Cours - Thème: Python - Je fais des rosaces avec Turtle

Lien vers une interface de programmation en ligne
https://repl.it/languages/python3

Lien vers un site permettant d'installer une interface de programmation
https://thonny.org/

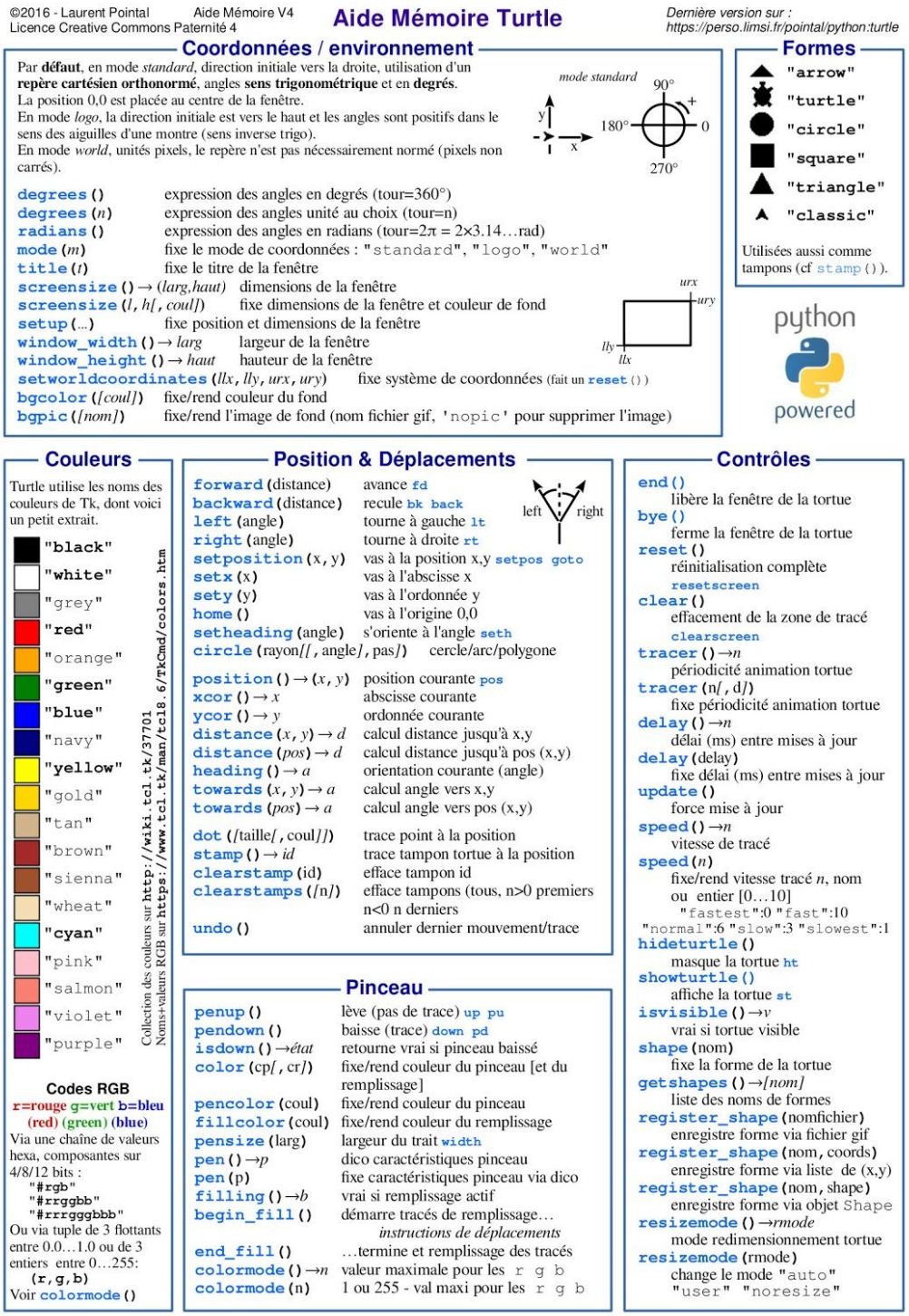
QrCode pour la prise en main de Thonny →

