



Manual Turtle Modul

Modul Turtle adalah modul yang dapat digunakan oleh bahasa Pemrograman Python untuk menggambar bentuk ataupun grafik. Manual ini akan berisi perintah-perintah dasar yang digunakan untuk menggambar bentuk dengan menggunakan modul turtle.

Terdapat dua hal utama agar modul turtle ini dapat dijalankan :

1. Import Turtle
2. Inisialisasi Subclass

Import Turtle

Seperti halnya package atau modul yang lain, untuk menggunakan atau memanfaatkan semua method atau fungsi yang terdapat pada modul turtle, maka terlebih dahulu dilakukan perintah import, seperti berikut:

```
import turtle
```

Inisialisasi Subclass

Turtle sendiri terbagi akan 2 Subclass agar dapat berjalan, **turtle.Screen** (yang berfungsi sebagai dasar layar untuk turtle) dan **turtle.Turtle** (yang berfungsi sebagai deklarasi awal obyek turtle). Inisialisasi screen dan turtle ini, adalah sebagai berikut :

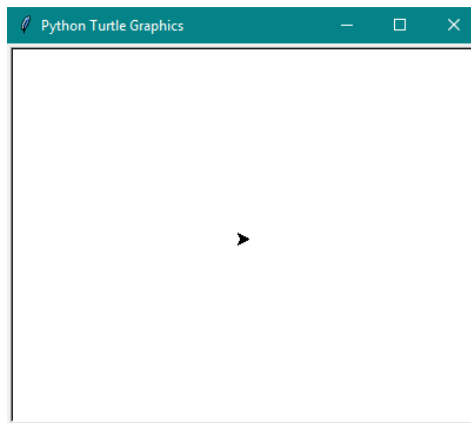
```
NamaVariabel1=turtle.Screen()  
NamaVariabel2=turtle.Turtle()
```

Berikut adalah property dan method yang terdapat pada turtle dan dapat digunakan untuk menggambar suatu pola atau grafik.

Koordinat layar

Untuk mengerjakan objek turtle hal yang perlu diperhatikan adalah masalah kordinat. Turtle selalu dimulai pada titik (0,0) dan menghadap ke Timur. Koordinat (0,0) terdapat di tengah layar turtle, seperti yang terlihat pada Gambar 1. Code untuk inisialisasi dapat dilihat pada code berikut.

```
import turtle  
  
objPointer = turtle.Turtle()  
windowX = turtle.Screen()  
  
windowX.setup(450,350) # merupakan ukuran layar height x width
```



Gambar 1. Koordinat (0,0) layar turtle

Pergerakan Obyek

Pergerakan objek dalam turtle bergantung pada arah pointer saat ini, karena dalam turtle pergerakan dari titik A ke titik B dapat ditempuh dengan beberapa cara,

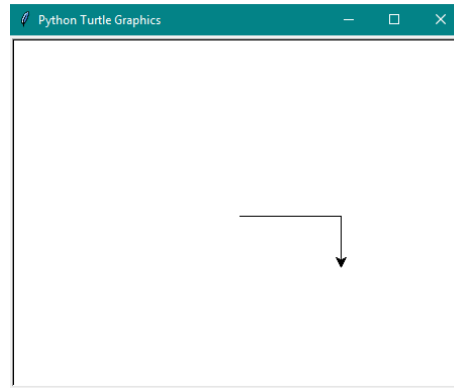
- a. Memanfaatkan method forward dan point left/right,
- b. Menggunakan setpos, goto,
- c. setx ataupun sety.

Forward, point left/right

Inti dari gabungan method ini adalah pergerakan lurus sesuai arah pointer yang telah diputar beberapa derajat, seperti contoh code berikut ini.

```
1. import turtle
2.
3. objPointer = turtle.Turtle()
4. windowX = turtle.Screen()
5.
6. windowX.setup(450,350)
7.
8. objPointer.forward(100)
9. objPointer.right(90)
10. objPointer.forward(50)
```

Pada baris ke 8, adalah perintah agar pointer bergerak maju sebanyak 100 titik. Karena default arah pointer ke arah timur, maka, perintah maju adalah memerintahkan kursor bergerak ke arah timur (kanan) sebanyak 100 titik. Baris ke 9, adalah perintah untuk memutar arah pointer ke kanan sebanyak 90°. Setelah pointer diputar sebanyak 900, pada baris ke-10 adalah perintah untuk maju sebanyak 50 langkah. Contoh pergerakan yang dihasilkan oleh perintah baris ke-8 sampai dengan baris ke-10, dapat dilihat pada Gambar 2.



