos ingleses "Monty Python".

- Gratuito.

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. El nombre proviene del grupo de cómicos ingleses "Monty Python".

- Gratuito.
- Multiplataforma (Windows, Linux, Unix, Mac, ...).

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. El nombre proviene del grupo de cómicos ingleses "Monty Python".

- Gratuito.
- Multiplataforma (Windows, Linux, Unix, Mac, ...).
- Lenguaje interpretado o de "scripts" o guiones. Los lenguajes compilados tienen una ejecución más rápida, los interpretados son más flexibles y más portables. Realmente Python es semiinterpretado, se puede obtener un pseudocódigo de máquina llamado "bytecode".

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. El nombre proviene del grupo de cómicos ingleses "Monty Python".

- Gratuito.
- Multiplataforma (Windows, Linux, Unix, Mac, ...).
- Lenguaje interpretado o de "scripts" o guiones. Los lenguajes compilados tienen una ejecución más rápida, los interpretados son más flexibles y más portables. Realmente Python es semiinterpretado, se puede obtener un pseudocódigo de máquina llamado "bytecode".
- Tipado dinámico, no es necesario declarar el tipo de las variables, Python escoge la manera más adecuada.

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. El nombre proviene del grupo de cómicos ingleses "Monty Python".

- Gratuito.
- Multiplataforma (Windows, Linux, Unix, Mac, ...).
- Lenguaje interpretado o de "scripts" o guiones. Los lenguajes compilados tienen una ejecución más rápida, los interpretados son más flexibles y más portables. Realmente Python es semiinterpretado, se puede obtener un pseudocódigo de máquina llamado "bytecode".
- Tipado dinámico, no es necesario declarar el tipo de las variables, Python escoge la manera más adecuada.
- Orientado a objetos.

Descarga, instalación y uso

- **Descarga**:

  `www.python.org`

  ▷ DOWNLOAD

  ▷ Python 2.7.2 Windows Installer (Windows binary – does not include source)

Descarga, instalación y uso

- **Descarga**:

  `www.python.org`

  ▷ DOWNLOAD

  ▷ Python 2.7.2 Windows Installer (Windows binary – does not include source)
- **Instalación**:

  Activar el archivo descargado: `python-2.7.2.msi`

Descarga, instalación y uso

- **Descarga**:

  `www.python.org`

  ▷ DOWNLOAD

  ▷ Python 2.7.2 Windows Installer (Windows binary – does not include source)
- **Instalación**:

  Activar el archivo descargado: `python-2.7.2.msi`
- **Uso**:

  ▷ Inicio

  ▷ Todos los programas

  ▷ Python 2.7

  ▷ IDLE (Python GUI)

Primeros pasos en el interpretador de Python

Al empezar en Python, aparece una ventana con algo semejante a:

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Primeros pasos en el interpretador de Python

Al empezar en Python, aparece una ventana con algo semejante a:

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

>>> es el "prompt" (perdón por el anglicismo) de Python. Al frente de él se escriben las órdenes en el interpretador. Obviamente, al acabar cada orden, se oprime la tecla Enter.

Primeros pasos en el interpretador de Python

Al empezar en Python, aparece una ventana con algo semejante a:

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

>>> es el "prompt" (perdón por el anglicismo) de Python. Al frente de él se escriben las órdenes en el interpretador. Obviamente, al acabar cada orden, se oprime la tecla Enter.

```
>>> 25/4
```

Primeros pasos en el interpretador de Python

Al empezar en Python, aparece una ventana con algo semejante a:

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

>>> es el "prompt" (perdón por el anglicismo) de Python. Al frente de él se escriben las órdenes en el interpretador. Obviamente, al acabar cada orden, se oprime la tecla Enter.

```
>>> 25/4
```

```
>>> 25.0/4
```

Primeros pasos en el interpretador de Python

Al empezar en Python, aparece una ventana con algo semejante a:

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

>>> es el "prompt" (perdón por el anglicismo) de Python. Al frente de él se escriben las órdenes en el interpretador. Obviamente, al acabar cada orden, se oprime la tecla Enter.

```
>>> 25/4
```

```
>>> 25.0/4
```

Para salir de Python:

```
>>> quit()
```

Un programita con el editor

- File
- New Window

Escribir

```
n = 6
f = 1
for i in range(2, n+1):
    f = f*i
print 'factorial = ', f
```

- File
- Save

Nombre: `ejem01.py`

- Run
- Run Module F5

Otro programita

En otro archivo:

```
# raices de la ecuacion  a x^2 + b x + c
import math
a = 1
b = 5
c = 6
d = b*b - 4*a*c
a2 = 2*a
if d >= 0:
    t = math.sqrt(d)
    r1 = (-b + t )/a2
    r2 = (-b - t )/a2
    print 'raices: ', r1, r2
else:
    print 'No hay raices reales.'
```

El módulo turtle

Para cargar turtle se escribe

```
>>> from turtle import *
```

No aparece nada raro, simplemente Python está listo para recibir órdenes de turtle.

