Softech®
Gather to thrive together

Aptech™
COMPUTER EDUCATION
Unleash your potential

BÀI 11

Turtle Graphic
Đồ họa hình con Rùa

Tóm Tắt Nội Dung

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

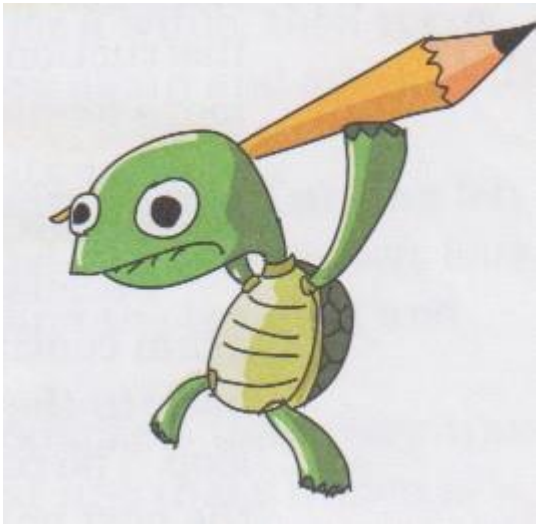
1. Tổng quan về Đồ họa con Rùa - **Turtle Graphics**
2. Thiết lập môi trường sân khấu đồ họa
3. Các lệnh điều khiển Rùa trên màn hình đồ họa

11.1 Tổng quan đồ họa con Rùa

Turtle Graphics là gì?

Turtle Graphics là một chương trình con có sẵn trong Python khi bạn đã cài Python lên máy bạn.

Turtle (con rùa) đây chỉ là hình ảnh tưởng tượng, nó cầm cây bút để vẽ, di chuyển đến đâu là nó vẽ đường thẳng đến đó.
Nó vẽ bằng những cái chấm (dots-pixels) trên màn hình.



Bạn có thể vẽ các hình đồ họa theo ý thích của mình với Turtle Graphic

11.2 Thiết lập môi trường sân khấu đồ họa

Để khởi động môi trường độ hòa chúng ta cần đến thư viện **turtle**

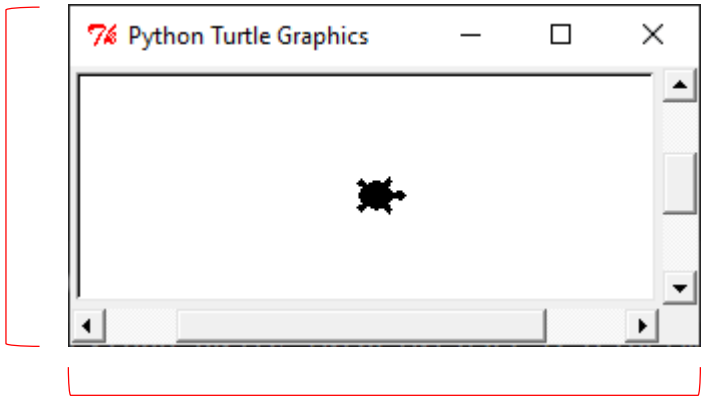
```
import turtle as t
```

Trước tiên chúng ta cần làm quen với một số thuật ngữ quan trọng liên quan đến môi trường đồ họa Turtle trong Python

Cửa sổ đồ họa

Là khung cửa sổ làm việc trong chế độ đồ họa.

Cửa số này mặc định nằm chính giữa màn hình và có kích thước mặc định 640x480px



11.2 Thiết lập môi trường sân khấu đồ họa

Thiết lập kích thước cửa sổ đồ họa

- **Lệnh setup()**

Thiết lập chiều rộng, chiều cao cửa sổ đồ họa và vị trí tương đối so với màn hình máy tính.

setup(width=0.5, height=0.75, startX=None, startY=None)

```
t.setup(400, 300) # Kích thước cửa sổ là 400x300px
t.setup(0.5, 0.75) # Tỉ lệ so với chiêu rộng, cao của màn hình
t.setup(400, 300, 100, 200) # nằm cách vị trí bên trái 100px và bên trên 200px của màn hình
```

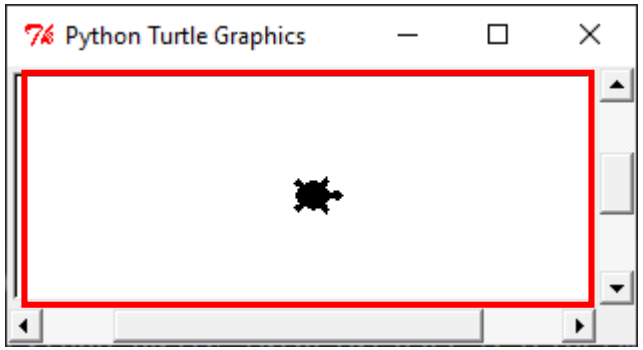
11.2 Thiết lập môi trường sân khấu đồ họa

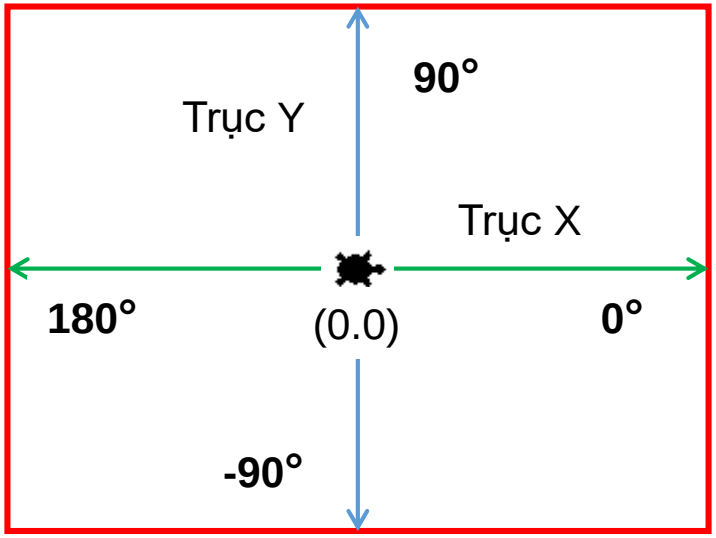
Vùng làm việc Canvas

Là vùng tọa độ logic mà nhân vật Rùa có thể di chuyển bên trong

Vùng này được xác định bởi thông số như chiều cao, rộng, màu nền màn hình

Vùng này được chia theo hệ trục tọa độ (X,Y). Mặc định tâm (X=0, Y=0) ở chính giữa vùng.





11.2 Thiết lập môi trường sân khấu đồ họa

Thiết lập kích thước Vùng làm việc Canvas

- **Lệnh screensize()**

Thiết lập chiều rộng, chiều cao cho vùng làm việc (canvas) hiện thời

screensize(width=None, height=None, bg=None)

```
t.screensize() # Trả về kích thước vùng làm việc canvas
t.screensize(bg=blue) # Thiết lập màu nền cho vùng làm việc canvas
t.screensize(400, 300) # Thiết lập  kích thước cho vùng làm việc
```

11.2 Thiết lập môi trường sân khấu đồ họa

Thiết lập kích thước Vùng làm việc Canvas

- **Lệnh setworldcoordinates()**

Thiết lập hệ tọa độ ảo cho vùng làm việc hiện thời. Hệ tọa độ này sẽ được áp lên vùng canvas hiện thời, và tâm của vùng làm việc sẽ được thiết lập lại.

setworldcoordinates(llx, lly, urx, urly)

- llx: tọa độ x của điểm góc trái bên dưới vùng làm việc
- lly: tọa độ y của điểm góc trái bên dưới vùng làm việc
- llx: tọa độ x của điểm góc phải bên trên vùng làm việc
- lly: tọa độ y của điểm góc phải bên trên vùng làm việc