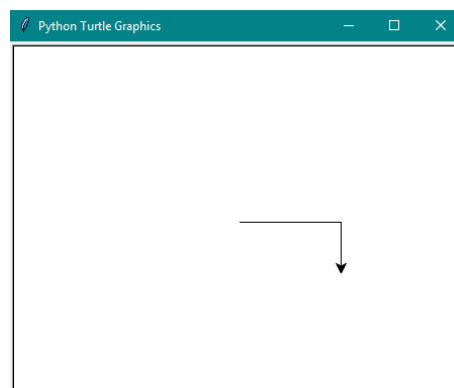
Gambar 2. Pergerakan pointer `forward(100)`, `right(90)`, dan `forward(50)`

setpos atau goto

Perpindahan dengan cara ini bergantung akan kordinat yang telah di tetapkan tanpa bergantung akan arah pointer saat ini. Contoh code untuk menggerakkan pointer dengan setpos adalah sebagai berikut :

```
1. import turtle
2.
3. objPointer = turtle.Turtle()
4. windowX = turtle.Screen()
5.
6. windowX.setup(450,350)
7.
8. objPointer.setpos(100,0)
9. objPointer.setpos(100,-50)
```

Pada baris ke-8, adalah perintah untuk menetapkan lokasi pointer berada di (100,0), sedangkan baris ke-9 adalah perintah untuk memindah pointer ke arah (100,-50). Contoh pergerakan ini dapat dilihat pada Gambar 3.



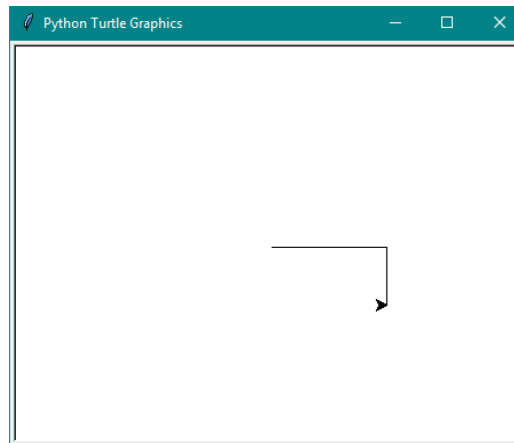
Gambar 3. Pergerakan pointer `setpos(100,0)`, `setpos(100,-50)`

setx atau sety

Hampir sama dengan method setpos tetapi perpindahan ini hanya memerlukan 1 kordinat x ataupun y. Perintah setx digunakan untuk bergerak searah sumbu x, sedangkan sety digunakan untuk bergerak searah sumbu y. Contoh penggunaan kedua fungsi ini, dapat dilihat sebagai berikut :

```
1. import turtle
2.
3. objPointer = turtle.Turtle()
4. windowX = turtle.Screen()
5.
6. windowX.setup(450,350)
7.
8. objPointer.setx(100)
9. objPointer.sety(-50)
```

Hasil dari code tersebut, dapat dilihat pada Gambar 4.









Gambar 4. Pergerakan pointer setx(100), sety(-50)

Bentuk Pointer

Secara default, pointer berbentuk seperti tanda panah (lihat Gambar 1 sampai dengan Gambar 4). Tampilan pointer default ini dapat dirubah sesuai kebutuhan dan bentuk yang sudah disiapkan, dengan menggunakan perintah `shape (. . .)`

Contoh bentuk pointer dan syntax perubahan pointer, dapat dilihat pada Tabel 1.

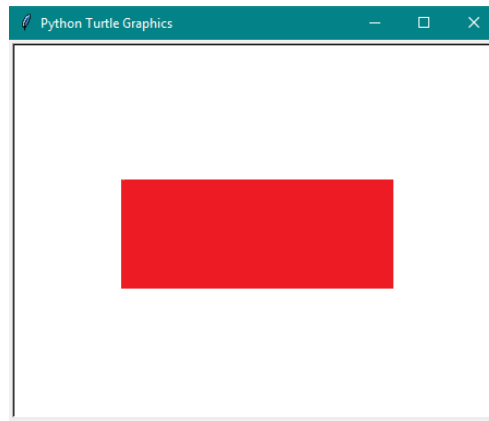
Tabel 1. Bentuk Pointer

Jenis Pointer	Syntax	Bentuk Pointer
arrow	<code>objPointer.shape("turtle")</code>	
blank	<code>objPointer.shape("blank")</code>	
circle	<code>objPointer.shape("circle")</code>	
classic	<code>objPointer.shape("classic")</code>	
square	<code>objPointer.shape("square")</code>	
triangle	<code>objPointer.shape("triangle")</code>	
turtle	<code>objPointer.shape("turtule")</code>	

model pointer tersebut juga dapat di manipulasi untuk diganti dengan gambar buatan sendiri (dalam bentuk gif), dengan memanfaatkan fungsi `addshape`, seperti contoh berikut:

```
1. import turtle
2.
3. objPointer = turtle.Turtle()
4. windowX = turtle.Screen()
5.
6. windowX.setup(450,350)
7.
8. turtle.addshape("1.gif")
9. objPointer.shape("1.gif")
```

Contoh pointer yang telah dirubah, dapat dilihat padag Gambar 5.