Objectif

Créer des jolis dessins avec python en utilisant la bibliothèque Turtle et comprendre les paramètres et les boucles

Table des matières

1 Rappel des commandes principales



Aide Mémoire Turtle

Dernière version sur :
https://perso.limsi.fr/pointal/python:turtle

Coordonnées / environnement

Par **défaut**, en mode *standard*, direction initiale vers la droite, utilisation d'un **repère cartésien orthonormé**, angles **sens trigonométrique** et en **degrés**.
La position 0,0 est placée au centre de la fenêtre.
En mode *logo*, la direction initiale est vers le haut et les angles sont positifs dans le sens des aiguilles d'une montre (sens inverse trigo).
En mode *world*, unités pixels, le repère n'est pas nécessairement normé (pixels non carrés).

mode standard : 90° / 180° / 0 / 270°, y, x

`degrees()`	expression des angles en degrés (tour=360°)
`degrees(`*n*`)`	expression des angles unité au choix (tour=n)
`radians()`	expression des angles en radians (tour=2π = 2×3.14…rad)
`mode(`*m*`)`	fixe le mode de coordonnées : `"standard"`, `"logo"`, `"world"`
`title(`*t*`)`	fixe le titre de la fenêtre
`screensize()` → (*larg*,*haut*)	dimensions de la fenêtre
`screensize(`*l*, *h*[, *coul*]`)`	fixe dimensions de la fenêtre et couleur de fond
`setup(`…`)`	fixe position et dimensions de la fenêtre
`window_width()` → *larg*	largeur de la fenêtre
`window_height()` → *haut*	hauteur de la fenêtre
`setworldcoordinates(`*llx*, *lly*, *urx*, *ury*`)`	fixe système de coordonnées (fait un `reset()`)
`bgcolor(`[*coul*]`)`	fixe/rend couleur du fond
`bgpic(`[*nom*]`)`	fixe/rend l'image de fond (nom fichier gif, `'nopic'` pour supprimer l'image)

urx, ury, lly, llx

Formes

`"arrow"` `"turtle"` `"circle"` `"square"` `"triangle"` `"classic"`

Utilisées aussi comme tampons (cf `stamp()`).

python powered

Couleurs

Turtle utilise les noms des couleurs de Tk, dont voici un petit extrait.

`"black"` `"white"` `"grey"` `"red"` `"orange"` `"green"` `"blue"` `"navy"` `"yellow"` `"gold"` `"tan"` `"brown"` `"sienna"` `"wheat"` `"cyan"` `"pink"` `"salmon"` `"violet"` `"purple"`

Collection des couleurs sur `http://wiki.tcl.tk/37701`
Noms+valeurs RGB sur `https://www.tcl.tk/man/tcl8.6/TkCmd/colors.htm`

Codes RGB
r=rouge g=vert b=bleu
(red) (green) (blue)
Via une chaîne de valeurs hexa, composantes sur 4/8/12 bits :
`"#rgb"`
`"#rrggbb"`
`"#rrrgggbbb"`
Ou via tuple de 3 flottants entre 0.0…1.0 ou de 3 entiers entre 0…255:
`(r,g,b)`
Voir `colormode()`

Position & Déplacements

left / right

`forward(`distance`)`	avance `fd`
`backward(`distance`)`	recule `bk` `back`
`left(`angle`)`	tourne à gauche `lt`
`right(`angle`)`	tourne à droite `rt`
`setposition(`x, y`)`	vas à la position x,y `setpos` `goto`
`setx(`x`)`	vas à l'abscisse x
`sety(`y`)`	vas à l'ordonnée y
`home()`	vas à l'origine 0,0
`setheading(`angle`)`	s'oriente à l'angle `seth`
`circle(`rayon[*[*, angle*]*, pas*]*`)`	cercle/arc/polygone
`position()` → (*x*, *y*)	position courante `pos`
`xcor()` → *x*	abscisse courante
`ycor()` → *y*	ordonnée courante
`distance(`*x*, *y*`)` → *d*	calcul distance jusqu'à x,y
`distance(`*pos*`)` → *d*	calcul distance jusqu'à pos (x,y)
`heading()` → *a*	orientation courante (angle)
`towards(`*x*, *y*`)` → *a*	calcul angle vers x,y
`towards(`*pos*`)` → *a*	calcul angle vers pos (x,y)
`dot(`[taille[, coul]]`)`	trace point à la position
`stamp()` → *id*	trace tampon tortue à la position
`clearstamp(`id`)`	efface tampon id
`clearstamps(`[n]`)`	efface tampons (tous, n>0 premiers n<0 n derniers
`undo()`	annuler dernier mouvement/trace

