COVER

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

LIBRARY TURTLE

1. Introduction

Grafik Turtle adalah implementasi dari [alat menggambar geometris populer yang diperkenalkan di Logo](https://en.wikipedia.org/wiki/Turtle_(robot)), dikembangkan oleh Wally Feurzeig, Seymour Papert dan Cynthia Solomon pada tahun 1967.

Dengan Python, grafik penyu memberikan representasi fisik “kura-kura” (robot kecil dengan pena) yang menggambar pada selembar kertas di lantai. Ini adalah cara yang efektif dan terbukti bagi pelajar untuk menemukan konsep pemrograman dan interaksi dengan perangkat lunak, karena memberikan umpan balik yang instan dan terlihat. Ini juga menyediakan akses mudah ke output grafis secara umum.

Gambar penyu pada awalnya dibuat sebagai alat pendidikan, untuk digunakan oleh guru di kelas. Bagi pemrogram yang perlu menghasilkan beberapa keluaran grafis, ini bisa menjadi cara untuk melakukannya tanpa perlu mengeluarkan perpustakaan yang lebih kompleks atau eksternal ke dalam pekerjaan mereka.

1. Tutorial

Pengguna baru harus mulai dari sini. Dalam tutorial ini kita akan menjelajahi beberapa dasar menggambar kura-kura.

1. Memulai turtle environment

Di shell Python, impor semua objek dari turtle modul:

**from** **turtle** **import** \*

Jika anda mengalami eror : No module named '\_tkinter', anda harus menginstall [Tk interface package](https://docs.python.org/3/library/tkinter.html#module-tkinter) pada system anda.

1. Dasar Menggambar

Kirim kura-kura ke depan 100 langkah:

forward(100)

Anda akan melihat (kemungkinan besar, di jendela baru di layar Anda) garis yang ditarik oleh kura-kura, menuju ke Timur. Ubah arah penyu, sehingga berbelok 120 derajat ke kiri (berlawanan arah jarum jam):

left(120)

Mari lanjutkan dengan menggambar segitiga:

forward(100)

left(120)

forward(100)

Perhatikan bagaimana kura-kura, yang diwakili oleh anak panah, menunjuk ke arah yang berbeda saat Anda mengarahkannya.

Bereksperimenlah dengan perintah-perintah itu, dan juga dengan backward() and right().

1. Pen Control

Coba ubah warnanya - misalnya, color('blue') - dan lebar garis - misalnya, width(3) - lalu menggambar lagi.

Anda juga dapat menggerakkan kura-kura tanpa menggambar, dengan mengangkat pena: up() ebelum pindah. Untuk mulai menggambar lagi, gunakan down().

1. The turtle’s Position

Kirim kura-kura Anda kembali ke titik awalnya (berguna jika ia menghilang di luar layar):

home()

Posisi home berada di tengah-tengah layar turtle. Jika Anda perlu mengetahuinya, dapatkan koordinat xy penyu tersebut dengan:

pos()

Rumah ada di (0, 0).

Dan setelah beberapa saat, mungkin akan membantu untuk menghapus jendela sehingga kita dapat memulai lagi:

clearscreen()

1. Making Algorithmic Patterns

Dengan menggunakan loop, dimungkinkan untuk membuat pola geometris:

**for** steps **in** range(100):

**for** c **in** ('blue', 'red', 'green'):

color(c)

forward(steps)

right(30)

- yang tentu saja hanya dibatasi oleh imajinasi!

Mari menggambar bentuk bintang di bagian atas halaman ini. Kami ingin garis merah, diisi dengan warna kuning:

color('red')

fillcolor('yellow')

Sama seperti up() Dan down() entukan apakah garis akan ditarik, pengisian dapat dihidupkan dan dimatikan:

begin\_fill()

Selanjutnya kita akan membuat lingkaran:

**while** **True**:

forward(200)

left(170)

**if** abs(pos()) < 1:

**break**

abs(pos()) < 1 adalah cara yang baik untuk mengetahui kapan penyu kembali ke posisi asalnya.

Terakhir, selesaikan pengisiannya:

end\_fill()

(Perhatikan bahwa pengisian hanya dilakukan saat Anda memberikan perintah end\_fill() )

1. How to…
2. Get started as quickly as possible

Salah satu keunggulan grafis turtle adalah umpan balik visual langsung yang tersedia dari perintah sederhana - ini adalah cara terbaik untuk memperkenalkan ide-ide pemrograman kepada anak-anak, dengan overhead minimal (bukan hanya anak-anak, tentu saja)..

Modul turtle memungkinkan hal ini dengan memaparkan semua fungsi dasarnya sebagai fungsi, tersedia dengan from turtle import \*.  [turtle graphics tutorial](https://docs.python.org/3/library/turtle.html#turtle-tutorial) mencakup pendekatan ini.

Perlu dicatat bahwa banyak perintah turtle juga memiliki padanan yang lebih singkat, seperti as fd() untuk [forward()](https://docs.python.org/3/library/turtle.html#turtle.forward). Hal ini sangat berguna ketika bekerja dengan pelajar yang tidak memiliki keterampilan mengetik.

Anda harus memiliki [Tk interface package](https://docs.python.org/3/library/tkinter.html#module-tkinter) diinstal pada sistem Anda agar grafis penyu dapat berfungsi. Berhati-hatilah karena hal ini tidak selalu mudah, jadi periksalah terlebih dahulu jika Anda berencana menggunakan turtle graphics dengan pelajar.

1. Use the turtle module namespace

Menggunakan from turtle import \* memang nyaman - namun berhati-hatilah karena ini mengimpor koleksi objek yang cukup besar, dan jika Anda melakukan apa pun selain grafis turtle, Anda berisiko mengalami konflik nama (ini menjadi lebih menjadi masalah jika Anda menggunakan grafis turtle di a skrip tempat modul lain mungkin diimpor).

Solusinya adalah dengan menggunakan import turtle - fd() karena turtle.fd(), width() menjadi turtle.width() dan seterusnya. (Jika mengetik “kura-kura” berulang kali menjadi membosankan, gunakan misalnya import turtle as t alih-alih.)

1. Use turtle graphics in a script

Disarankan untuk menggunakan turtle module namespace seperti yang dijelaskan tepat di atas, misalnya:

**import** **turtle** **as** **t**

**from** **random** **import** random

**for** i **in** range(100):

steps = int(random() \* 100)

angle = int(random() \* 360)

t.right(angle)

t.fd(steps)

Namun langkah lain juga diperlukan - segera setelah skrip berakhir, Python juga akan menutup jendela penyu. Menambahkan:

t.mainloop()

sampai akhir naskah. Skrip sekarang akan menunggu untuk ditutup dan tidak akan keluar sampai dihentikan, misalnya dengan menutup jendela grafis penyu.

1. Use object oriented turtle graphics

**Lihat juga**[Penjelasan antarmuka berorientasi objek](https://docs.python.org/3/library/turtle.html#turtle-explanation)

Selain untuk tujuan perkenalan yang sangat mendasar, atau untuk mencoba berbagai hal secepat mungkin, pendekatan berorientasi objek pada grafik penyu akan lebih umum dan jauh lebih efektif. Misalnya, ini memungkinkan beberapa kura-kura muncul di layar sekaligus.

Dalam pendekatan ini, berbagai perintah turtle adalah metode objek (kebanyakan berupa Penyu objek). Anda *Bisa* gunakan pendekatan berorientasi objek di shell, tetapi ini akan lebih umum dalam skrip Python.

Contoh di atas kemudian menjadi:

**from** **turtle** **import** Turtle

**from** **random** **import** random

t = Turtle()

**for** i **in** range(100):

steps = int(random() \* 100)

angle = int(random() \* 360)

t.right(angle)

t.fd(steps)

t.screen.mainloop()

Perhatikan baris terakhir. t.screen adalah contoh dari [Screen](https://docs.python.org/3/library/turtle.html#turtle.Screen) dimana ada instance Turtle; itu dibuat secara otomatis bersama dengan penyu.

Layar kura-kura dapat dikustomisasi, misalnya:

t.screen.title('Object-oriented turtle demo')

t.screen.bgcolor("orange")

1. Turtle graphics reference
2. **Turtle Methodes**
3. Turtle motion
4. Move and Draw
5. forward() | fd()

Metode turtle.forward() digunakan untuk menggerakkan turtle ke depan berdasarkan nilai argumen yang dibutuhkan. Ini memberi garis untuk berpindah ke posisi atau arah lain.

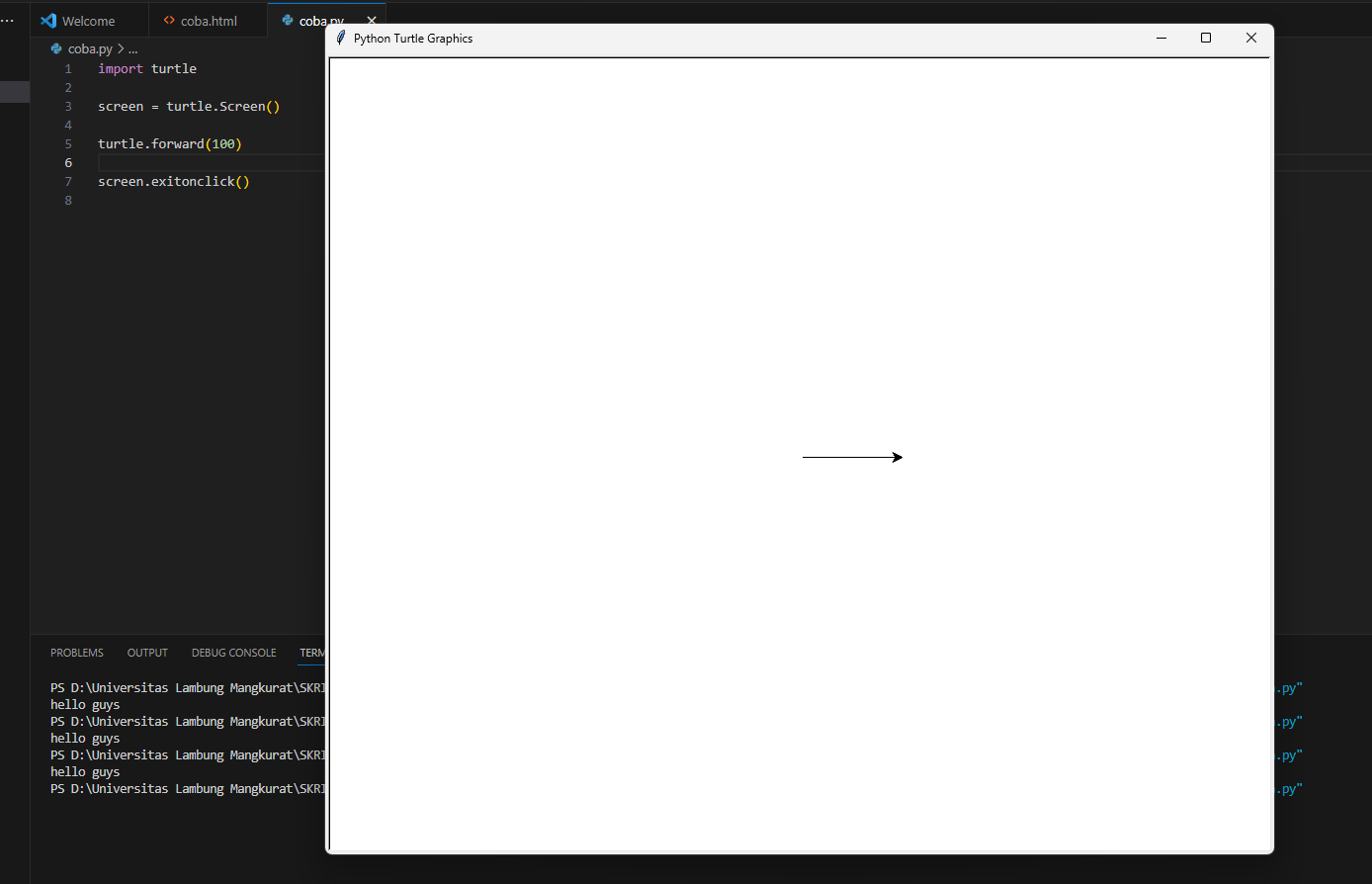
turtle.forward(*jarak*)

Argumen yang diperlukan adalah jarak { suatu bilangan (integer atau float) }. Jadi, ia menggerakkan penyu ke depan dengan jarak yang ditentukan, sesuai arah yang dituju penyu.

Contoh :

turtle.forward(100)

output :



Dalam penulisannya, kita bisa menggunakan forward() ataupun fd().

1. backward() | bk() | back()

Metode turtle.backward() digunakan untuk menggerakkan turtle ke belakang sesuai dengan nilai argumen yang dibutuhkan. Ini memberi garis untuk berpindah ke posisi atau arah lain dengan gerakan mundur.

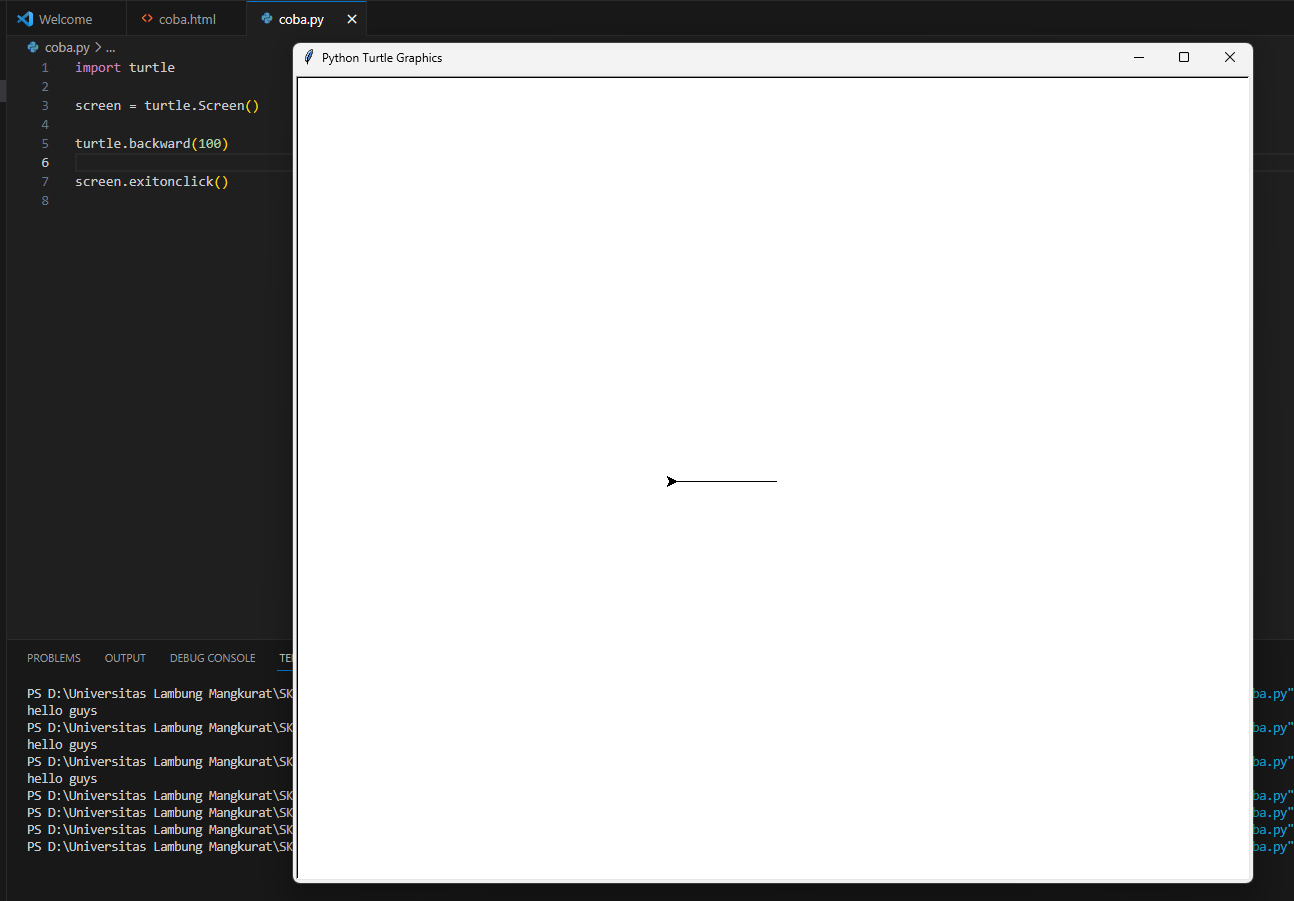
turtle.backward(*distance*)

Argumen yang diperlukan adalah jarak { suatu bilangan (integer atau float) }. Jadi, ia menggerakkan penyu ke belakang sejauh jarak yang ditentukan, berlawanan dengan arah yang dituju penyu.

Contoh :

turtle.backward(100)

output :



Dalam penulisannya, kita bisa menggunakan backward(), bk() ataupun back().

1. right() | rt()

Metode turtle.right() digunakan untuk mengubah arah turtle berdasarkan nilai argumen yang dibutuhkan. Ini memberikan pergerakan kepala kura-kura ke suatu arah.

turtle.right(*angle*)

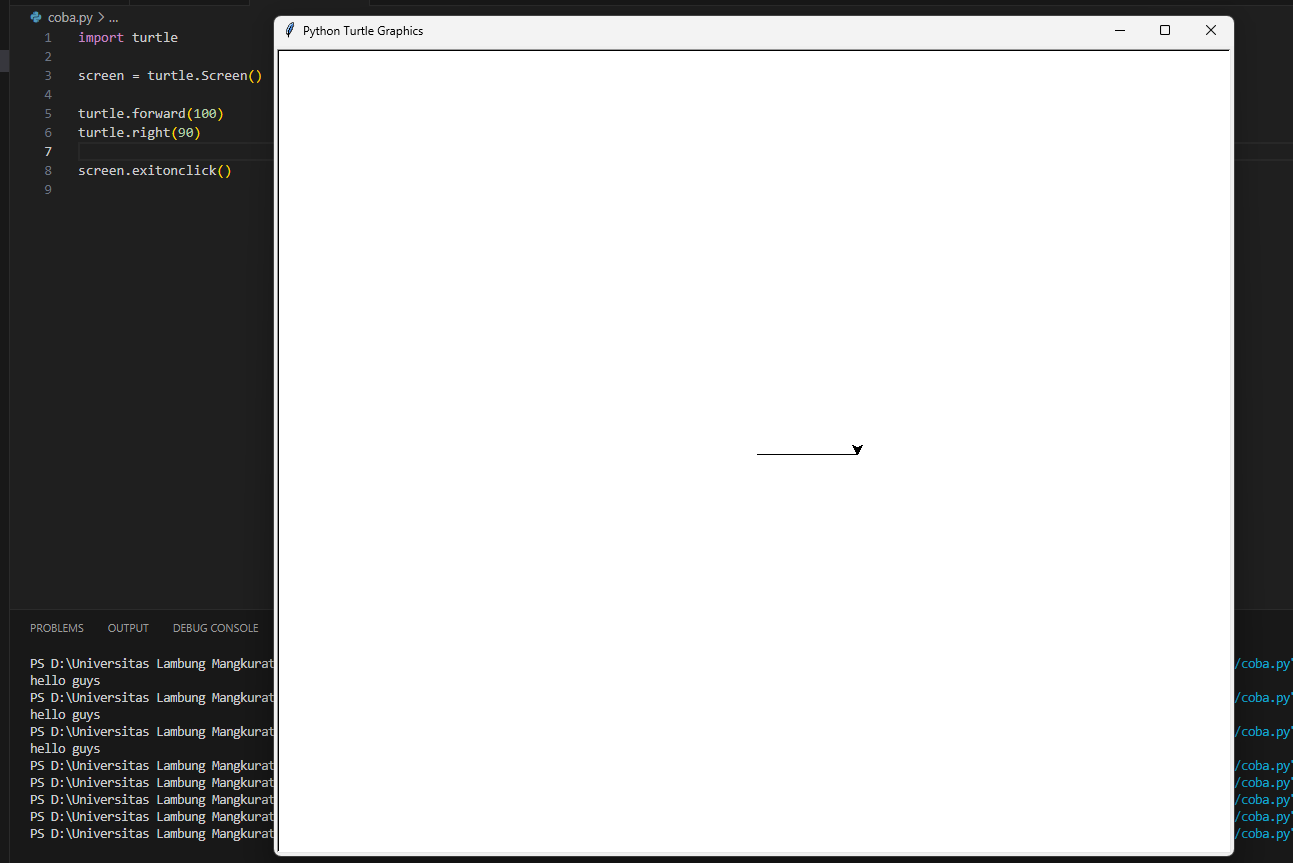
Argumen yang diperlukan adalah sudut { suatu bilangan (integer atau float) }. Jadi, ternyata penyu ke kanan berdasarkan satuan sudut. (Satuan secara default adalah derajat tetapi dapat diatur melalui fungsi derajat() dan radian().) Orientasi sudut bergantung pada mode.

Contoh 1 :

turtle.forward(100)

turtle.right(90)

output :



Pada kode tersebut kita membuat objek maju dengan jarak 100, kemudian kita mengubah arah objek sebanyak 90 derajat ke kanan, maka objek tersebut akan mengarah kebawah. Lihat apa yang terjadi Ketika kita menambahkan code forward(50) lagi setelah kode diatas.

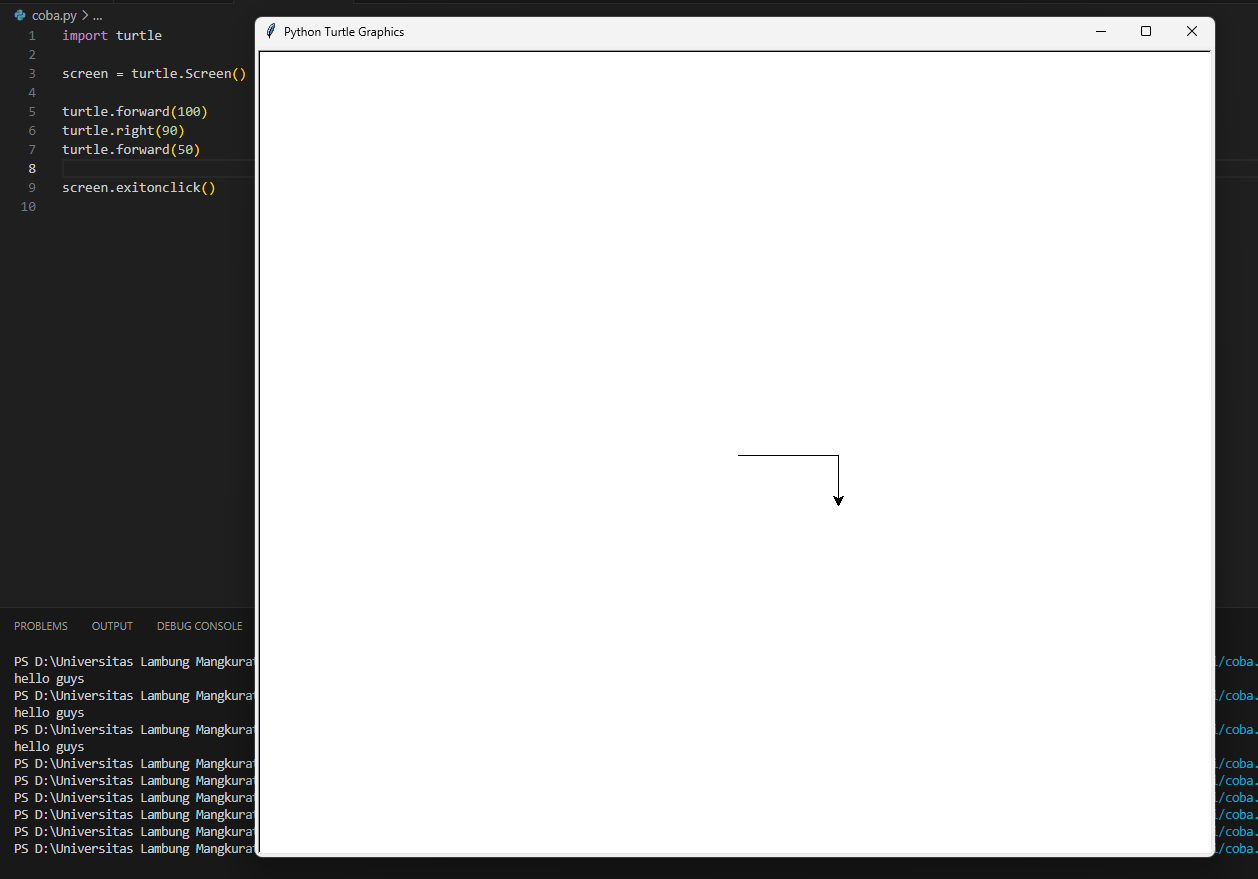
Contoh 2 :

turtle.forward(100)

turtle.right(90)

turtle.forward(50)

output :



Maka objek akan maju dengan jarak 50 sesuai dengan arah yang telah diubah sebelumnya.

1. left() | lt()

Metode turtle.left() digunakan untuk mengubah arah turtle berdasarkan nilai argumen yang dibutuhkan. Ini memberikan pergerakan kepala kura-kura ke suatu arah.

turtle.left(angle)

Argumen yang diperlukan adalah sudut { suatu bilangan (integer atau float) }. Jadi, ternyata penyu ke kiri sebesar satuan sudut. (Satuan secara default adalah derajat tetapi dapat diatur melalui fungsi derajat() dan radian().) Orientasi sudut bergantung pada mode.

Contoh :

turtle.forward(100)

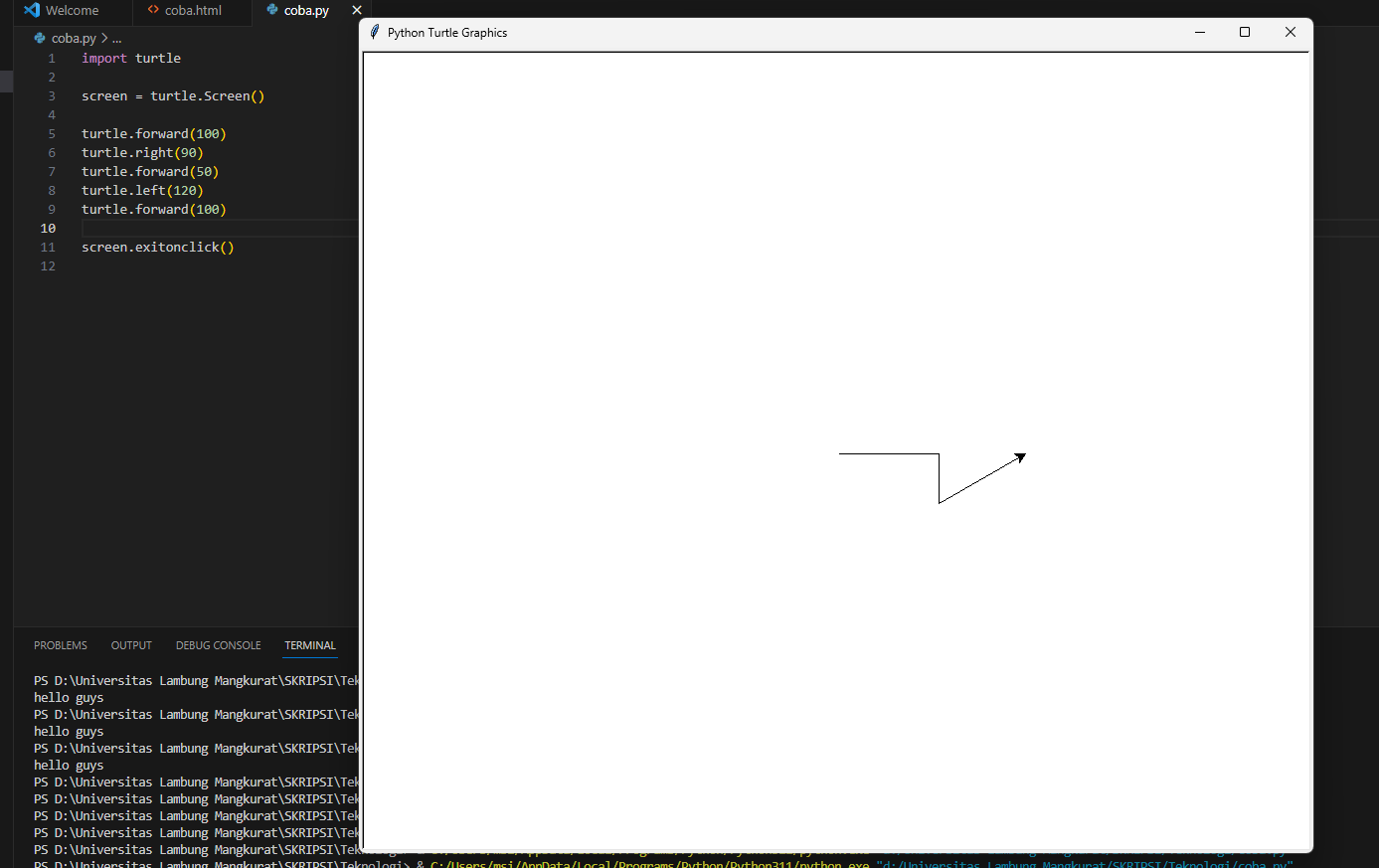
turtle.right(90)

turtle.forward(50)

**turtle.left(120)**

turtle.forward(100)

output :



1. goto() | setpos() | setposition()

Cara ini digunakan untuk menggerakkan penyu ke posisi absolut. Metode ini memiliki Alias: setpos, setposition, goto.

***Syntax:****turtle.setpos(x, y) or turtle.setposition(x, y) or turtle.goto(x, y)*

***Parameters:***

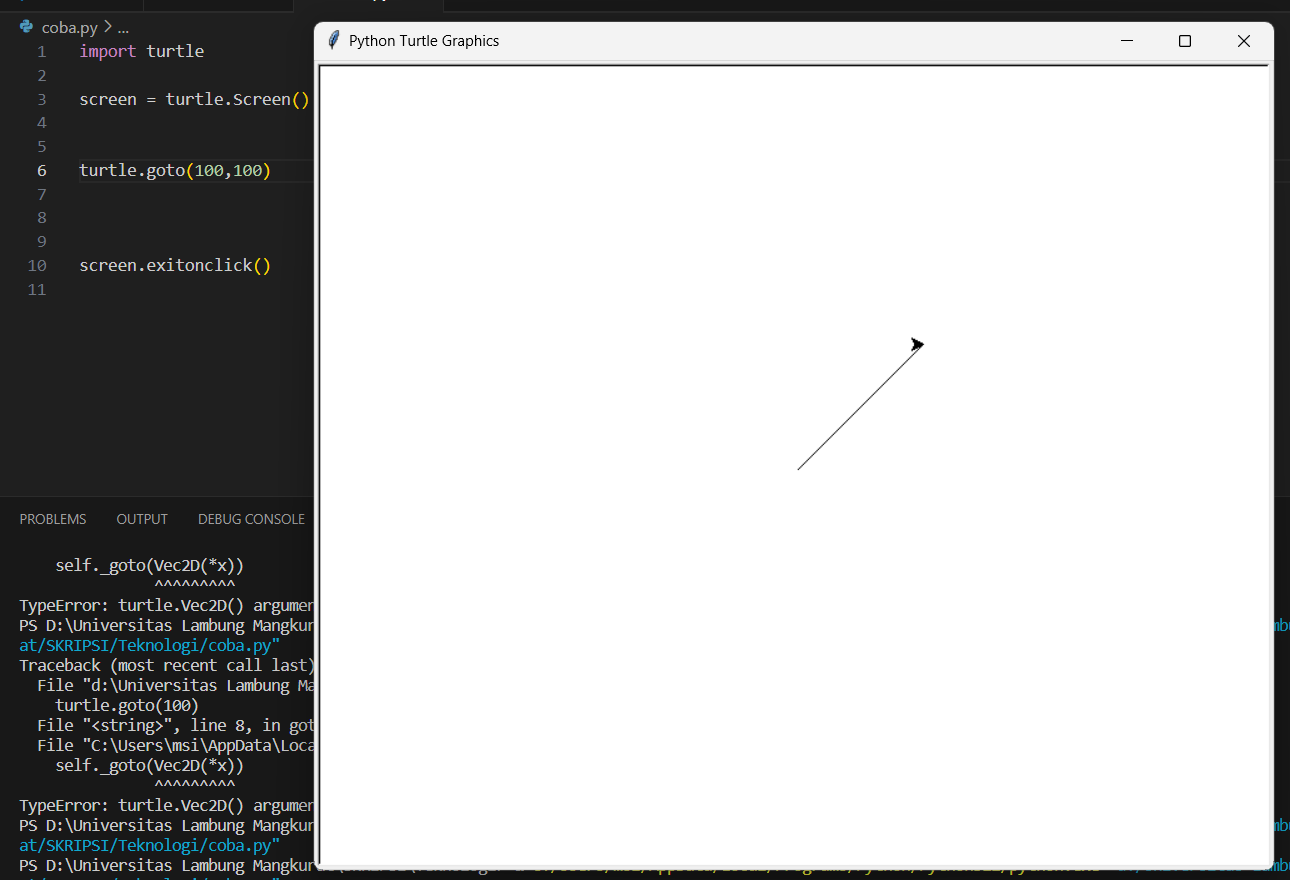
***x:****x koordinat vector 2D*

***y:****y koordinat vector 2D*

Contoh :

turtle.goto(100,100)

output :



1. setx()

Cara ini digunakan untuk mengatur koordinat pertama turtle ke x, biarkan koordinat kedua tidak berubah. Di sini, apa pun posisi kura-kura, atur koordinat x ke masukan yang diberikan dengan menjaga koordinat y tidak berubah.

***Syntax:****turtle.setx(x)*

***Parameter:***

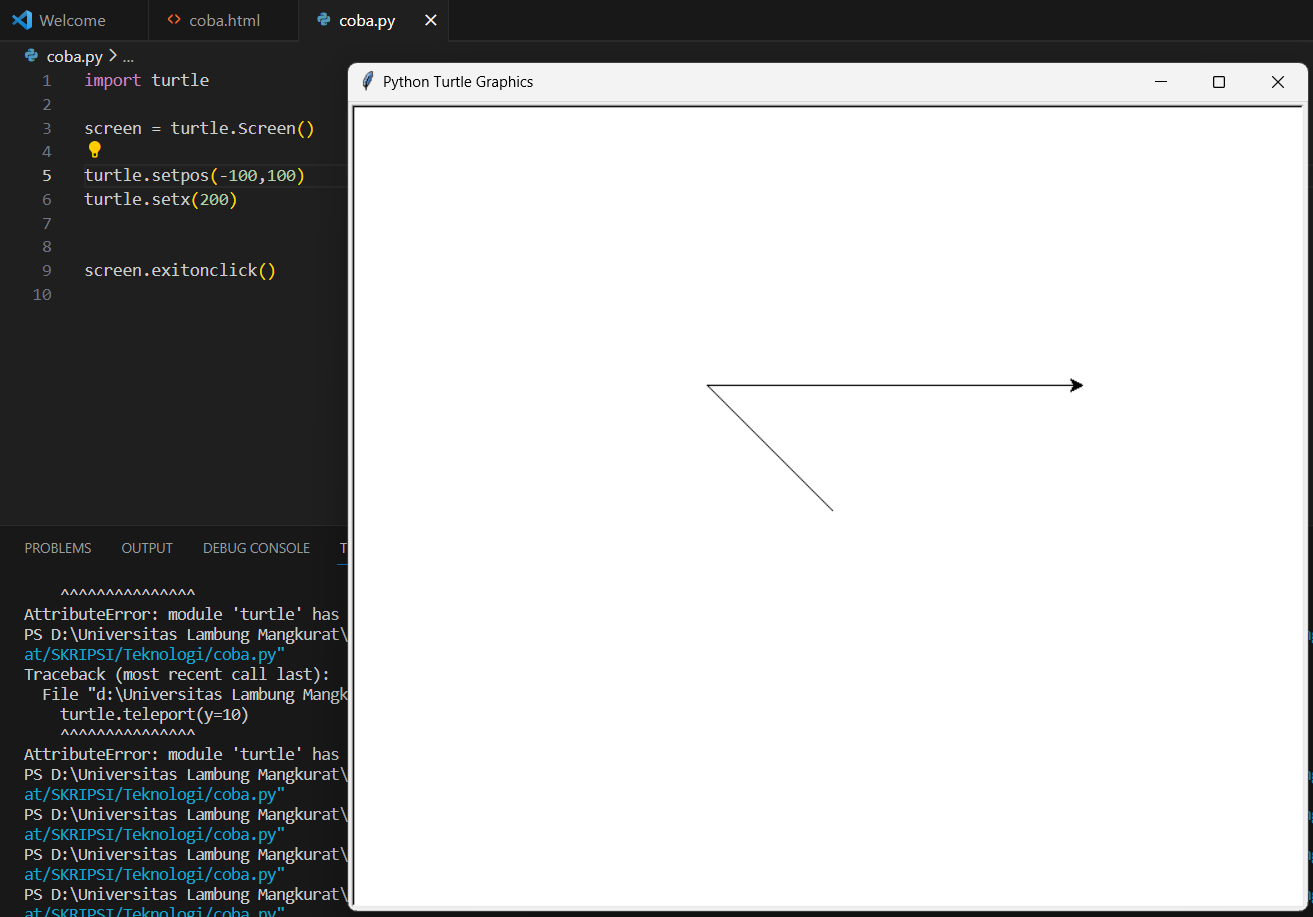
***x:****angka (integer atau float), itu adalah satu-satunya argumen yang diperlukan.*

contoh :

turtle.setpos(-100,100)

turtle.setx(200)

output :



pada kode baris pertama kita menempatkan posisi objek di (-100,100) setelah itu kita ubah nilai x yang awalnya -100 menjadi 200 menggunakan setx(200) maka objek akan langsung berpindah ke koordinat x 200 tanpa mengubah koordinat y.

1. sety()

Cara ini digunakan untuk mengatur koordinat pertama turtle ke x, biarkan koordinat kedua tidak berubah. Di sini, apa pun posisi kura-kura, atur koordinat x ke masukan yang diberikan dengan menjaga koordinat y tidak Metode ini digunakan untuk mengatur koordinat kedua turtle ke y, sehingga koordinat pertama tidak berubah. Di sini, apa pun posisi kura-kura, atur koordinat y ke masukan yang diberikan dengan menjaga koordinat x tidak berubah.

***Syntax:****turtle.sety(y)*

***Parameter:***

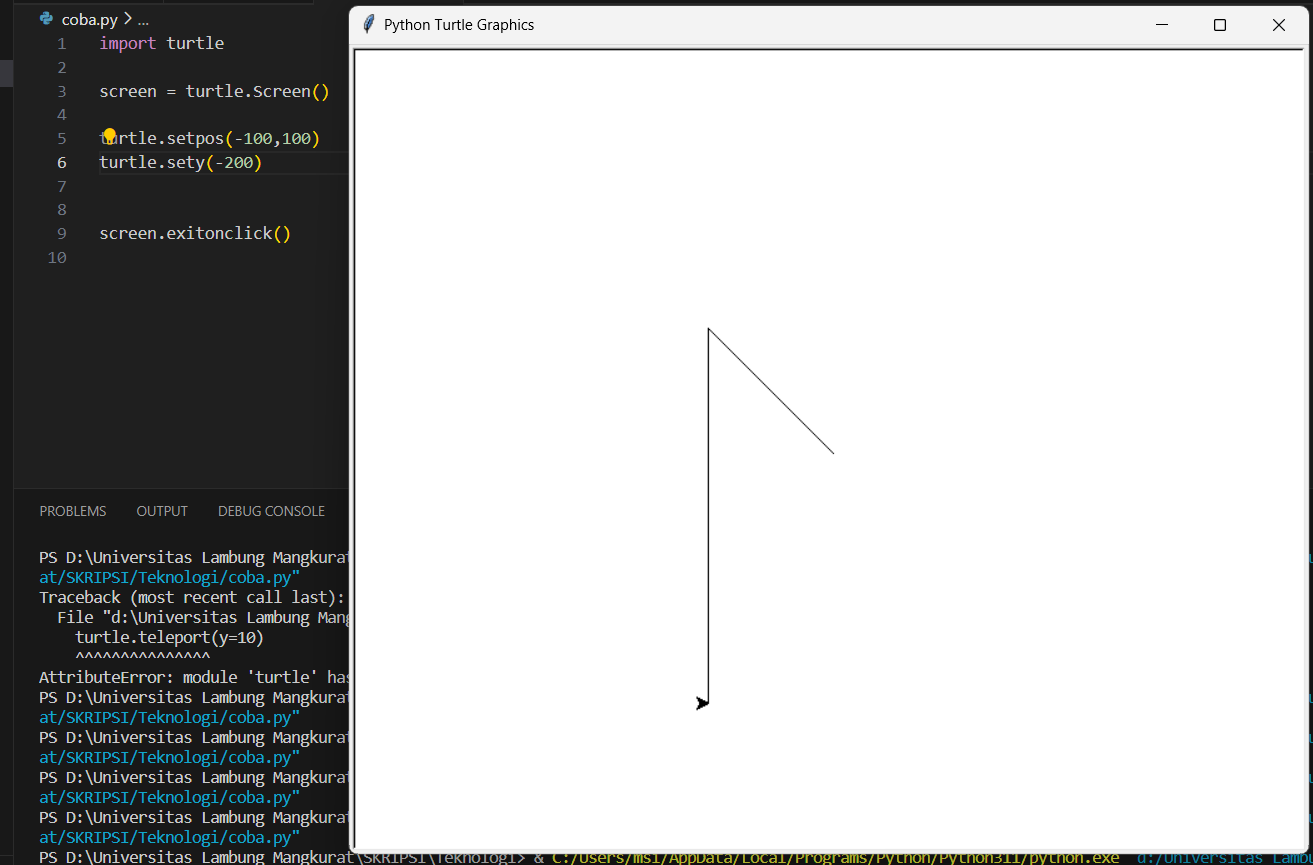
***y:****angka (integer atau float), itu adalah satu-satunya argumen yang diperlukan..*

contoh :

turtle.setpos(-100,100)

turtle.sety(-200)

output :



pada kode baris pertama kita menempatkan posisi objek di (-100,100) setelah itu kita ubah nilai y yang awalnya 100 menjadi -200 menggunakan sety(-200) maka objek akan langsung berpindah ke koordinat y -200 tanpa mengubah koordinat x.

1. setheading() | seth()

Atur orientasi penyu ke arah derajat. Berikut adalah beberapa arah umum dalam derajat:

|  |  |
| --- | --- |
| Standar Mode | Logo Mode |
| 0 – Timur | 0 – Timur |
| 90 – Utara | 90 – Utara |
| 180 – Barat | 180 - Selatan |
| 270 - Selatan | 270 - Barat |

***Syntax:****turtle.setheading(to\_angle)*

***Parameter:***

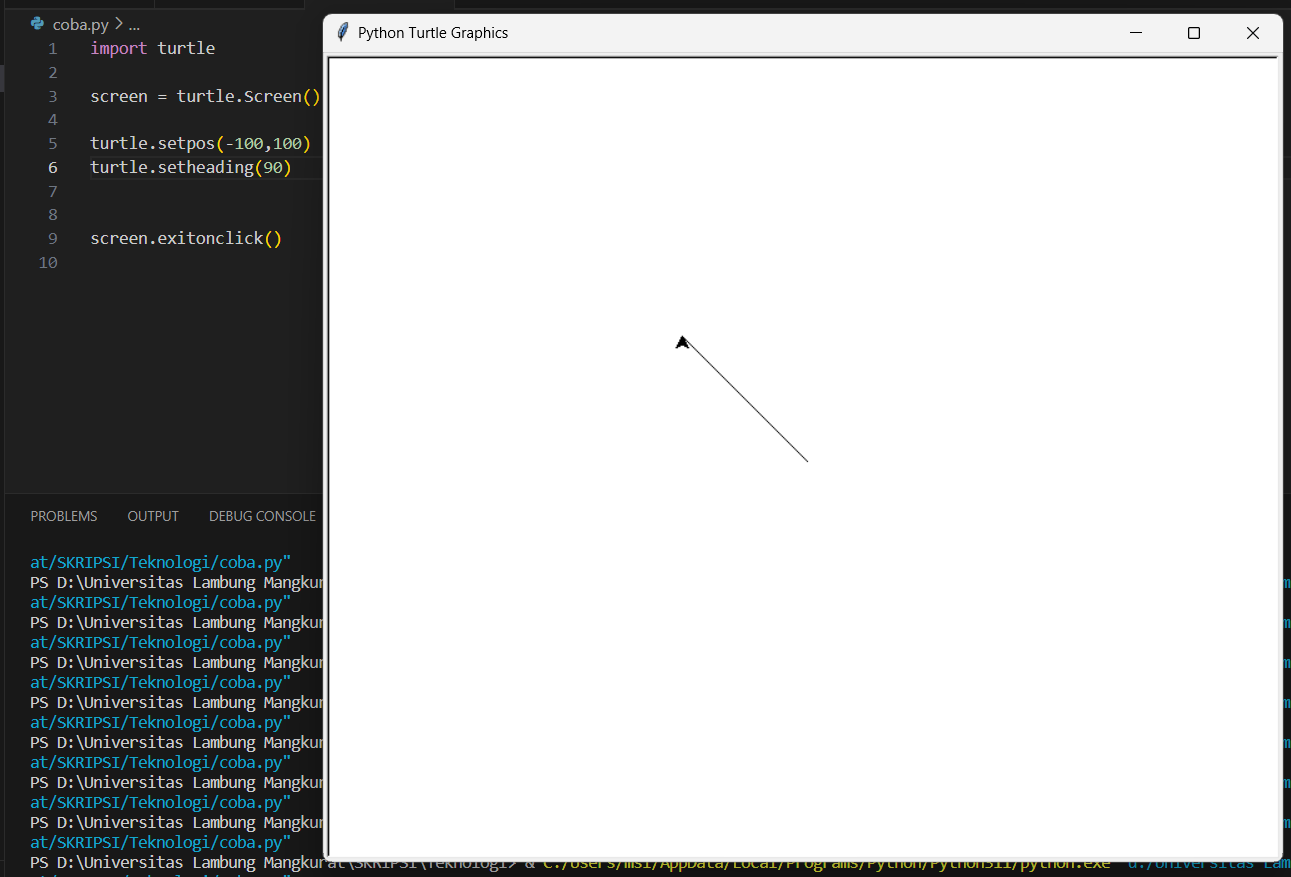
***To\_angle:****angka (integer atau float)*

Contoh :

turtle.setpos(-100,100)

turtle.setheading(90)

output :

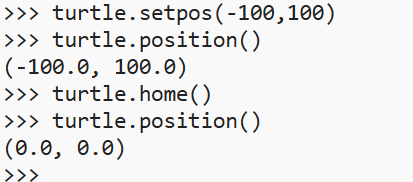


Pada standar mode, defaultnya arah objek adalah 0 atau mengarah ke timur. Ketika kita menggantinya menjadi 90, maka arah objek tersebut akan berubah menghadap utara.

1. home()

Pindahkan turtle ke titik asal – koordinat (0,0) – dan atur arahnya ke orientasi awal.

Contoh :



Posisi objek saat kita menggunakan fungsi setpos(-100,100) adalah (x = -100, y=100). Kemudian saat kita menggunakan fungsi home() maka posisi objek akan Kembali ke titik awal yaitu (x=0, y=0)

1. circle()

Cara ini digunakan untuk menggambar lingkaran dengan radius tertentu.

***Syntax:****turtle.circle(radius, extent=None, steps=None)*

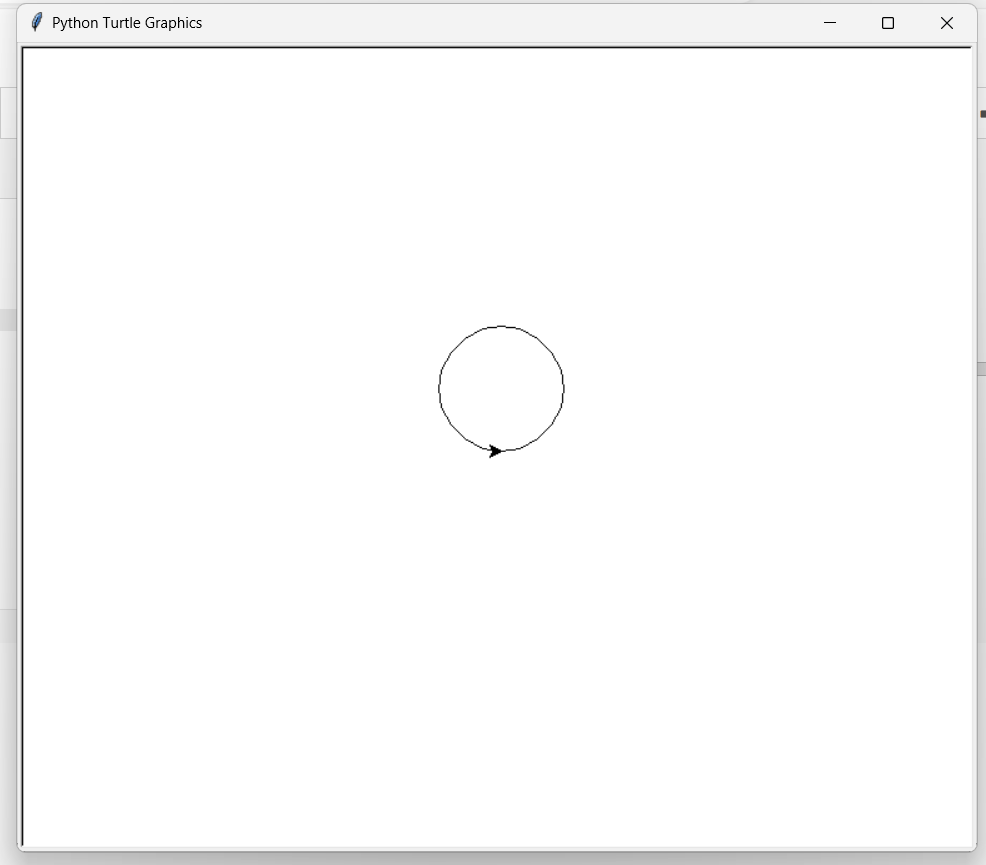
***Parameters:***

* ***radius:****Jari-jari lingkaran.*
* ***extent:****Bagian lingkaran dalam derajat sebagai busur.*
* ***steps:****Bagilah bentuk tersebut ke dalam jumlah langkah yang sama.*

Contoh 1 : (menggambar circle dengan radius 50

turtle.circle(50)

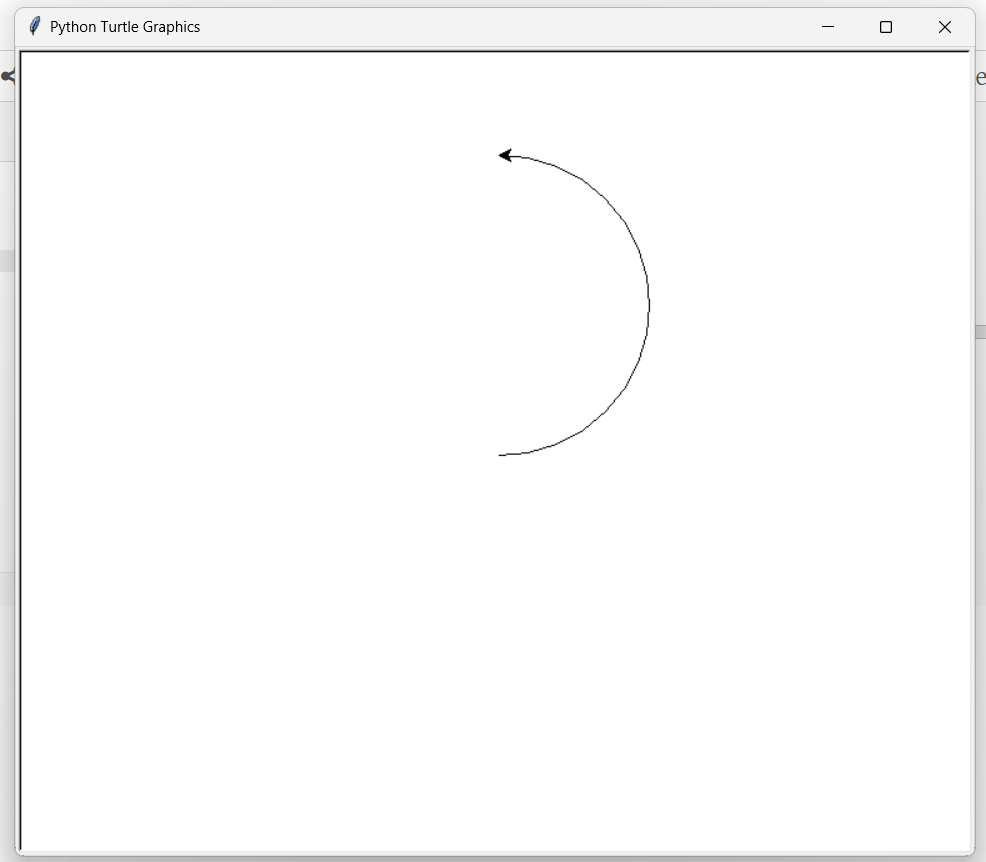
output :



Contoh 2 : (menggambar lingkaran dengan radius 120 dan extend 180)

turtle.circle(120, 180)

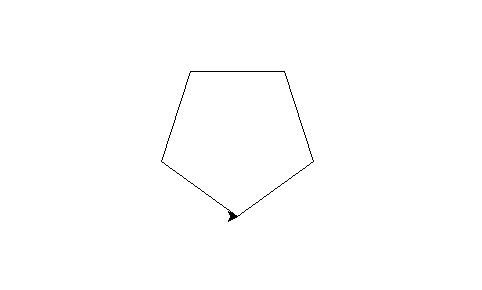
output :



Contoh 3 : (menggambar lingkaran dengan radius 80 dan step 5)

turtle.circle(80,steps=5)

output :



1. dot()

Fungsi ini digunakan untuk menggambar titik melingkar dengan ukuran tertentu, dengan warna tertentu. Jika ukuran tidak dicantumkan, maka yang digunakan maksimal pensize+4 dan 2\*pensize.

**Syntax :**

turtle.dot(size=None, \*color)

**Parameters:**

* **size** : bilangan bulat >= 1 (jika diberikan)
* **color** : rangkaian warna atau tupel warna numerik

contoh :

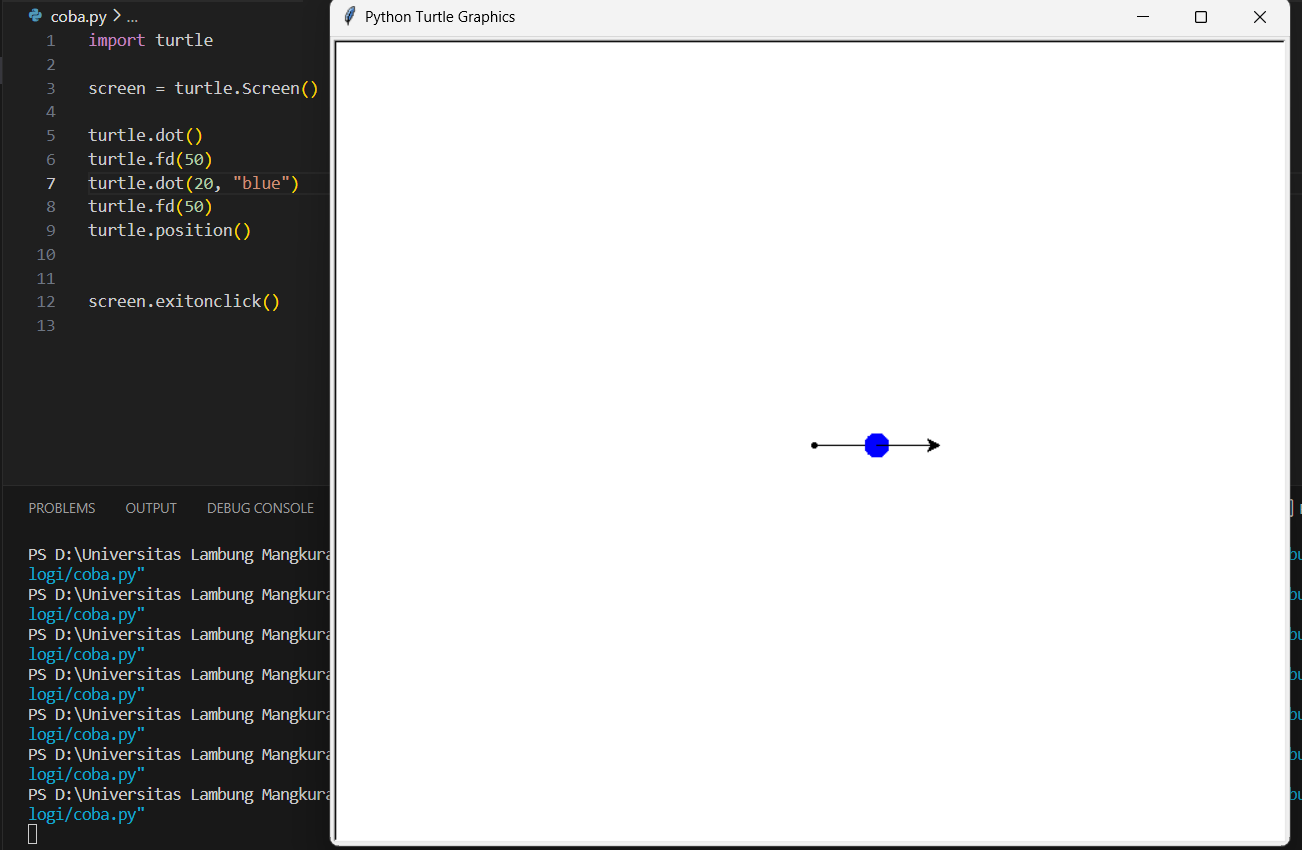
turtle.dot()

turtle.fd(50)

turtle.dot(20, “blue”)

turtle.fd(50)

output :



1. stamp()

Metode ini digunakan untuk mencap salinan bentuk penyu ke kanvas dan mengembalikan idnya. Itu tidak memerlukan argumen apa pun. Apapun bentuk kura-kuranya, dicetak pada titik itu dan dilanjutkan dengan instruksi selanjutnya.

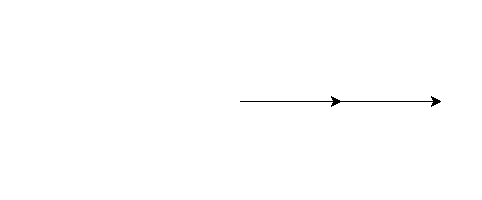
Contoh :

turtle.forward(100)

turtle.stamp()

turtle.forward(100)

output :



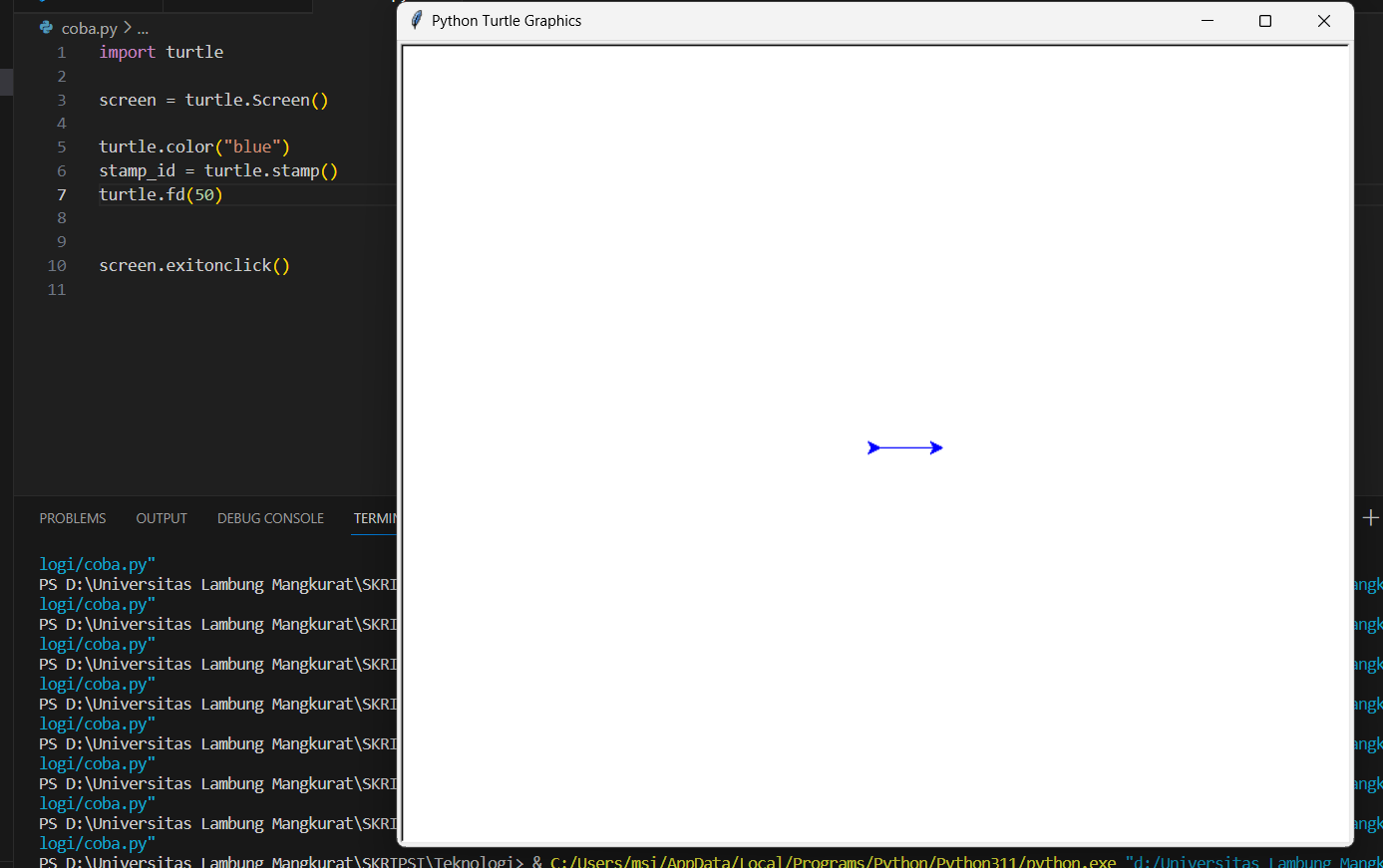
Contoh 2 :

turtle.color(“blue”)

Stamp\_id = turtle.stamp()

turtle.fd(50)

Output:



1. clearstamps()

Metode turtle.clearstamp() digunakan untuk menghapus seluruh atau n stempel penyu pertama/terakhir. Metode ini memerlukan argumen bilangan bulat. Jadi, stempel yang dibuat dengan id dihapus olehnya.

***Syntax :****turtle.clearstamp(stampid)*

***Parameter:***

***Stampid :*** *bilangan bulat, harus berupa nilai kembalian dari panggilan stamp() sebelumnya.*

Contoh :

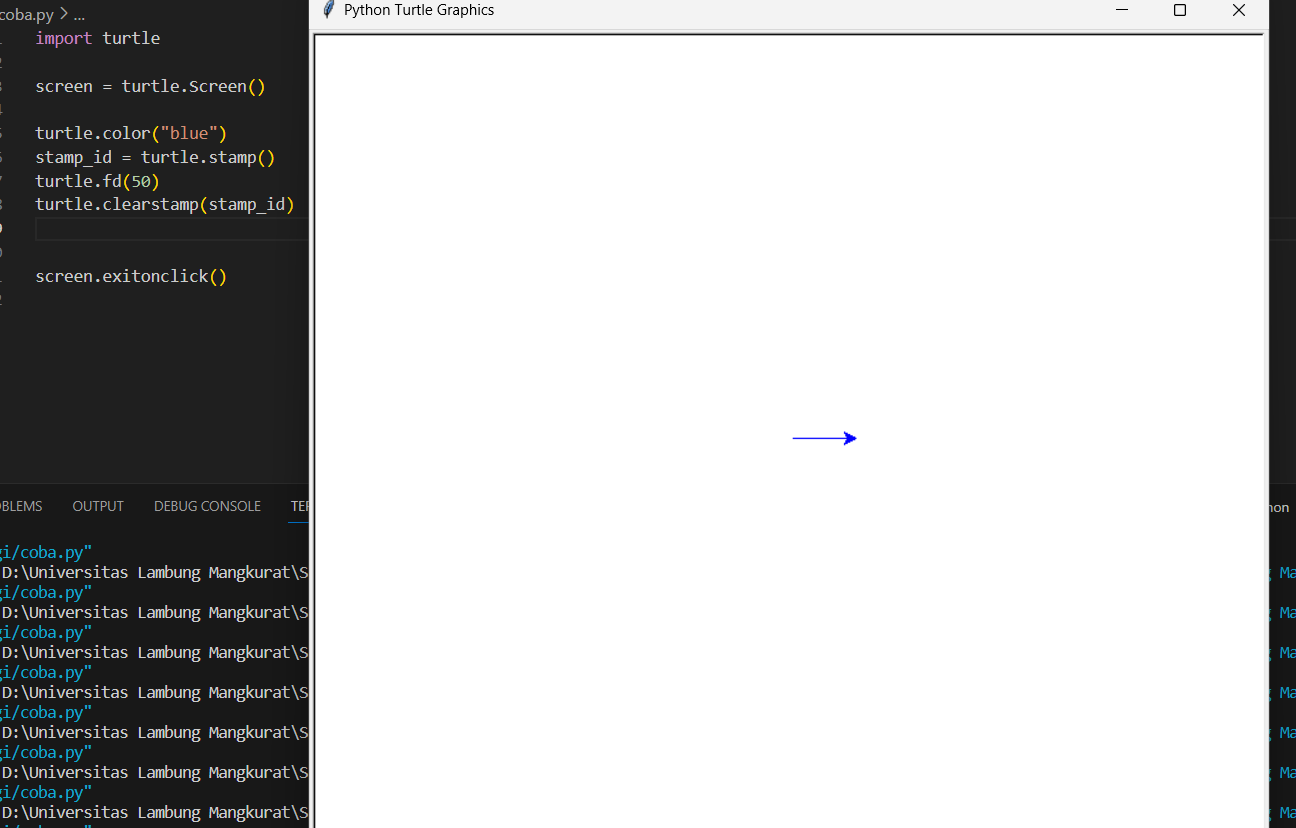
turtle.color(“blue”)

stamp\_id = turtle.stamp()

turtle.fd(50)

turtle.clearstamp(stamp\_id)

Output :



1. undo()

Fungsi ini digunakan untuk membatalkan (berulang kali) tindakan penyu terakhir. Jumlah tindakan undo yang tersedia ditentukan oleh ukuran undobuffer. Itu tidak memerlukan argumen apa pun.

**Syntax :**

turtle.undo()

contoh :

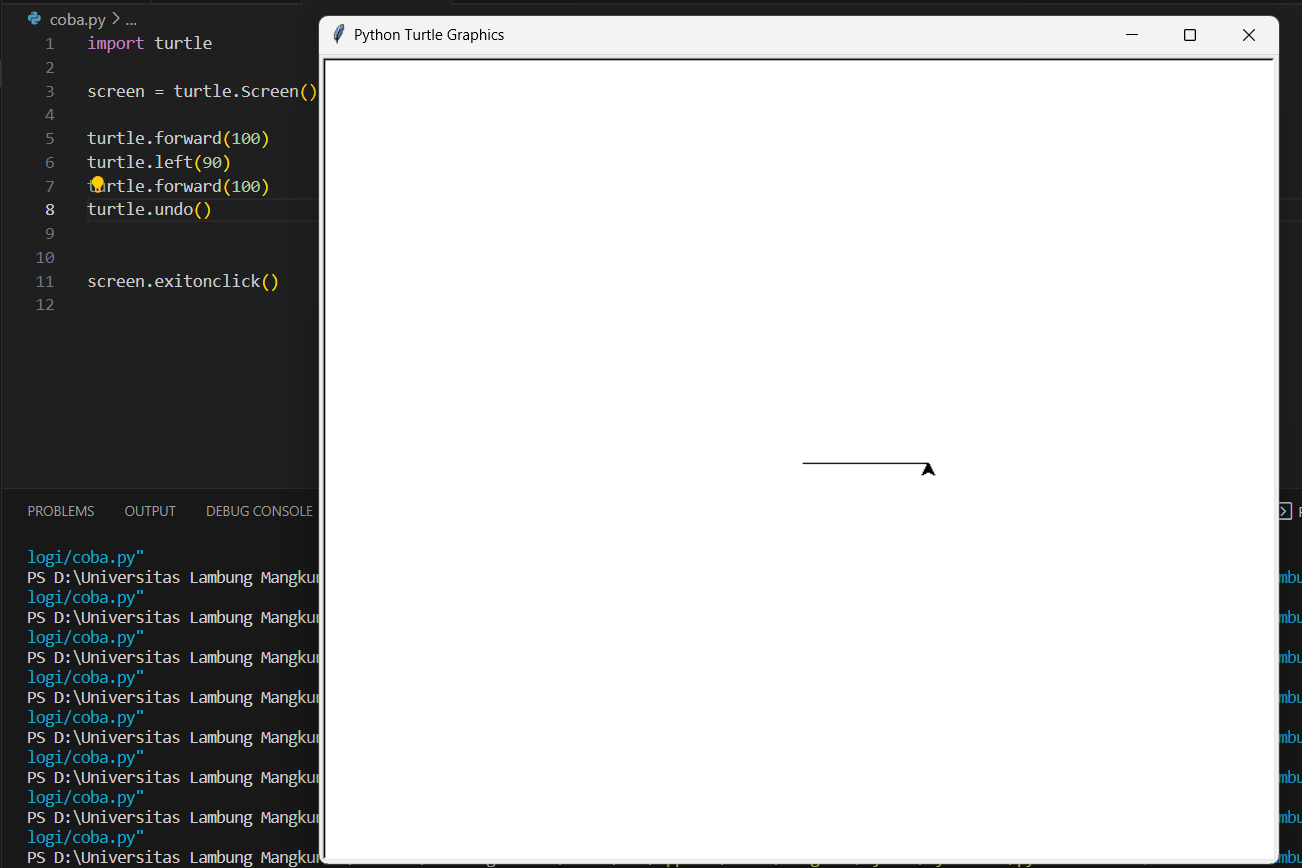
turtle.forward(100)

turtle.left(90)

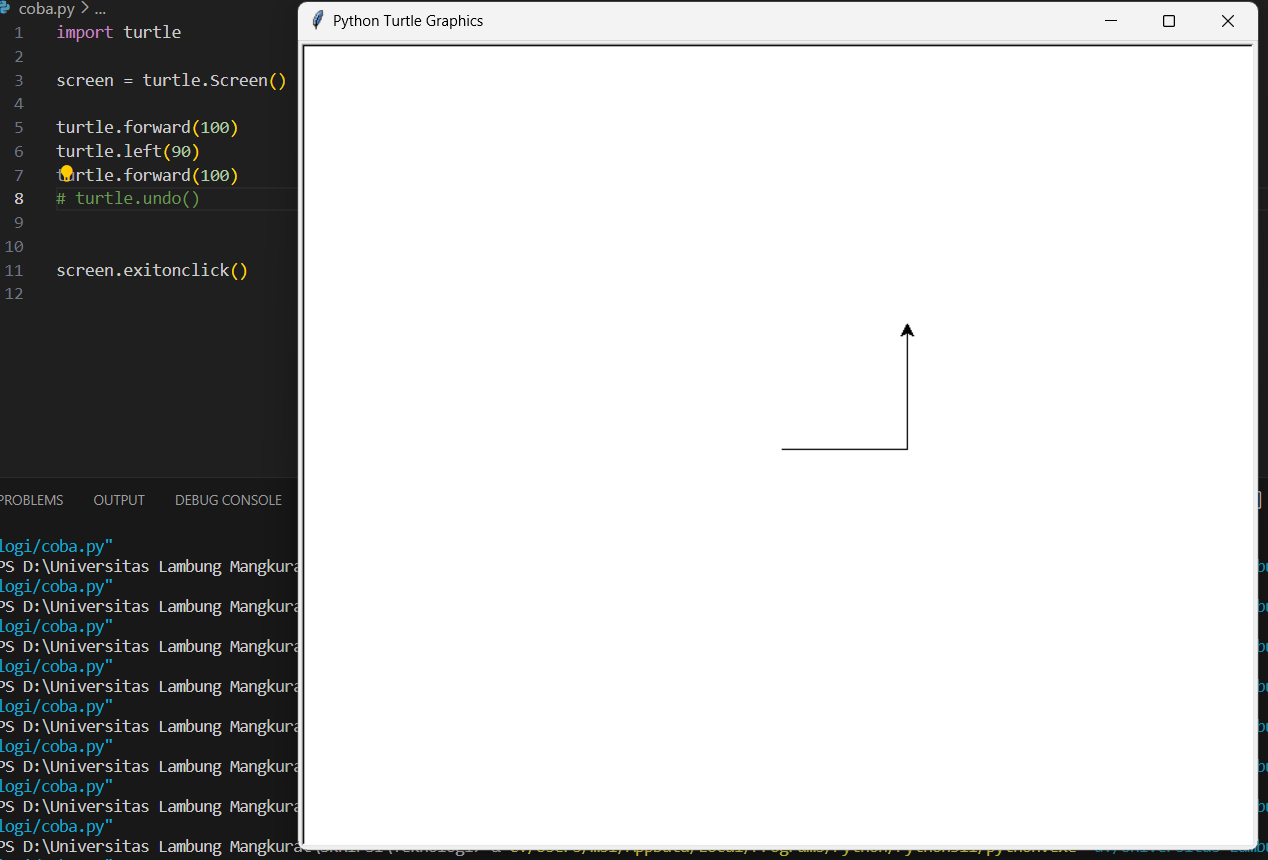
turtle.forward(100)

turtle.undo()

Output :



Ketika fungsi undo() kita hapus maka hasilnya akan seperti dibawah ini :



1. speed()

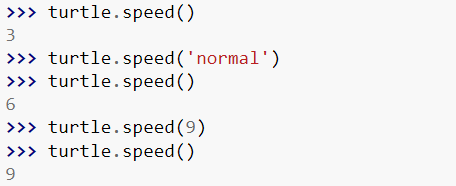
Metode turtle.speed() digunakan untuk mengubah kecepatan turtle berdasarkan nilai argumen yang dibutuhkan. Kembalikan atau atur kecepatan penyu.

**Syntax :**

turtle.speed(speed=None)

**Note:**

* Kecepatan penyu berada pada kisaran 0-10.
* Jika input berupa angka lebih besar dari 10 atau lebih kecil dari 0,5, kecepatan diatur ke 0.
* Speedstring  dipetakan ke nilai kecepatan dengan cara berikut:
  + ‘fastest’ :  0
  + ‘fast’    :  10
  + ‘normal’  :  6
  + ‘slow’    :  3
  + ‘slowest’ :  1
* Kecepatan dari 1 hingga 10 membuat animasi menggambar garis dan memutar penyu semakin cepat.



1. Tell Turtle’s State
   * 1. position() | pos()

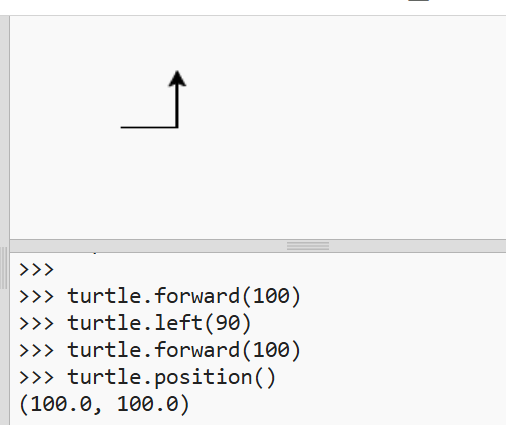
Metode ini digunakan untuk mencari lokasi penyu saat ini (x, y), sebagai vektor Vec2D. Metode ini memiliki Alias: pos | posisi.

***Syntax:****turtle.pos() or  turtle.position()*

***Return:****lokasi penyu saat ini dalam koordinat (x, y).*

Fungsi ini tidak memerlukan argumen apa pun dan mengembalikan posisi kura-kura saat ini dalam format (x,y) di mana x dan y mewakili vektor 2D. Nilai defaultnya adalah (0,0, 0,0).

Contoh :



* + 1. towards()

Fungsi ini digunakan untuk mengembalikan sudut antara garis dari posisi penyu ke posisi yang ditentukan oleh x, y, dan orientasi awal penyu.

**Syntax :**

turtle.towards(x, y=None)

**Parameters:**

* **x** – suatu bilangan atau sepasang/vektor bilangan atau contoh penyu
* **y** – suatu bilangan jika x adalah suatu bilangan, jika tidak None

contoh :

# go towards east

ang**=**turtle.towards(100,0)

# print angle

**print**(ang)

# go towards north

ang**=**turtle.towards(0,100)

# print angle

print(ang)

# go towards west

ang**=**turtle.towards(**-**100,0)

# print angle

print(ang)

# go towards south

ang**=**turtle.towards(0,**-**100)

# print angle

print(ang)

output :

0.0

90.0

180.0

270.0

* + 1. xcor()

Fungsi ini digunakan untuk mengembalikan koordinat x penyu dari posisi penyu saat ini. Itu tidak memerlukan argumen apa pun.

**Syntax :**

turtle.xcor()

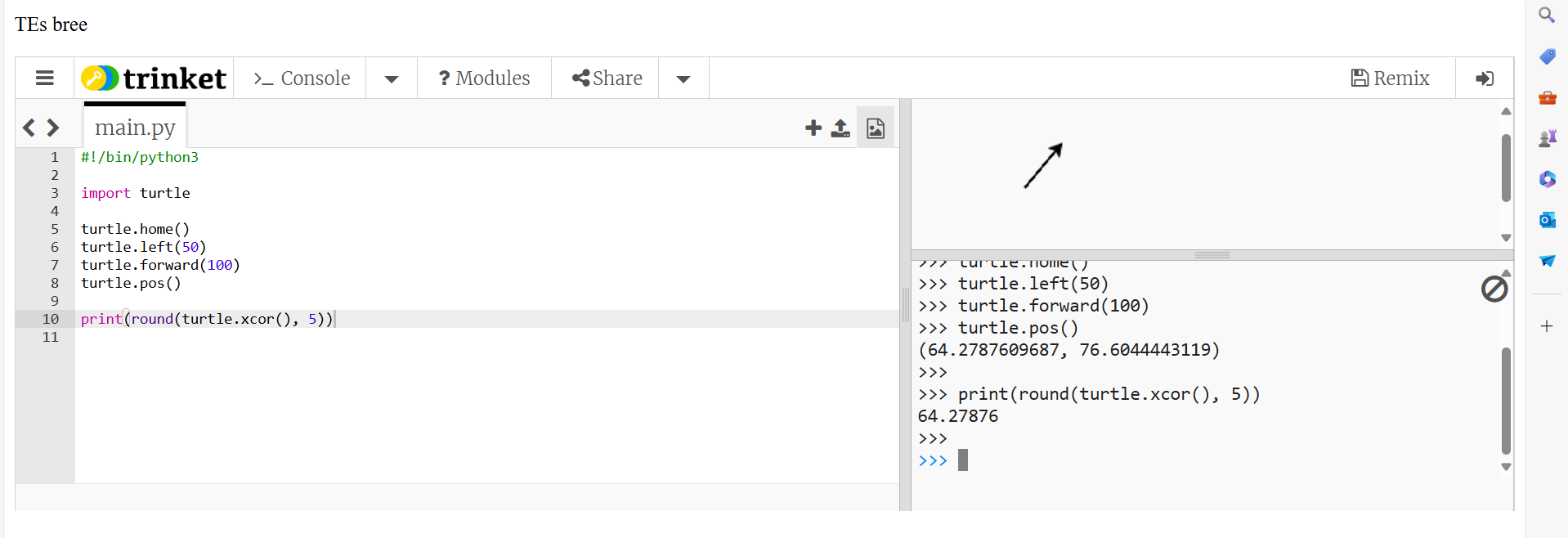
contoh :

turtle.left(50)

turtle.forward(100)

print(round(turtle.xcor(),5))

output :



* + 1. ycor()

Fungsi ini digunakan untuk mengembalikan koordinat y penyu dari posisi penyu saat ini. Itu tidak memerlukan argumen apa pun.

**Syntax :**

turtle.ycor()

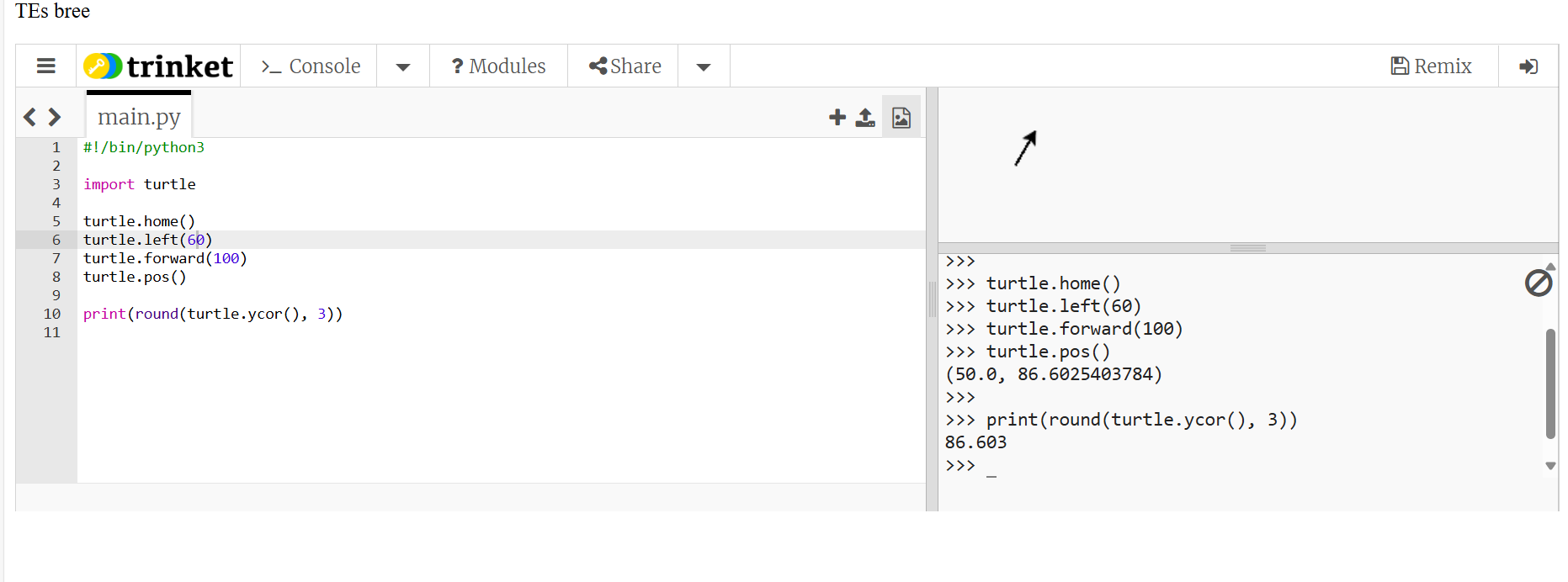
contoh :

turtle.left(60)

turtle.forward(100)

print(round(turtle.ycor(),3))

output :



* + 1. heading()

Fungsi ini digunakan untuk mengembalikan arah penyu saat ini. Itu tidak memerlukan argumen apa pun.

**Syntax :**

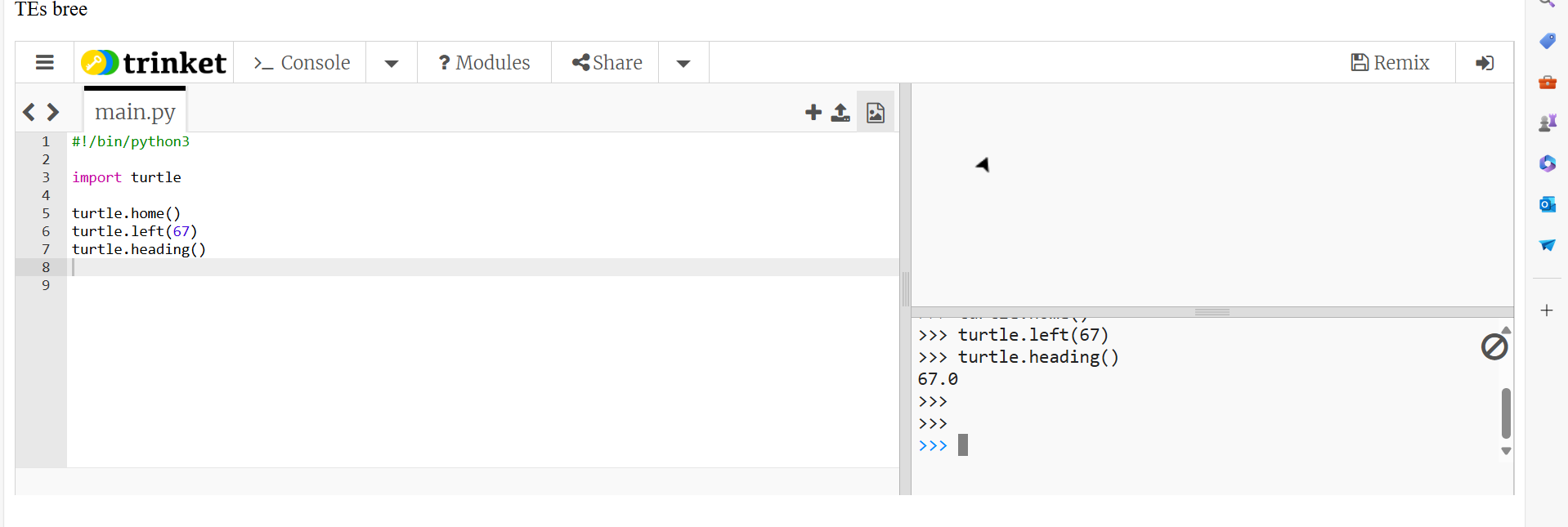
turtle.heading()

contoh :

turtle.left(47)

turtle.heading()

output :



* + 1. distance()

Cara ini digunakan untuk mengembalikan jarak dari penyu ke (x,y) dalam satuan langkah penyu.

***Syntax:****turtle.distance(x,y=None)*

***Parameters:***

***x:****x koordinat vector 2D.*

***y:****y koordinat vector 2D.*

Metode ini dapat dipanggil dalam format berbeda seperti yang diberikan di bawah ini:

* distance(x, y) # dua koordinat
* distance((x, y)) # sepasang (tupel) koordinat
* distance(vec) # e.g. seperti yang dikembalikan oleh pos()
* distance(mypen) # dimana mypen adalah kura-kura lainnya

contoh :

joe = turtle.Turtle()

joe.forward(77)

turtle.distance(joe)

output :

77

1. Setting and Measurement
2. degrees()

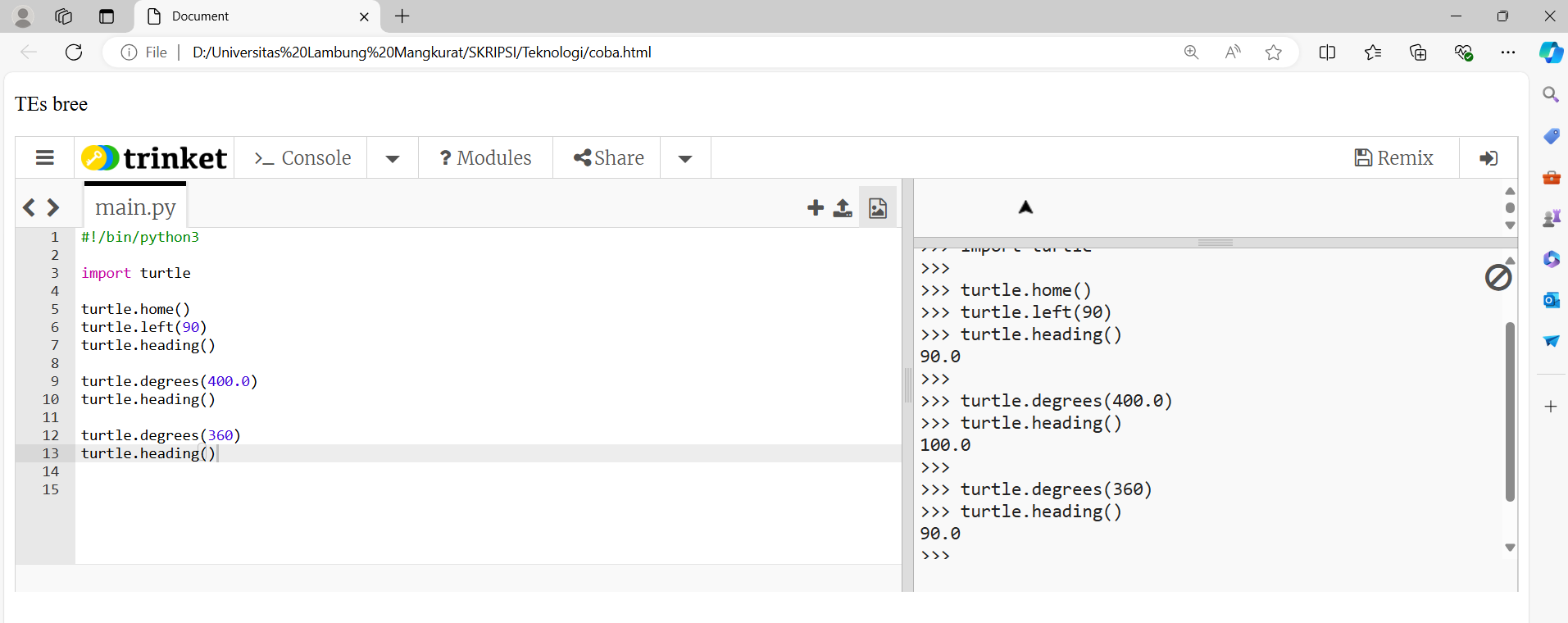
Cara ini digunakan untuk mengatur satuan pengukuran sudut menjadi derajat. Secara default, satuan pengukuran sudut adalah “derajat”.

***Syntax:****turtle.degrees(fullcircle=360.0)*

***Parameter:***

***fullcircle(optional):****sebuah angka. Tetapkan satuan pengukuran sudut, yaitu menetapkan jumlah 'derajat' untuk satu lingkaran penuh. Nilai defaultnya adalah 360 derajat.*

Contoh :

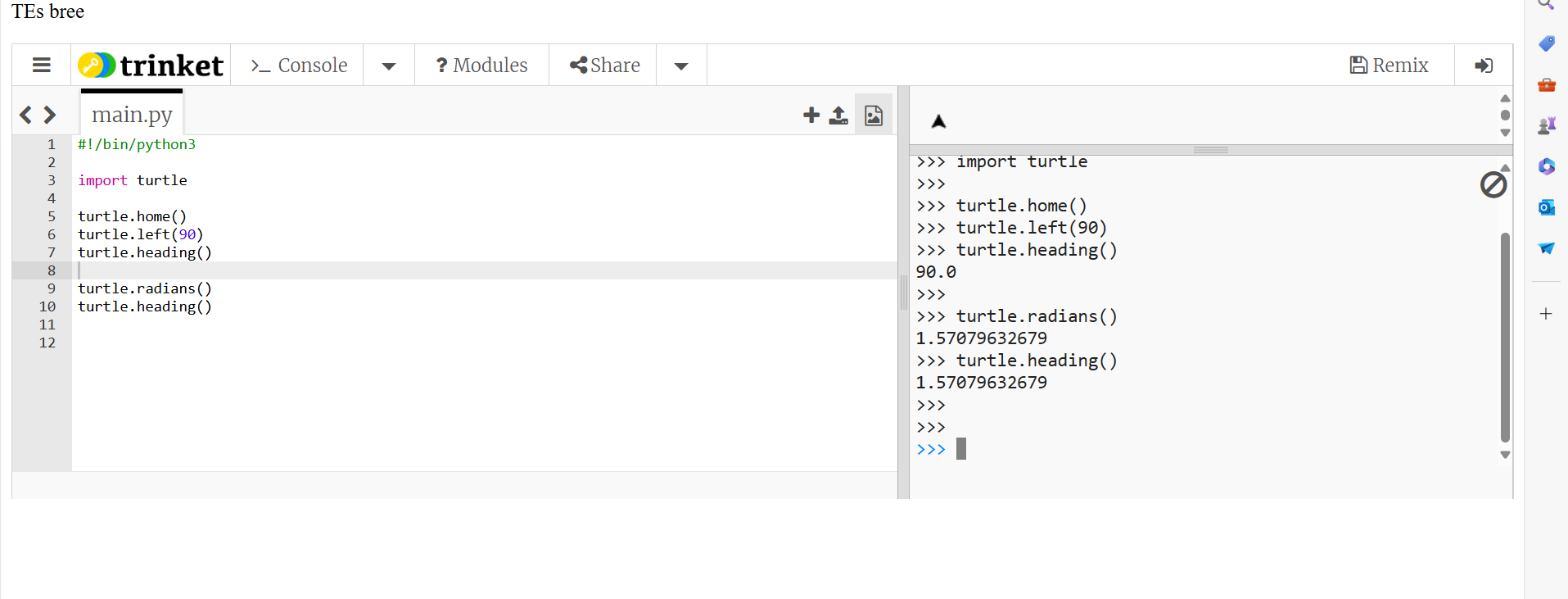


1. radians()

Cara ini digunakan untuk mengatur satuan ukuran sudut menjadi radian. Itu tidak memerlukan argumen apa pun. Secara default, satuan pengukuran sudut adalah “derajat”.

***Syntax:****turtle.radians()****Parameter:****None****Returns :****None*

Contoh :



1. Pen Control
2. Drawing State
3. pendown() | pd() | down()

Metode turtle.down() digunakan untuk menarik kembali pena ke bawah pada layar. Ini memberi gambaran tentang perpindahan ke posisi atau arah lain.

Di sini metode ini dapat dipanggil dengan tiga nama seperti yang tertulis di atas yaitu; ia memiliki Alias: pendown | pd | down. Tidak ada argumen yang diperlukan untuk metode ini.

Contoh :

turtle.forward(50)

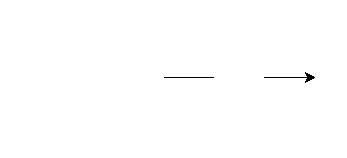
turtle.up()

turtle.forward(50)

turtle.down()

turtle.forward(50)

output :



1. penup() | pu() | up()

Metode turtle.up() digunakan untuk menarik pena dari layar. Tidak ada gambaran untuk berpindah ke posisi atau arah lain.

Di sini metode ini dapat dipanggil dengan tiga nama seperti yang tertulis di atas yaitu; ia memiliki Alias: penup | pu | up. Tidak ada argumen yang diperlukan untuk metode ini.

Contoh :

turtle.forward(50)

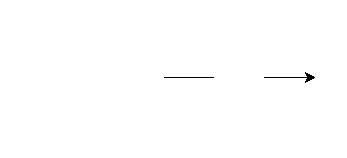
turtle.up()

turtle.forward(50)

turtle.down()

turtle.forward(50)

output :



Pada kode tersebut ketika kita menuliskan fungsi up() maka pena akan di angkat dan ketika kita memindahkan objeknya tidak akan membentuk Gambaran, setelah kita gunakan fungsi down() maka ketika kita memindahkan objek lagi maka akan membentuk gambar lagi.

1. pensize() | width()

Cara ini digunakan untuk mengatur atau mengembalikan ketebalan garis. Atur ketebalan garis menjadi lebar atau kembalikan. Jika resizemode diatur ke “auto” dan turtleshape adalah poligon, poligon tersebut digambar dengan ketebalan garis yang sama. Jika tidak ada argumen yang diberikan, ukuran pensize saat ini dikembalikan.

**Syntax :**

turtle.width(width=None)

turtle.pensize(width=None)

contoh :

turtle.pensize()

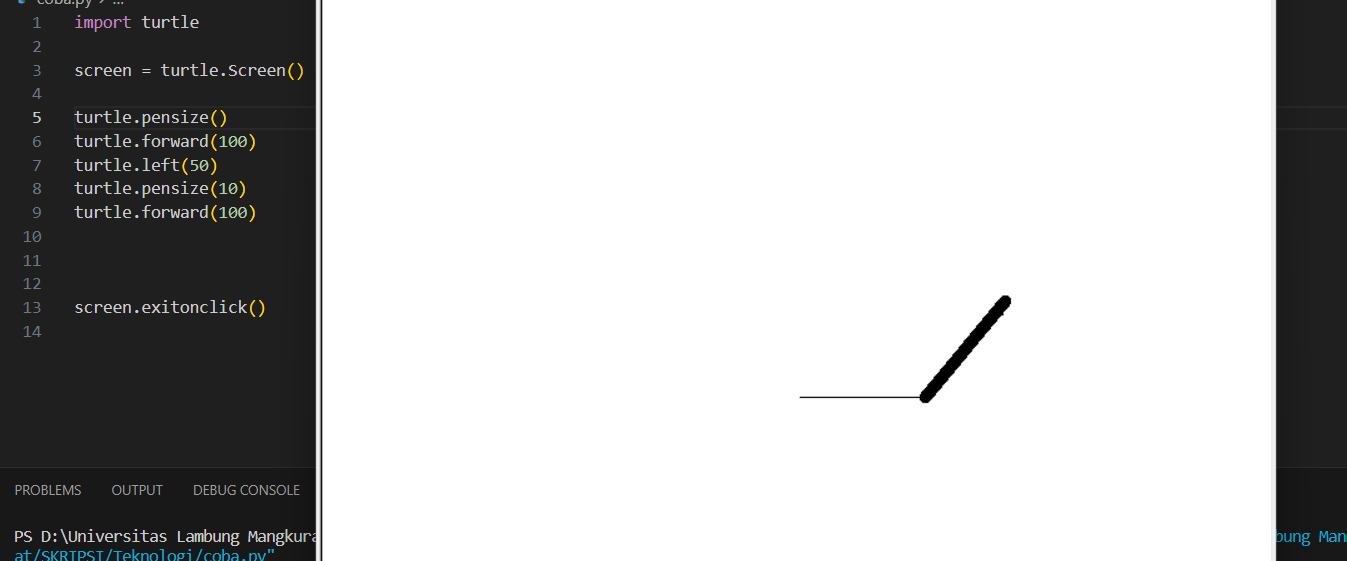
turtle.forward(100)

turtle.left(50)

turtle.pensize(10)

turtle.forward(100)

output :



Ketika argument tidak ditetapkan maka widthnya akan berukuran default.

1. pen()

Fungsi ini digunakan untuk mengembalikan atau mengatur atribut pena dalam 'kamus pena' dengan pasangan kunci/nilai berikut:

* “shown” : True/False
* “pendown” : True/False
* “pencolor” : color-string or color-tuple
* “fillcolor” : color-string or color-tuple
* “pensize” : positive number
* “speed” : number in range 0..10
* “resizemode” : “auto” or “user” or “noresize”
* “stretchfactor”: (positive number, positive number)
* “shearfactor” : number
* “outline” : positive number
* “tilt” : number

Satu atau lebih atribut ini dapat diberikan sebagai argumen kata kunci. Ini dapat digunakan untuk mengatur beberapa atribut pena dalam satu pernyataan.

***Syntax :****turtle.pen(pen=None, \*\*pendict)*

***Parameters:***

* ***pen :****dictionary dengan beberapa atau semua kunci yang tercantum di atas.*
* ***\*\*pendict :****satu atau lebih argumen kata kunci dengan kunci yang tercantum di atas sebagai kata kunci.*

Contoh :

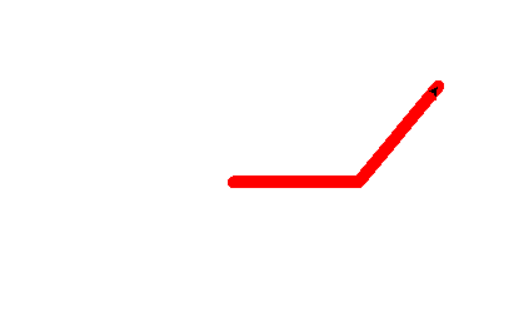
turtle.pen(*fillcolor="black",pencolor="red", pensize=10*)

turtle.forward(100)

turtle.left(50)

turtle.forward(100)

Output :



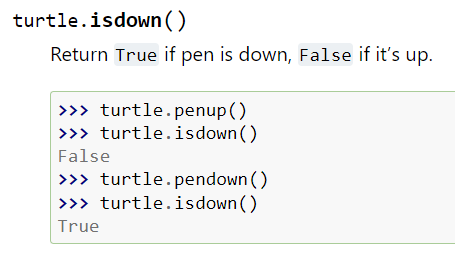
1. isdown()

Cara ini digunakan untuk mengecek apakah penyu sudah turun atau belum. Itu tidak memerlukan argumen apa pun dan mengembalikan nilai boolean sebagai True jika penanya turun, False jika penanya naik.

**Syntax :**

turtle.isdown()

Contoh :



1. Color Control
2. color()

Cara ini digunakan untuk mengubah warna penyu. Warna defaultnya adalah hitam.

**Syntax:**

turtle.color(\*args)

|  |  |  |
| --- | --- | --- |
| Format | Arguments | Deskription |
| turtle.color(colorstring) | colorstring | Serangkaian nama warna seperti “red”, “green”, dll. |
| turtle.color((r, g, b)) | (r, g, b) | Tupel tiga nilai r, g, dan b menggunakan kode warna rgb |
| turtle.color( r, g, b ) | r, g, b | Tiga nilai r, g, dan b menggunakan kode warna rgb |

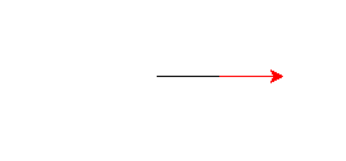
Contoh :

Turtle.forward(50)

Turtle.color(“red”)

Turtle.forward(50)

Output:



1. pencolor()

Cara ini digunakan untuk mengubah warna tinta gambar penyu. Warna defaultnya adalah hitam.

***Syntax:****turtle.pencolor(\*args)*

***Arguments:****Metode ini memiliki argumen berikut:*

* ***colorstring:****(opsional) colorstring Serangkaian nama warna seperti “red”, “green”, dll.*
* ***(r, g, b):****(opsional) Tupel tiga nilai r, g, dan b menggunakan kode warna rgb..*
* ***r, g, b:****(opsional) Tiga nilai r, g, dan b menggunakan kode warna rgb.*

***Return:****None*

Contoh:

turtle.forward(50)

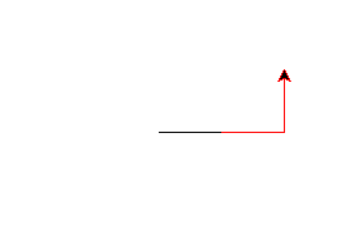
turtle.pencolor("red")

turtle.forward(50)

turtle.left(90)

turtle.forward(50)

Output:



1. fillcolor()

Metode ini digunakan untuk mengembalikan atau mengatur warna isian. Jika bentuk penyu adalah poligon, bagian dalam poligon tersebut digambar dengan warna isian yang baru ditetapkan.

***Syntax :****turtle.fillcolor(\*args)*

***Parameters:***

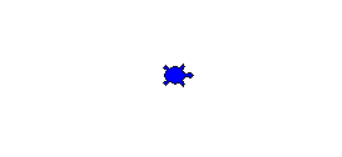
* ***fillcolor() :****Mengembalikan warna pengisi saat ini sebagai string spesifikasi warna, mungkin dalam format angka hex.*
* ***fillcolor(colorstring) :****Ini adalah string spesifikasi warna Tk, seperti “red” atau “yellow”.*
* ***fillcolor((r, g, b)) :****Tupel r, g, dan b, yang mewakili, warna RGB, dan masing-masing r, g, dan b berada dalam rentang 0 hingga mode warna.*
* ***fillcolor(r, g, b) :****r, g, dan b mewakili warna RGB, dan masing-masing r, g, dan b  berada dalam rentang 0 hingga mode warna.*

Contoh :

turtle.shape("turtle")

turtle.fillcolor("blue")

Output :



1. Filling
2. filling()

Fungsi ini digunakan untuk mengembalikan status pengisian (True jika terisi, jika tidak terisi maka False). Itu tidak memerlukan argumen apa pun.

Contoh 1 :

# check the default

# filling state

print(turtle.filling())

Output :

False

Contoh 2 :

# begin the fill and check

# the filling state

turtle.begin\_fill()

print(turtle.filling())

Output :

True

1. begin\_fill()

Metode ini digunakan untuk memanggil sesaat sebelum menggambar bentuk yang akan diisi. Tidak perlu argumen apa pun.

**Syntax :**

turtle.begin\_fill()

Contoh 1:

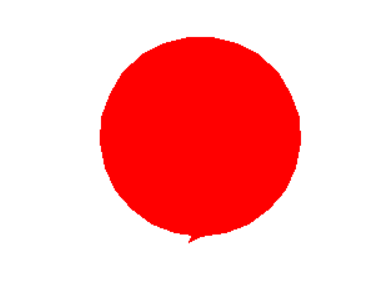
turtle.color("red")

turtle.begin\_fill()

turtle.circle(80)

turtle.end\_fill()

Output:



Ketika kita tidak menggunakan begin\_fill() maka gambar tersebut tidak akan terisi.

Contoh 2:

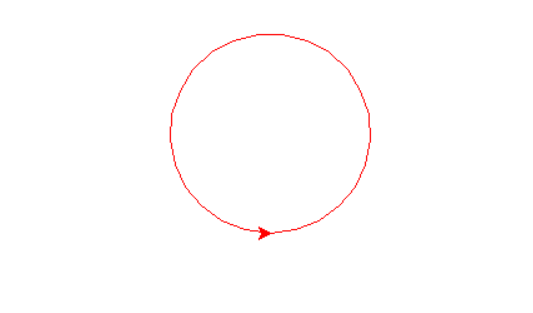
turtle.color("red")

# turtle.begin\_fill()

turtle.circle(80)

turtle.end\_fill()

Output:



1. end\_fill()

Metode ini digunakan untuk mengisi bentuk yang digambar setelah panggilan begin\_fill(). Tidak perlu argumen apa pun.

**Syntax :**

turtle.end\_fill()

Contoh 1:

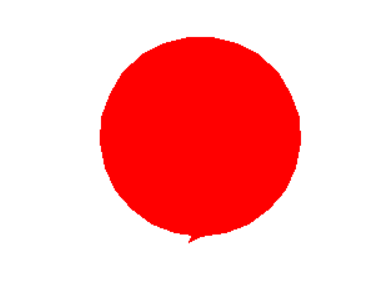
turtle.color("red")

turtle.begin\_fill()

turtle.circle(80)

turtle.end\_fill()

Output :



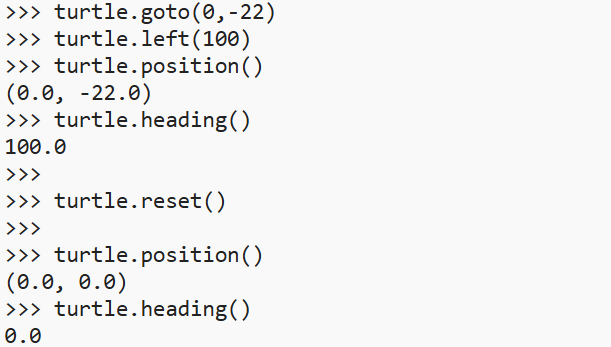
1. More Drawing Control
2. reset()

Fungsi ini digunakan untuk menghapus gambar kura-kura dan mengembalikan nilai defaultnya. Itu tidak memerlukan argumen apa pun.

**Syntax :**

turtle.reset()

Contoh:



Ketika kita gunakan fungsi reset(), maka position dan heading objek akan kembali ke nilai default.

1. clear()

Fungsi ini digunakan untuk menghapus gambar kura-kura dari layar. Jangan berpindah-pindah keadaan dan posisi penyu serta gambar penyu lainnya tidak terpengaruh. Itu tidak memerlukan argumen apa pun.

***Syntax:****turtle.clear()****Parameters :****None*

Contoh :

turtle.color("red")

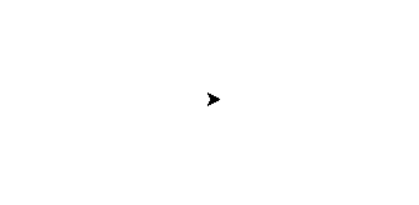
turtle.begin\_fill()

turtle.circle(80)

turtle.end\_fill()

turtle.reset()

Output :



1. write()

Fungsi ini digunakan untuk menulis teks pada posisi penyu saat ini.

**Syntax :**

*turtle.write(arg, move=False, align=’left’, font=(‘Arial’, 8, ‘normal’))*

**Parameters:**

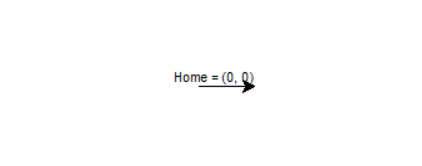
|  |  |
| --- | --- |
| Arguments | Description |
| arg | Info yang akan ditulis ke TurtleScreen |
| move | True/False |
| align | Salah satu strings “left”, “center” or right” |
| font | (fontname, fontsize, fonttype) |

Contoh:

turtle.write("Home= ", True, align="center")

turtle.write((0,0), True)

Output:



1. Turtle State
2. Using events
3. Special Turtle Methods
4. **Methods of TurtleScreen/Screen**
5. Window control
6. Animation control
7. Using Screen Event
8. Settings and special methods
9. Input method
10. Method specific to Screen
11. D