Gambar 5. Bentuk pointer kustom

penup dan pendown

Kombinasi dari method ini digunakan untuk memindahkan kordinat suatu pointer ke titik tertentu tanpa harus membuat suatu garis saat melakukan perpindahan. Berikut adalah code untuk mengatur pointer, agar pointer dapat bergerak tanpa meninggalkan jejak garis seperti sebelumnya.

```
1. import turtle
2.
3. objPointer = turtle.Turtle()
4. windowX = turtle.Screen()
5.
6. windowX.setup(450,350)
7.
8. objPointer.penup()
9. objPointer.forward(100)
10. objPointer.right(90)
11. objPointer.forward(50)
12. objPointer.pendown()
```

Pada baris ke-8, terdapat perintah untuk `penup`, hal ini berarti bahwa perintah-perintah pergerakan berikutnya (baris ke-9, baris ke-10, dan baris ke-11), tidak akan meninggalkan jejak garis. Jika ingin meninggalkan jejak garis, maka lakukan perintah `pendown`.

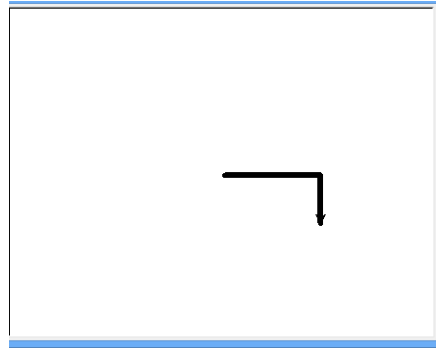
Pensize

Pensize berfungsi untuk memberi efek ketebalan garis. Contoh code untuk menebalkan garis ini adalah sebagai berikut:

```
1. import turtle
2.
3. objPointer = turtle.Turtle()
4. windowX = turtle.Screen()
5.
```

```
6. windowX.setup(450,350)
7.
8. objPointer.pensize(6)
9. objPointer.forward(100)
10. objPointer.right(90)
11. objPointer.forward(50)
```

Contoh ketebalan garis yang dihasilkan dari code tersebut, dapat dilihat pada Gambar 6.



Gambar 6. Penebalan garis

Pointer speed

Untuk mengatur kecepatan pointer saat berjalan dapat dimanipulasi dengan menggunakan speed, kecepatannya dapat menggunakan range angka dari 0-10 atau “fastest”:0, “fast”: 10, dan sebagainya.

Contoh code untuk pengaturan kecepatan ini adalah sebagai berikut

```
1. import turtle
2.
3. objPointer = turtle.Turtle()
4. windowX = turtle.Screen()
5.
6. windowX.setup(450,350)
7.
8. objPointer.speed("slowest")
9. objPointer.forward(100)
10. objPointer.right(90)
11. objPointer.forward(50)
```

Contoh code – bentuk rekursif

Berikut ini adalah contoh penggunaan modul turtle untuk menggambar segitiga Sierpinski ini yang merupakan salah satu bentuk fractal (bentuk berulang). Segitiga ini memiliki bentuk dasar (yaitu, suatu segitiga dan didalamnya terdapat empat buah segitiga dengan ukuran yang sama), dan bentuk dasar ini terus menerus dibangkitkan sampai derajat tertentu. Pembuatan segitiga ini dilakukan dengan cara rekursif

Langkah pertama yang harus dilakukan adalah dalam pembuatan turtle yaitu import dan installasi turtle kedalam dasar code. Inisiasi ini diletakkan dalam fungsi main, seperti contoh berikut :

```
1. def main():
2.     my_turtle = turtle.Turtle()
3.     my_win = turtle.Screen()
4.     my_points = [[-100, -50], [0, 100], [100, -50]]
5.     sierpinski(my_points, 0, my_turtle)
6.     my_win.exitonclick()
```

Baris ke-4 pada code tersebut adalah penentuan posisi awal dari pointer, yang memiliki tiga buah koordinat. Baris ke-5 digunakan untuk memanggil fungsi sierpinski, yaitu fungsi untuk menggambar segitiga secara rekursif. Fungsi rekursif ini memiliki 3 parameter, yaitu, parameter pertama merupakan koordinat awal dari segitiga, parameter kedua berupa derajat atau level dari rekursif, dan parameter ketiga adalah obyek turtle. Jika derajat bernilai 0, maka hanya akan dihasilkan satu segitiga saja, akan tetapi ketika derajat bernilai 1, maka akan dihasilkan 1 segitiga besar, dan 4 segitiga kecil di dalam segitiga besar, dan begitu seterusnya.