Pinceau

`penup()`	lève (pas de trace) `up` `pu`
`pendown()`	baisse (trace) `down` `pd`
`isdown()` →*état*	retourne vrai si pinceau baissé
`color(`cp[, cr]`)`	fixe/rend couleur du pinceau [et du remplissage]
`pencolor(`coul`)`	fixe/rend couleur du pinceau
`fillcolor(`coul`)`	fixe/rend couleur du remplissage
`pensize(`larg`)`	largeur du trait `width`
`pen()` →*p*	dico caractéristiques pinceau
`pen(`p`)`	fixe caractéristiques pinceau via dico
`filling()` →*b*	vrai si remplissage actif
`begin_fill()`	démarre tracés de remplissage… *instructions de déplacements*
`end_fill()`	…termine et remplissage des tracés
`colormode()` →*n*	valeur maximale pour les `r g b`
`colormode(n)`	1 ou 255 - val maxi pour les `r g b`

Contrôles

`end()`
libère la fenêtre de la tortue
`bye()`
ferme la fenêtre de la tortue
`reset()`
réinitialisation complète `resetscreen`
`clear()`
effacement de la zone de tracé `clearscreen`
`tracer()` →*n*
périodicité animation tortue
`tracer(`n[, d]`)`
fixe périodicité animation tortue
`delay()` →*n*
délai (ms) entre mises à jour
`delay(`delay`)`
fixe délai (ms) entre mises à jour
`update()`
force mise à jour
`speed()` →*n*
vitesse de tracé
`speed(`*n*`)`
fixe/rend vitesse tracé *n*, nom ou entier [0…10]
`"fastest":0` `"fast":10` `"normal":6` `"slow":3` `"slowest":1`
`hideturtle()`
masque la tortue `ht`
`showturtle()`
affiche la tortue `st`
`isvisible()` →*v*
vrai si tortue visible
`shape(`nom`)`
fixe la forme de la tortue
`getshapes()` →*[nom]*
liste des noms de formes
`register_shape(`nomfichier`)`
enregistre forme via fichier gif
`register_shape(`nom, coords`)`
enregistre forme via liste de (x,y)
`register_shape(`nom, shape`)`
enregistre forme via objet `Shape`
`resizemode()` →*rmode*
mode redimensionnement tortue
`resizemode(`rmode`)`
change le mode `"auto"` `"user"` `"noresize"`

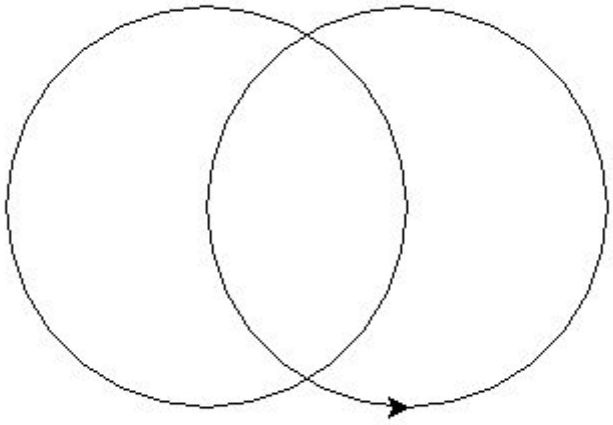
Avec ceci vous pouvez tout dessiner!
Source https://perso.limsi.fr/pointal/_media/python:turtle:turtleref.pdf

2 Premiers pas vers la rosace

Votre objectif est d'écrire des petits programmes répondant aux objectifs suivants. Puisque nous utilisons Turtle, ne pas oublier d'appeler cette bibliothèque, donc vos programmes commenceront toujours par:
from turtle import *

- **Objectif 1: Tracer des cercles**

Tracer deux cercles de rayon 100 décalés dont les centres sont décalés de 100.
J'ai eu besoin de circle, pendown, penup



Votre code:

Vos remarques:

- **Objectif 2: Tracer le début d'une rosace comme quand on était jeune**

Tester le code suivant en mode débug et commenter la colonne de droite du tableau suivant:

Script	Vos commentaires Commenter chaque ligne commençant par #

from turtle import *
 	
# dessin des axes 	
home()
penup()
goto(-200,0)
pendown()
fd(400)
penup()
goto(0,-200)
pendown()
left(90)
fd(400)
penup()
home()
 	
# cercle centre 0,0 	
pencolor("red")
goto(0,-50)
pendown()
circle(50)
penup()
home()
 	
# 1er cercle bleu 	
goto(0,-50)
circle(50,30)
dot()
penup()
right(90)
forward(50)
right(-90)
pendown()
pencolor("blue")
circle(50)
penup()
home()
 	
# 2eme cercle bleu 	
goto(0,-50)
circle(50,30+60)
dot()
	# from… =...
	
	# un commentaire commence par #
	# home()=...
	# penup()=...
	# goto(x,y)=...
	# pendown()=...
	# fd()= raccourci de forward()=...
	
	# home()= retour à l'origine et orientation de...
	
	# pencolor(“ “)=...
	
	# circle(50,30)=...
	# dot()=...
	
	# pourquoi un right(90) ici?=...
	
	# pourquoi le 60 dans circle(50,30+60)?=...

penup() right(90) forward(50) right(-90) pendown() circle(50) penup() home() # 3eme cercle bleu # 4eme cercle bleu # 5eme cercle bleu # 6eme cercle bleu	# Les lignes suivantes ne servent à rien pour l'instant!!!

Vos remarques si besoin:

3 La rosace finale

- **Objectif 3: Tracer le début d'une rosace comme quand on était jeune**

Vous pouvez aborder la suite en 2 temps ou faire une proposition de script final directement. L'objectif est de dessiner une rosace de 6 cercles bleus autour du cercle central rouge.

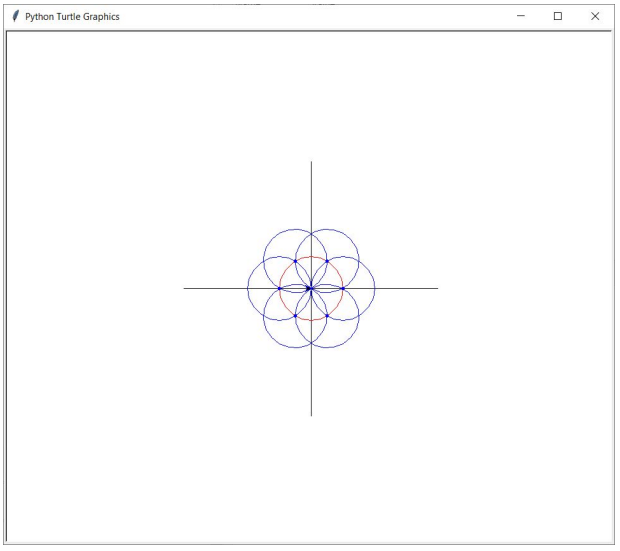
- en deux étapes:
  - faire une proposition pour les 3eme, 4eme, 5eme et 6eme cercle en vous inspirant fortement (copier/coller avec une légère modif du code traçant le # 2ème cercle bleu)

```
50 penup()
51 home()
52
53 # 3eme cercle bleu centree
54
55 # 4eme cercle bleu centree
56
57 # 5eme cercle bleu centree
58
59 # 6eme cercle bleu centree
60
Shell
>>> %Run aaa.py
```

  - Comprendre que tous les cercles bleus peuvent intégrer une boucle for i in range(6)
  - Comprendre la subtile ligne 42 et la modifier en conséquence (elle devrait selon moi contenir un i entre autre ;)

```
41 goto(0,-50)
42 circle(50,30+60)
43 dot()
```

  - Tester votre script et le simplifier car vous tracez probablement 2 fois un même cercle ;)
- en une étape:
  - proposer un script avec une boucle donnant ceci:



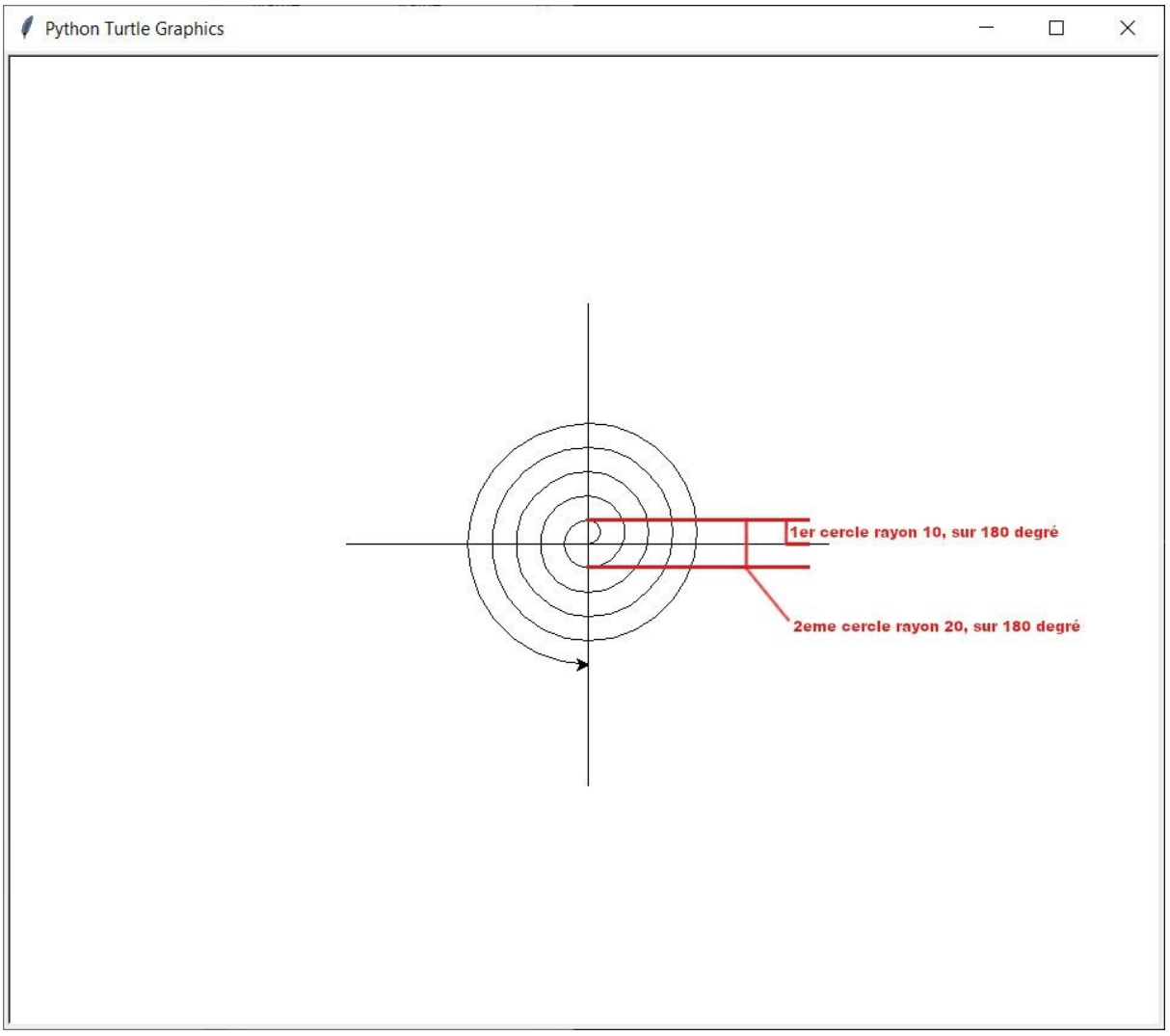
Votre code final:

4 Le challenge spirale

- **Challenge final: The spirale...**

Faire ce joli escargot!
Pour vous aider:

- j'ai repris les lignes de code traçant le dessin des axes
- je dessine des moitiés de cercle donc je tourne de 180 degrés
- mon premier demi-cercle fait 10 de rayon
- j'augmente ce rayon de 10 pour chaque nouveau ½ cercle
- et pour info j'ai bouclé 10 fois



Et pour vous motivez, cadeau d'un script

Alors vous avez fait comment? Votre code final: