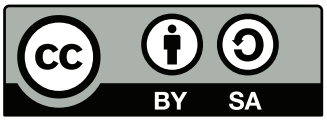
Labs_p5

October 17, 2014



1 Sessió introductòria 5: programació gràfica amb `Turtle`

1.1 Objectiu

La llibreria `turtle` té per objectiu ajudar en l'aprenentatge de la programació a través d'una activitat tant natural com dibuxar. De forma intuitiva, fa possible entendre i practica conceptes com condicionals, bucles o funcions. A més a més, les seves eines permeten realitzar dibuixos senzills essent una bona introducció a la programació de gràfics.

Com qualsevol llibreria de `Python`, cal `import`ar-la; només una vegada i abans de fer-la servir.

```
import turtle
```

1.2 Primer pas: preparació del dibuix amb `Screen`

El primer pas en l'ús de `turtle` és preparar l'àrea de dibuix. Aquest dibuix apareixerà en una finestra independent que s'obre en executar el programa (aneu amb compte, a vegades queda amagada darrera la finestra de notebook i cal clicar per fer-la visible), i podem definir les seves característiques usant les funcions següents:

1. Creem la finestra de treball i l'associem a una variable (`finestra`)

   ```
   finestra = turtle.Screen()
   ```

2. Definim la mida de la finestra de treball

   ```
   finestra.setup(320,240)        # defineix la mida de la finestra, en aquest
                                  # cas un espai de treball de
                                  # 320 punts horitzonals i 240 verticals
   ```

3. Establim un color de fons

   ```
   finestra.bgcolor("white")   # estableix el fons blanc
                               #      accepta qualsevol color en anglès
   ```

4. Establim un títol (apareixerà a dalt de la finestra)

   ```
   finestra.title("El meu primer dibux amb turtle")
   ```

5. Incloem les comandes per traçar el dibuix

   ```
   (Les veurem en les seccions següents)
   ```

6. Després de fer el traçat acabem definint com acabar el procés. La comanda següent fa que la finestra de dibuix es pugui tancar clicant sobre ella.

   ```
   finestra.exitonclick()       #ATENCIÓ: AQUESTA COMANDA HA D'ANAR DESPRËS DE TOTES
                                #         LES COMANDES DE DIBUIX
   ```

1.3 Comandes per dibuixar

A continuació, ja podem començar a dibuixar. Tal i com indica el seu nom, Turtle assumeix que els nostres dibuixos els realitzen unes tortuguetes pintores que obeeixen les nostres instruccions. Aquestes tortugues s'arrosseguen per la finestra de treball amb un pinzell a la panxa deixant un rastre de color.

Abans de començar a dibuixar, hem de "crear" una tortuga. Per això només cal fer

```
alex = turtle.Turtle()  # crea una nova tortuga i l'assigna a la variable 'alex'
```

Les tortugues obeeixen instruccions simples, com ara avançar, retrocedir, girar a l'esquerra o a la dreta, canviar de color de pinzell, etc. Podem fer que la tortuga creada executi aquestes instruccions mitjançant diverses funcions associades. A continuació téniu un exemple amb les instruccions de moviment més rellevants.

```
alex.forward(100)    # La tortuga alex avança 100 punts
alex.right(90)       # La tortuga alex gira 90 graus cap a la dreta
alex.backward(200)   # La tortuga alex retrocedeix 200 punts
alex.left(45)        # La tortuga alex gira 45 graus cap a l'esquerra
```

Les instruccions anteriors desplacen la tortuga de forma *relativa* a la seva posició i orientació actuals. També és possible utilitzar coordenades *absolutes*: el centre de la finestre de treball és la posició (0,0), els eixos cartesians (x,y) tenen el signe i horientacions habituals:

```
alex.goto(45, -34)   # alex es trasllada a les coordenades (45, -34)
alex.setx(10)        # alex es desplaça, en horitzonal, a la posició d'abscissa 10
alex.sety(-50)       # alex es desplaça, en vertical, a la posició d'ordenada -50
alex.setheading(180) # alex gira i apunta a la direcció 180° (cap a la dreta)
alex.home()          # alex es desplaça al centre de la pantalla i s'orienta cap a dalt
```

Per defecte les accions anteriors fan que la tortuga es desplaci dibuixant el seu rastre. Si el que volem és desplaçar la tortuga a un altre punt, però sense dibuixar disposem de les següents eines

```
alex.penup()       # Alex aixeca el pinzell, després d'aquesta instrucció la tortuga no deixa rastre
alex.pendown()     # Alex abaixa el pinzell, després la tortuga torna a deixar rastre
```

També és possible cambiar les caractarístiques del rastre de la tortuga; per exemple

```
alex.pensize("3")   # El pinzell de la tortuga alex té un gruix de 3 punts
alex.color("red")   # El rastre de la tortuga alex serà vermell
```

i també modificar l'aspecte de la tortuga

```
alex.shape("circle")  # La tortuga alex tindrà aspecte de cercle.
                      #   altres valors possible són:
                      #    \arrow", \turtle", \circle", \square", \triangle", \classic"
```

O bé fer que deixi la seva empremta

```
alex.stamp()          # La tortuga alex deixarà la seva emprenta en el lloc actual
```

Finalment, és possible reiniciar totes les propietats d'una tortuga.

```
alex.reset()          # alex torna a la posició inicial amb els paràmetres de color, etc.
                      # inicials i s'esborra tot el que hagi dibuixat!!
```

Com a comentari final, recordeu que podeu "crear" tantes tortugues com volgueu en un mateix dibuix. L'únic requisit és que s'anomenin diferent (noms de variable diferents) por poder-les cridar independentment!