Langkah kedua adalah Pembuatan fungsi rekursif segitiga sierpinski, seperti yang terdapat pada code berikut ini.

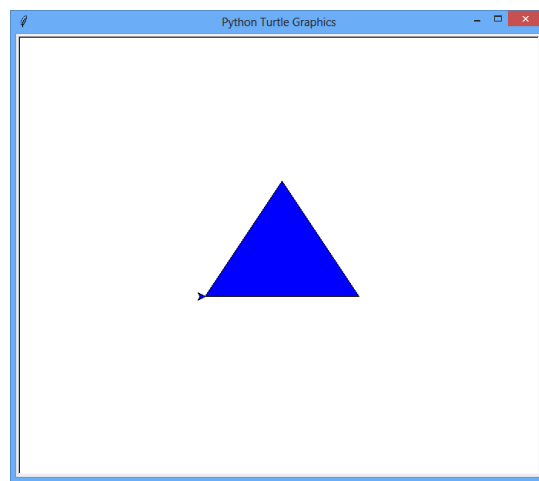
```
1. import turtle
2. def draw_triangle(points, color, my_turtle):
3.     my_turtle.speed('slowest')
4.     my_turtle.fillcolor(color)
5.     my_turtle.up()
6.     my_turtle.goto(points[0][0], points[0][1])
7.     my_turtle.down()
8.     my_turtle.begin_fill()
9.     my_turtle.goto(points[1][0], points[1][1])
10.    my_turtle.goto(points[2][0], points[2][1])
11.    my_turtle.goto(points[0][0], points[0][1])
12.    my_turtle.end_fill()
13. def get_mid(p1, p2):
14.     return ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2)
15. def sierpinski(points, degree, my_turtle):
16.     color_map = ["blue", "red", "green", "white", "yellow", "violet", "orange"]
17.     draw_triangle(points, color_map[degree], my_turtle)
18.     if degree > 0:
19.         sierpinski([points[0], get_mid(points[0], points[1]), get_mid(points[0], points[2])], degree-1, my_turtle)
20.         sierpinski([points[1], get_mid(points[0], points[1]), get_mid(points[1], points[2])], degree-1, my_turtle)
21.         sierpinski([points[2], get_mid(points[2], points[1]), get_mid(points[0], points[2])], degree-1, my_turtle)
```

Pada baris ke-2 sampai dengan baris ke-12 adalah fungsi pembuatan segitiga. Fungsi ini memiliki tiga parameter, yaitu koordinat segitiga, warna, dan obyek turtle. Baris ke-6 sampai dengan baris ke-11, adalah perintah-perintah untuk membuat segitiga, mulai dari menggambar satu sisi, sampai dengan sisi terakhir.

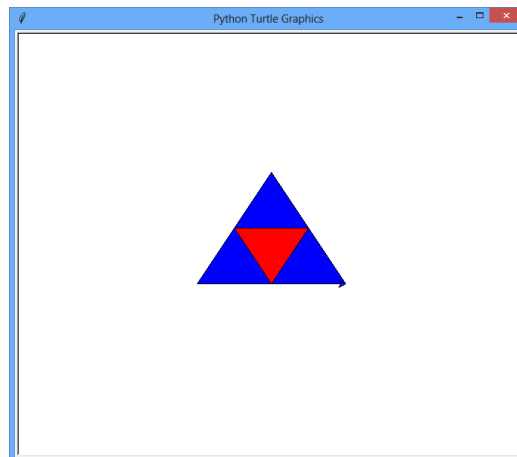
Fungsi `get_mid` pada baris ke-13 sampai dengan ke-14, adalah fungsi untuk mendapatkan nilai tengah dari dua buah sisi. Nilai-nilai tengah ini digunakan untuk menggambar segitiga kecil di dalam segitiga besar.

Fungsi `sierpinski` pada baris ke-15 sampai dengan baris ke-21, adalah fungsi rekursif untuk pembuatan segitiga secara berulang. Baris ke-19 sampai dengan ke-21, adalah fungsi rekursif untuk membentuk tiga segitiga di dalam suatu segitiga, sehingga akan terbentuk 4 segitiga, di dalam suatu segitiga.

Gambar segitiga sierpinski untuk derajat 0, dapat dilihat pada Gambar 7. Sedangkan segitiga sierpinski untuk derajat 1 dan derajat 2, dapat dilihat pada Gambar 8 dan Gambar 9. Dapat dilihat pada Gambar 8, terdapat 4 segitiga (3 berwarna biru yang merupakan hasil dari fungsi `sierpinski`, dan 1 berwarna merah yang merupakan background dari segitiga besar yang tersisa), di dalam segitiga besar



Gambar 7. Segitiga Sierpinski untuk derajat 0



Gambar 8. Segitiga Sierpinski untuk derajat 1

