Python Turtle

So far the only **output** we have looked at is **text**. Our python programs can give us an answer to a question or respond to a choice by displaying text.
Another important **output** of computer programming is **images**. In fact, displaying images is one of the most important functions of a computer….think about:

- The display on your phone,
- video games,
- touch screens,
- Websites

All the above outputs rely on displaying images on a screen.

There are many different types of software and programming languages that are used to help programmers display images on a screen. Python **turtle** is not the most efficient way to create images, but it will teach you some more fundamentals of python and give you some of the basics tools that are common to **all** image creation in computer programing.

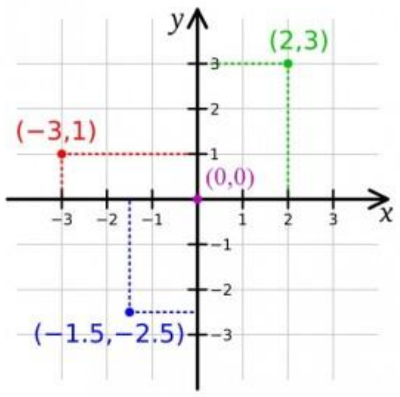
What is turtle?

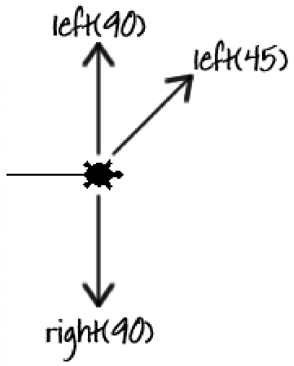
Turtle is a Python library that allows you to draw simple images on the screen using simple python commands. The intention of turtle is NOT to give you a powerful/easy way to display images, but rather give you a cool way to learn python commands where you can see visually appealing output (rather than just text).

In this section we will use turtle to learn commands and techniques that will help **ALL** your programming in the future regardless of whether or not you are creating images

How does turtle work?

The turtle library in python that allows you to **draw images in a similar way to a pen**. Essentially you control a "turtle" that can trace lines on a **coordinate system** as he walks. The turtle can turn, move forward, change colors and many other commands.





Exercise #1

- Read the following code carefully.
- Enter the following code into trinket (don't cut and paste. **Type it in**.)
- Run the code to see what it does!

#1

```
from turtle import Turtle
tom=Turtle()

tom.shape("turtle")
tom.speed(2)

tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
```

#2

Now add the following red lines to the previous code

```
from turtle import Turtle
tom=Turtle()

tom.shape("turtle")
tom.speed(2)

tom.begin_fill()
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.end_fill()
tom.penup()
tom.goto(100,100)
```

#3

Now add the following green lines to the previous code:

```
from turtle import Turtle
tom=Turtle()

tom.shape("turtle")
tom.speed(2)

tom.begin_fill()
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.end_fill()
tom.penup()
tom.goto(100,100)

tom.hideturtle()

tom.begin_fill()
tom.pendown()
tom.color('green')
tom.circle(40)
tom.end_fill()
```

#4 Now add the following blue lines to the previous code

```
from turtle import Turtle
tom=Turtle()

tom.shape("turtle")
tom.speed(2)

tom.begin_fill()
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.end_fill()
tom.penup()
tom.goto(100,100)

tom.hideturtle()

tom.begin_fill()
tom.pendown()
tom.color('green')
tom.circle(40)
tom.end_fill()

tom.pensize(4)
tom.penup()
tom.goto(-100,100)
tom.pendown()
tom.color('purple')
tom.circle(40)
tom.end_fill()
```

#5 Now add the following purple lines to the previous code

```
from turtle import Turtle
tom=Turtle()

tom.shape("turtle")
tom.speed(7)

tom.begin_fill()
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.left(90)
tom.forward(50)
tom.end_fill()
tom.penup()
tom.goto(100,100)

tom.hideturtle()
tom.begin_fill()
tom.pendown()
tom.color('green')
tom.circle(40)
tom.end_fill()

tom.penup()
tom.goto(-100,100)
tom.pendown()
tom.begin_fill()
tom.color('purple')
tom.circle(40)
tom.end_fill()

tom.penup()
tom.goto(-100,-120)
tom.pensize(10)
tom.color('blue')
tom.pendown()
tom.begin_fill()
tom.left(60)
tom.forward(95)
tom.left(120)
tom.forward(95)
tom.left(120)
tom.forward(95)
tom.end_fill()
```

For loops - Dealing with repeated commands

Often in programming we will ask the computer to **repeat** a series of commands.
For example as we saw in our previous code:

```
tom.forward(50)
tom.left(90)
```

We can repeating the above command 4 times will draw a square.

How do we do that in python?
We can use something called a **for loop**.

```
for x in range (0,4):
      tom.forward(50)
      tom.left(90)
```

This will repeat the commands in green 4 times.

Exercise #2

Type the following into trinket (exactly) watch carefully for **indenting**. Correct indenting is a super important part of python code. The code won't run if you haven't indented correctly.

```
from turtle import Turtle
tom=Turtle()

tom.shape("turtle")
tom.speed(2)
for x in range (0,4):
  tom.forward(50)
  tom.left(90)
tom.penup()
tom.goto(0,-100)
tom.pendown()
for x in range(0,10):
  tom.forward(20)
  tom.left(36)
tom.penup()
tom.goto(-120,-70)
tom.pendown()
for x in range(0,3):
  tom.forward(60)
  tom.left(120)
```

Exercise #2…continued

Writing with Turtle

Type in the following code **at the end of your previous code**, run the code, save it, **make sure you know how it works**:

```
import turtle as tom
tom.penup()
tom.goto(-20,100)
tom.color('blue')
tom.write("Mr. Walzl is the Best!", None, align="center",
font=('Verdana', 12, 'bold'))
tom.hideturtle()
```

User interaction with Turtle

Type in the following code at the **end of your previous code**, then:

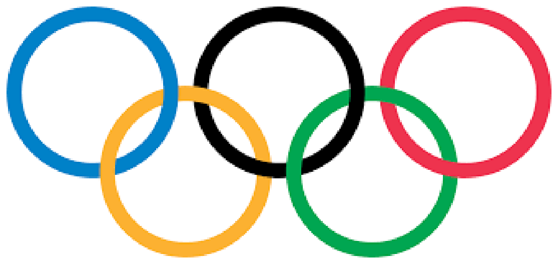
a) run the code,
b) Use your mouse and click on the turtle to see what happens
c) Save your code.
d) **make sure you know how it works**:

```
tom.showturtle()
def move_ahead(x,y):
  tom.pendown()
  tom.forward(100)
tom.onclick(move_ahead)
```

What's happening in the code above? We are creating a **function** called `move_ahead()`. When we **click on** the **turtle** he should move ahead! **Try it out**! Save your work as part of Exercise#2.

There are dozens of cool features and commands like this in **python's turtle library** you can use. Click on the course page link that says: "**python turtle commands**" to see what turtle has to offer!

Exercise#3



See if you can use python turtle to draw the Olympics rings accurately. You can try on your own or there is a 5 minute video tutorial packed with goodies **on the course page** you can follow along with.

Exercise#4

Use the **Turtle Race** link tutorial **on the course page** to create a new and improved turtle race game!

- Run the turtle race code presented at the link.
- Cut and paste the code into your own trinket.
- Create a new and improved version of the race.

Make sure your **new version** of the race that has the at least one following features:

1. Allow the user to select the turtle they think will win.
2. Tell the user at the end of the race if they were correct or not.
3. See if you can make the motion of the turtles "smoother:.

You may modify the game in any additional way you wish in a way that you think **improves** the game.

Bonus : Allows *multiple users* to select the turtle they think will win. Each user should also select a unique user name. Points will be given out after each race depending on whose turtle comes in first, second, third etc. The game should have multiple rounds and player with the most points after the last round should be identified as the winner.

Exercise #5

```
import turtle as t
t.up()
t.pencolor("green")
t.fillcolor("green")
t.penup()
t.goto(0,0)
t.pendown()
t.begin_fill()
t.forward(67)
t.left(90)
t.forward(10)
t.left(90)
t.forward(67)
t.left(90)
t.forward(10)
t.end_fill()

t.penup()
t.goto(10, 5)
t.pendown()
t.setheading(45)
t.forward(20)
t.setheading(0)
t.forward(25)
t.setheading(-45)
t.forward(20)
t.setheading(90)
t.penup()
t.end_fill()

t.penup()
t.goto(20,0)
t.pendown()
t.color("black")
t.fillcolor("black")
t.begin_fill()
t.circle(5)
t.end_fill()
t.penup()
t.goto(58,0)
t.pendown()
t.color("black")
t.fillcolor("black")
t.begin_fill()
t.circle(5)
t.end_fill()
t.hideturtle()
t.up()
t.setheading(0)
```

1.**Cut and Paste** this code into trinket.
2. Investigate the code so you know what it's doing.
3. Modify the car if you like.
3. Add 2 more images to the page that have as much or more detail then the car has.

Exercise #6

1. **Add** the code below to the end of the code for exercise#5.
2. Make sure you click on the turtle window after the car finishes drawing.
3. Use the left and right arrow keys on your keyboard to move the turtle!...cool right?
4. See of you can add to the code so the turtle goes up and down!

```
t.goto(-50,-50)
t.showturtle()

def left():
  t.setheading(180)
  t.forward(10)

def right():
  t.setheading(0)
  t.forward(10)

screen=t.Screen()
screen.onkey(left,"Left")
screen.onkey(right,"Right")
screen.listen()
screen.mainloop()
```

Exercise #7 Mini Video Game!

1. Click on the link that says **Bird and Turtle Game** on the course page. This will bring you to some code for a mini video game.
2. Read the instructions and try out the game.
3. Look at the code and see if you can figure out how it works. **This might be tough**, there is a lot going on in the code and most of it we haven't covered yet. **But try**, it will be a good way for you to practice looking at Python code and you might learn a thing or two.
4. Your task. After you have played the game and examined the code. Try to improve the game in **at least** one of the following ways:

    (a) Add some new still or animated features to the game. Examples: add some trees, a road for the turtle, A title for the game.
    (b) See if you can make the bird move up and down
    (c) See if you can make the turtle have a non-linear path
    (d) Make the motion of the bird "smoother"
    (e) See if you can import a bird icon for the bird and move the icon around instead of just a triangle.
    (f) Have the game repeat and have a scoring system
    (g) Make the turtle speed up each repeat of the game
    (h) Any other cool feature you can think of adding!

Exercise#8

1. Click on the link that says **Plane and Clouds Game** on the course page. This will bring you to another cool video game.
2. Use the left and right arrow keys to move the plane and the up arrow key to shoot!
3. Look at the code of the game (and code below) and see if you can figure out how the turtle's shape was turned into an actual images!

```
screen = turtle.Screen()
image = "planef.gif"
image1 = "clouds.gif"
screen.addshape(image)
screen.addshape(image1)

t.shape(image)
t.showturtle()

d.shape(image1)
d.showturtle()
```

4. Notice how the images have been **uploaded** to the **File section of the Replit** on the left-hand side. Notice how the files are **.gif** files.
5. Now it's your turn. Copy and paste the game code into your own Replit and see if you can **modify the game in the following ways**:

   a) Upload small **.gif** files to your own Replit that you can use to **replace** the plane and cloud images. Get creative! What is your game going to look like? Note: I have only had success with **.gif** files you can try other types...but I don't think Python Turtle in Replit supports anything other than **.gif** files. Also, you can **resize**, **modify**, and change any image you find online **to a .gif** using a free image editor like: https://image.online-convert.com/convert-to-gif
   I will see if I can put some other good links on the course page.
   b) See if you can insert a cool background for your game.
   c) See if you can introduce other motions or functions with other key strokes.
   d) See if you can have something happen if there is a **collision** between the images in your game. Hint: Look up the distance command ex:
   `if t.distance(c)<30:`