Podeu trobar una referència complerta a totes les comandes de turtle en el següent enllaç

docs.python.org/2/library/turtle.html

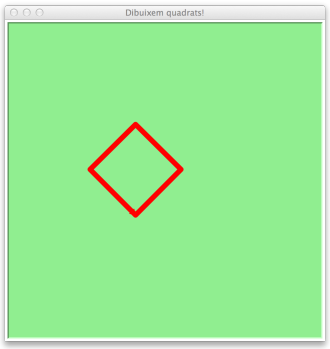
El següent codi exemplifica totes les comandes vistes en aquest apartat. Analitzeu-lo i comproveu el resultat de la seva execució.

```
In [2]: # Exemple bàsic amb Turtle
        import turtle

        finestra = turtle.Screen()      # creem una nova finestra de dibuix
        finestra.setup(400, 400)              # mida 400x400 punts
        finestra.bgcolor("lightgreen")        # color verd clar
        finestra.title("La tortuga Maria!")   # amb un títol

        maria = turtle.Turtle()   # creem la tortuga maria

        maria.forward(40)             # dibuixem un triangle equilater
        maria.left(120)
        maria.pensize("5")            # amb diferents estils de línia i color
        maria.forward(40)
        maria.left(120)
        maria.color("blue")
        maria.forward(40)
        maria.left(120)

        maria.penup()                 # desplacem la tortuga sense dibuixar
        maria.goto(-100, -100)
        maria.stamp()                 # deixem una emprempta (fletxa)
        maria.setheading(90)
        maria.forward(30)
        maria.shape("turtle")         # cambiem l'emprempta a tortuga
        maria.stamp()

        maria.home()                  # tornem a la posició inicial

        finestra.exitonclick()  # definim la forma de tancar la finestra de treball
```

1.3.1 Exercici 1

Feu un programa que dibuixi un quadrat de 100 punts de costat tal que:

- El quadrat sigui vermell
- Tigui la vora groixuda
- Estigui girat 45 graus
- Estigui centrat fora de l'orígen



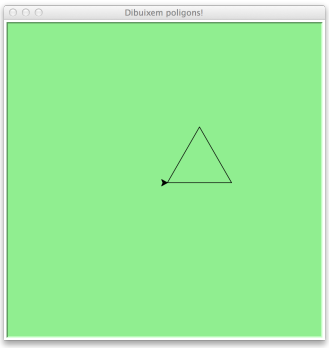
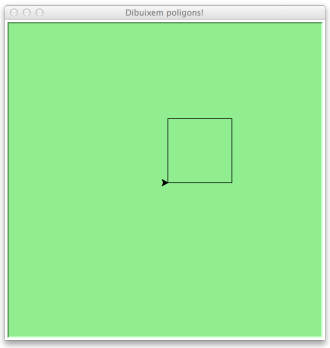
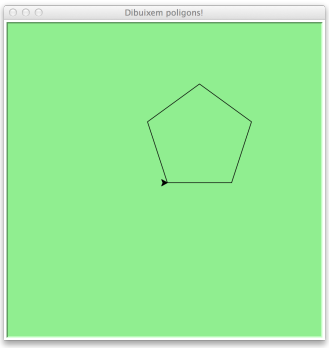
1.3.2 Exercici 2

Feu un programa que dibuixi un polígon regular de **qualsevol** nombre de costats. El programa ha de demanar per consola el nombre de costats i la mida del polígon i dibuixar-lo.

Opcional: Estructureu el programa de forma que el traçat del polígon estigui encapsulat en una funció:

```
dibuixa_poligon(tortuga, costats, mida)
```

on `tortuga` és la variable que heu creat a la tortuga.

N=3 L=100	N=4 L=100	N=5 L=100
		

1.4 Números aleatoris

- `random`: La llibreria random, conté un seguit d'eines per generar números (pseudo)aleatoris. Per utilitzar-la, cal `import`ar-la al nostre programa, només una vegada i abans d'utilitzar-la.
- `random.randint(min,max)`: genera un enter aleatori entre `min` i `max` (ambdós inclosos).
- `random.random()`: genera un número de coma flotant entre 0.0 i 1.0.

Exemple: números aleatoris
Executeu l'exemple següent varies vegades:

```
import random
enter = random.randint(1,3)     # genera un enter aleatori que pot valdre 1, 2 o 3.
decimal = random.random()       # genera un real aleatrori entre 0.0 i 1.0
print(enter,decimal)
```

Comenteu el resultat

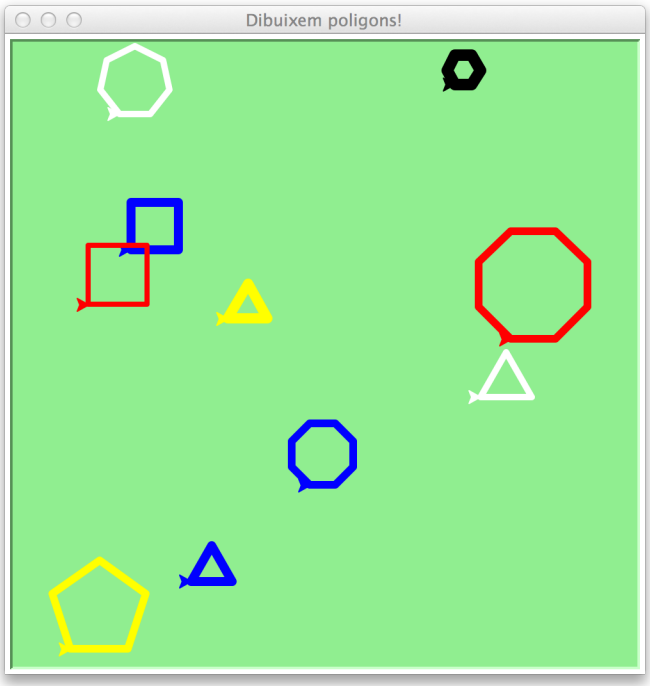
1.4.1 Exercici 3

Modifiqueu l'exercici anterior per fer un programa que dibuixi 10 polígons regulars de característiques aleatòries: entre 3 i 10 costats, i mides entre 5 i 50 punts de costat. Aquesta modificació us resultarà molt més fàcil si heu empaquetat la generació d'un polígon en una funció `dibuixa_poligon()`

Ajuda: amb la funció `random.randint(min, max)` heu de generar números aleatoris - costats entre 3 i 10 i mida entre 5 i 50 - que usareu per generar els polígons.

Opcional: modifiqueu el programa per que a més a més:

- Utilitzi gruixos de línia aleatoris (entre 1 i 10)
- Utilitzi 5 colors aleatoris diferents
- Centri els polígons en posicions aleatòries del dibuix



1.5 Data i hora

- La llibreria `datetime`, inclosa en Python, conté eines per conèixer el dia i l'hora del sistema.
  El següent codi, permet obtenir informació sobre el moment actual:

  from datetime import datetime
  ara = datetime.now() print ara

- Després de llegir el moment actual (`datetime.now()`) i guardar-lo a la variable `ara`, podem demanar, per separat, els diferents elements de la data i l'hora:

  print ara.year print ara.month print ara.day print ara.hour print ara.minute print ara.second

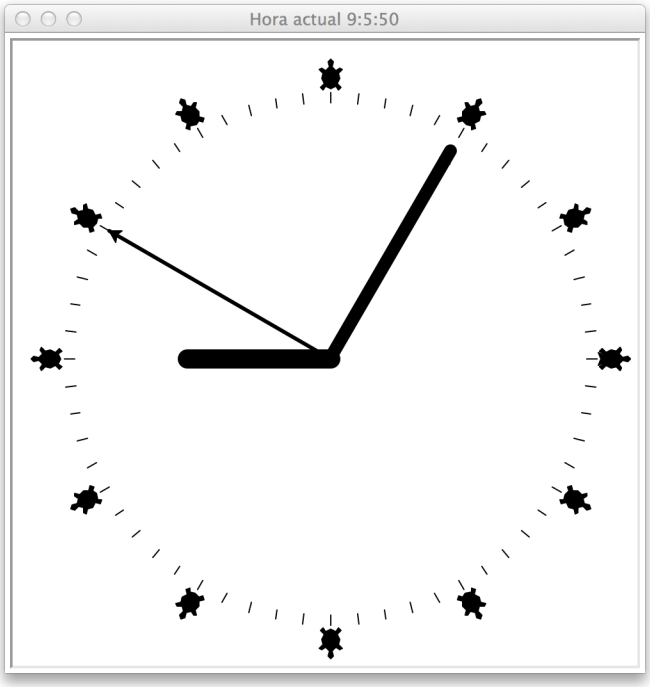
1.5.1 Exercici 4 (opcional)

Feu un programa amb turtle que representi l'hora actual en un rellotge analògic.

- En primer lloc, familiaritzeu-vos amb l'ús de la llibreria `datetime`, seguint els exemples de l'apartat anterior.

- Seguidament dibuixeu un cercle amb les 12 marques de les hores (i opcionalment els minuts) del rellotge
- Dibuixeu les busques del rellotge (hores, minuts i opcionalment segons) com a linies (de gruix i longitud diferent) amb els angles correctes per representar l'hora

Nota: no us demanem un rellotge dinàmic, sinó estàtic, que marqui la hora en l'instànt de l'execució.



1.6 Informe a entregar

Construiu un Notebook que resolgui, amb les explicacions i comentaris adequats, els 3(4) exercicis anteriors. En acabar, entregeu el fitxer Notebook generat a través de Campus Virtual

```
In []:
```