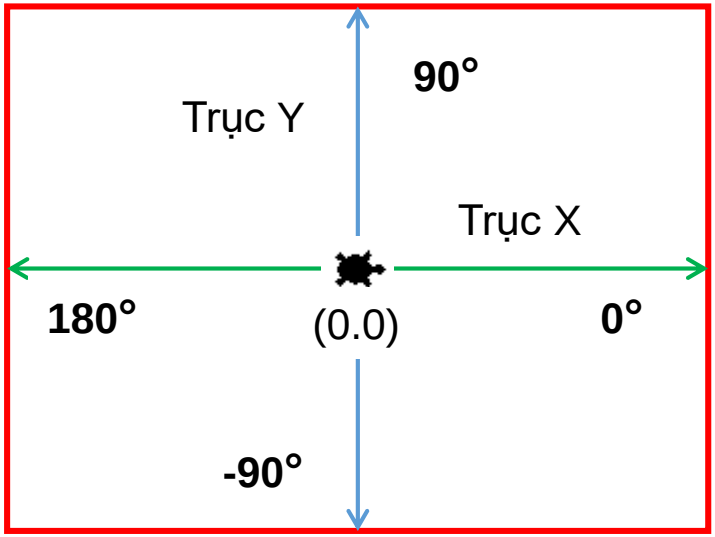
```
t. setworldcoordinates(-240, -180, 240, 180) # Thiết lập hệ tọa độ với gốc trái bên dưới (-240, -180) và góc phải bên trên (240, 180). Do vậy tâm nằm giữa cửa sổ và các trục tọa độ theo chiều rộng là 480, chiều cao 360
```

11.3 Các lệnh điều khiển Rùa trên màn hình đồ họa

Hướng Rùa, chế độ vẽ và tọa độ màn hình

Trạng thái của Rùa trên màn hình được xác định bởi 3 yêu tố:

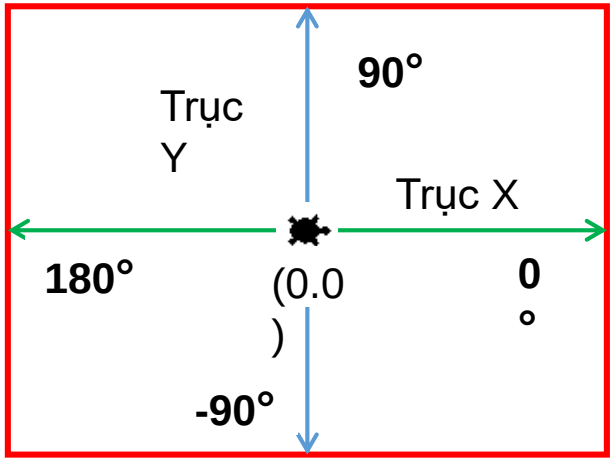
- Hướng Rùa (heading): mặc định là 0°
- Tọa độ X
- Tọa độ Y



Chế độ vẽ: khi di chuyển Rùa sẽ không để lại dấu vết là “nâng bút” (penup), ngược lại chế độ “hạ bút” (pendown). Chế độ mặc định là hạ bút.

16.3 Các lệnh điều khiển Rùa trên màn hình đồ họa

Thông số hiện thời của Rùa



Chúng ta có các lệnh sau:

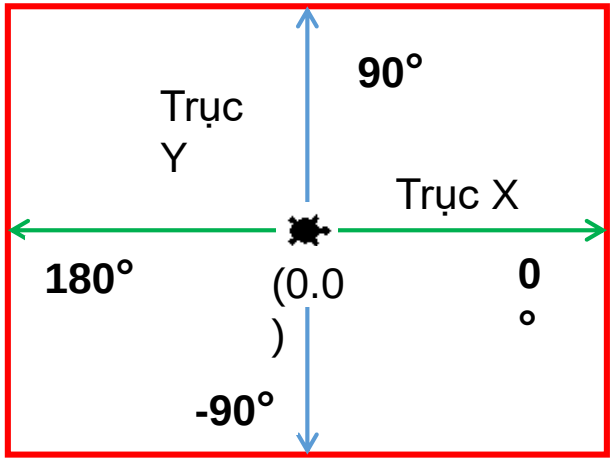
- **heading():** Hướng hiện thời của rùa, 0 - 360°
- **xcor():** tọa độ x hiện thời của rùa
- **ycor():** tọa độ y hiện thời của rùa
- **penup()/up/pu:** chế độ nâng bút
- **pendown()/down/pd:** chế độ hạ bút
- **isdown():** trả về trạng thái bút vẽ, True nếu đang hạ bút, False nếu đang nâng bút
- **pensize(number):** thiết lập độ dày nét vẽ của rùa
- **pencolor(color):** thiết lập màu sắc nét vẽ của rùa

11.3 Các lệnh điều khiển Rùa trên màn hình đồ họa

Hình dáng của Rùa

Chúng ta có thể thay đổi hình dáng thực thể Rùa với nhiều kiểu như:

- **arrow**: hình mũi tên
- **turtle**: hình con rùa
- **circle**: hình tròn
- **square**: hình vuông
- **triangle**: hình chữ nhật




```
t.shape("turtle")
t.shape("arrow")
t.shape("circle")
```

11.3 Các lệnh điều khiển Rùa trên màn hình đồ họa

Các lệnh điều khiển Rùa trên màn hình

Chúng ta có thể thay đổi hình dáng thực thể Rùa với nhiều kiểu như:

- **forward(n)/ fd(n)**: di chuyển về trước với khoảng cách n, n có thể là số nguyên hoặc thập phân. Nếu số âm thì đi lùi.
- **backward(n)/ bk(n)**: di chuyển về sau với khoảng cách n, n có thể là số nguyên hoặc thập phân. Nếu số âm thì đi tới.
- **left(**alpha**)/ lt(**alpha**)**: quay sang trái một góc alpha độ
- **right(**alpha**)/ rt(**alpha**)**: quay sang phải một góc alpha độ
- **seth(**angle**)/ setheading(**angle**)**: thay đổi hướng rùa theo số đo angle độ
- **goto(**x, y**)/ setpos(**x,y**)**: di chuyển nhanh đến vị trí tọa độ (x, y)
- **set(**x**)** : Thiết lập tọa độ x của Rùa
- **set(**y**)** : Thiết lập tọa độ y của Rùa
- **home()** : đưa Rùa về gốc và hướng về 0 độ
- **reset()** : Xóa màn hình và đưa rùa về chế độ mặc định

11.3 Các lệnh điều khiển Rùa trên màn hình đồ họa

Các lệnh điều khiển Rùa trên màn hình

- **speed(**tốc độ**)**: tốc độ tăng dần trong khoảng 1 - 10
- **circle(**d, n**)**:
  - vẽ hình tròn với bán kính d, nếu d > 0 hình tròn vẽ ngược chiều kim đồng hồ, d < 0 cùng chiều kim đồng hồ.
  - n: tùy chọn, là góc vẽ
- **fillcolor(**color**)**: tô màu nền cho thực thể Rùa

11.3 Các lệnh điều khiển Rùa trên màn hình đồ họa

Các lệnh điều khiển Rùa trên màn hình

Ví dụ để vẽ một hình vuông

```
import turtle as t
t.shape("turtle") # Thiết lập hình dáng là Rùa
t.pendown() # Chế độ hạ bút
t.pencolor('red') # Thiết lập màu cho nét vẽ

def square(d):
    for i in range(4):
        t.forward(d)
        t.left(90)

square(100) # vẻ hình vuông, với cạnh là 100px
t.mainloop() # Duy trì cửa sổ đồ họa sau khi vẽ xong
```

Tổng kết lại bài

1. Nắm được tổng quan lỗi xảy ra trong Python
2. Nắm được cách bắt lỗi và xử lý lỗi