El módulo turtle

Para cargar turtle se escribe

```
>>> from turtle import *
```

No aparece nada raro, simplemente Python está listo para recibir órdenes de turtle.

```
>>> forward(100)
```

El módulo turtle

Para cargar turtle se escribe

```
>>> from turtle import *
```

No aparece nada raro, simplemente Python está listo para recibir órdenes de turtle.

```
>>> forward(100)
```

Se abre una nueva ventana, en ella la tortuga ha dibujado un segmento de recta cuya longitud es 100 pixeles.

El módulo turtle

Para cargar turtle se escribe

```
>>> from turtle import *
```

No aparece nada raro, simplemente Python está listo para recibir órdenes de turtle.

```
>>> forward(100)
```

Se abre una nueva ventana, en ella la tortuga ha dibujado un segmento de recta cuya longitud es 100 pixeles.

```
>>> right(90)
```

El módulo turtle

Para cargar turtle se escribe

```
>>> from turtle import *
```

No aparece nada raro, simplemente Python está listo para recibir órdenes de turtle.

```
>>> forward(100)
```

Se abre una nueva ventana, en ella la tortuga ha dibujado un segmento de recta cuya longitud es 100 pixeles.

```
>>> right(90)
```

La punta de la flecha, giró hacia la derecha 90 grados.

El módulo turtle

Para cargar turtle se escribe

```
>>> from turtle import *
```

No aparece nada raro, simplemente Python está listo para recibir órdenes de turtle.

```
>>> forward(100)
```

Se abre una nueva ventana, en ella la tortuga ha dibujado un segmento de recta cuya longitud es 100 pixeles.

```
>>> right(90)
```

La punta de la flecha, giró hacia la derecha 90 grados.

```
>>> forward(200)
```

Otras órdenes

Otras órdenes

```
>>> left(135)
```

Otras órdenes

```
>>> left(135)
>>> reset()
```

Otras órdenes

```
>>> left(135)
>>> reset()
>>> up()
```

Otras órdenes

```
>>> left(135)
>>> reset()
>>> up()
>>> back(150)
```

Otras órdenes

```
>>> left(135)
>>> reset()
>>> up()
>>> back(150)
>>> down()
```

Otras órdenes

```
>>> left(135)
>>> reset()
>>> up()
>>> back(150)
>>> down()
>>> color('red')
```

Otras órdenes

```
>>> left(135)
>>> reset()
>>> up()
>>> back(150)
>>> down()
>>> color('red')
>>> width(3)
```

Otras órdenes

```
>>> left(135)
>>> reset()
>>> up()
>>> back(150)
>>> down()
>>> color('red')
>>> width(3)
>>> goto(-20,200)
```

Otras órdenes

```
>>> left(135)
>>> reset()
>>> up()
>>> back(150)
>>> down()
>>> color('red')
>>> width(3)
>>> goto(-20,200)
>>> speed(3)
...
```

El primer programa

Utilizar el editor del ambiente Python:

▷ *File* ▷ *New Window*

Enseguida, en la pantalla blanca que aparece, escriba el programa, por ejemplo,

```
# primer programita de turtle
# 8 de julio 2011, Hector Mora
from turtle import *
forward(100)
```

y lo guarda con extensión `.py`, por ejemplo con nombre `ej001.py`.
Una vez escrito y guardado lo ejecuta picando en la ventana del editor

▷ *Run* ▷ *Run Module*

o, simplemente, mediante la tecla F5.
Así en el ambiente Python aparecerán los resultados del programa (o los errores del programa).

Un cuadrado

Otro archivo, `ej002.py`

```
# un cuadrado
from turtle import *

a = 200
forward(a)
right(90)
forward(a)
right(90)
forward(a)
right(90)
forward(a)
right(90)
```

while

```
# un cuadrado
# escritura mas corta, while
from turtle import *

a = 200
i = 1
while i <= 4:
    forward(a)
    right(90)
    i = i+1
```

Importante:

- while

while

```
# un cuadrado
# escritura mas corta, while
from turtle import *

a = 200
i = 1
while i <= 4:
    forward(a)
    right(90)
    i = i+1
```

Importante:

- `while`
- `:` los dos puntos.

while

```
# un cuadrado
# escritura mas corta, while
from turtle import *

a = 200
i = 1
while i <= 4:
    forward(a)
    right(90)
    i = i+1
```

Importante:

- `while`
- `:` los dos puntos.
- **La sangría** ("indentación"). Con espaciador y no con tabulador.

Ayuda

```
>>> help()
```

Aparece

```
help>
```

Digitar

```
help> turtle
```

Para salir de la ayuda de turtle (vuelve a `help>`)

```
q
```

Digitar

```
help> while
```

Para salir `q`

Para salir del módulo de ayuda:

```
help> quit
```

Vuelve al prompt `>>>`

Con una función

```
# con una funcion
from turtle import *

def cuadrado1(x):
    i = 1
    while i <= 4:
        forward(x)
        right(90)
        i = i+1

a = 200
reset()
cuadrado1(a)
up()
goto(30,30)
down()
cuadrado1(a)
up()
goto(60,60)
down()
cuadrado1(a)
```

Una función con más parámetros

```
from turtle import *
def cuadrado(dir_ini, der_izq, lado, tono):
    # dibuja un cuadrado
    # dir_ini  :  direccion inicial en  grados
    # derecha o izquierda:  'd'  o  'i'
    # lado : medida
    # tono: 'red', 'blue', 'green', ...
    setheading(dir_ini)
    color(tono)
    i = 1
    while i <= 4:
        forward(lado)
        if der_izq == 'd':
            right(90)
        else:
            left(90)
        i = i+1
```

Continuación

```
reset()
speed(50)
width(3)
a = 100
t = 0
while t <= 360:
    cuadrado(t, 'i', a, 'red')
    t = t+10
```

Funciones de movimiento

```
Move and draw
    forward() | fd()
    backward() | bk() | back()
    right() | rt()
    left() | lt()
    goto() | setpos() | setposition()
    setx()
    sety()
    setheading() | seth()
    home()
    circle()
    dot()
    stamp()
    clearstamp()
    clearstamps()
    undo()
    speed()
Tell Turtle's state
    position() | pos()
    towards()
    xcor()
    ycor()
    heading()
    distance()
Setting and measurement
    degrees()
    radians()
```

Lápiz

```
Drawing state
    pendown() | pd() | down()
    penup() | pu() | up()
    pensize() | width()
    pen()
    isdown()
Color control
    color()
    pencolor()
    fillcolor()
Filling
    fill()
    begin_fill()
    end_fill()
More drawing control
    reset()
    clear()
    write()
```

Estado de la tortuga

```
Visibility
    showturtle() | st()
    hideturtle() | ht()
    isvisible()
Appearance
    shape()
    resizemode()
    shapesize() | turtlesize()
    settiltangle()
    tiltangle()
    tilt()
```

Using events

- onclick()
- onrelease()
- ondrag()

Special Turtle methods

- begin_poly()
- end_poly()
- get_poly()
- clone()
- getturtle() | getpen()
- getscreen()
- setundobuffer()
- undobufferentries()
- tracer()
- window_width()
- window_height()

Pantalla

```
Window control
    bgcolor()
    bgpic()
    clear() | clearscreen()
    reset() | resetscreen()
    screensize()
    setworldcoordinates()
Animation control
    delay()
    tracer()
    update()
Using screen events
    listen()
    onkey()
    onclick() | onscreenclick()
    ontimer()
```

```
Settings and special methods
    mode()
    colormode()
    getcanvas()
    getshapes()
    register_shape() | addshape()
    turtles()
    window_height()
    window_width()
Methods specific to Screen
    bye()
    exitonclick()
    setup()
    title()
```

Ejercicios

Determinar los parámetros necesarios para poder realizar los dibujos propuestos a continuación. Puede haber varias formas de escoger los parámetros. Escribir una función que haga el dibujo.

1. Dibujar una línea recta de trazos.
2. Dibujar una línea recta de trazos y puntos.
3. Dibujar una línea recta punteada.
4. Un triángulo equilátero. Determinar las coordenadas de los vértices.
5. Un triángulo rectángulo. Determinar las coordenadas de los vértices.
6. Un triángulo isósceles. Determinar las coordenadas de los vértices.
7. Un triángulo cualquiera. Determinar las coordenadas de los vértices.
8. Dibujar las alturas de un triángulo. Determinar las coordenadas del ortocentro.
9. Dibujar las medianas de un triángulo. Determinar las coordenadas del baricentro.
10. Dibujar las bisectrices de un triángulo. Determinar las coordenadas del incentro.

Otros ejercicios

1. Dibujar las bisectrices de un triángulo. Determinar las coordenadas del incentro.
2. Dibujar las mediatrices de un triángulo. Determinar las coordenadas del circuncentro.
3. Dibujar la recta de Euler. Determinar ecuación de la recta.
4. Dibujar un rectángulo. Determinar las coordenadas de los vértices.
5. Dibujar un paralelogramo. Determinar las coordenadas de los vértices.
6. Dibujar un rombo. Determinar las coordenadas de los vértices.
7. Dibujar un trapecio rectángulo. Determinar las coordenadas de los vértices.
8. Dibujar un trapecio isósceles. Determinar las coordenadas de los vértices.
9. Dibujar un trapecio. Determinar las coordenadas de los vértices.
10. Dibujar un polígono regular, conocida la longitud de un lado. Determinar las coordenadas de los vértices.

Otros ejercicios

1. Dibujar un polígono regular, conocido el radio del círculo circunscrito. Determinar las coordenadas de los vértices.
2. Dibujar una casa usando la función del rectángulo y la del trapecio.