## Public Technology Training

## Code Week: Draw with Python

## Turtle Drawing

Join us in celebrating the [International Hour of Code](https://code.org/) by learning how to draw with code! We'll learn some basic Python coding skills to make art with Raspberry Pi computers.

Experience with using a mouse and typing is required. Kids under the age of 12 must be accompanied by an adult.

Registration is required and will be available online, November 16.

Tuesday, December 6, 2 p.m. – 4 p.m., Computer Training Room

**Learning Objectives**:

* Introduce programming concepts around loops, syntax, problem solving
* Build confidence with ability to gain new skills
* Connect learners with others in the community with similar learning interests
* Connect learners with further learning resources

**Need**:

* Handout1 : [DWV-#1172445-PATRON TECHNOLOGY TRAINING DRAW WITH CODE HANDOUT](PCDOCS://DWV/1172445/1)
* Graph paper
* Pencils
* <https://trinket.io/> < Python emulator <https://trinket.io/python>

|  |  |  |
| --- | --- | --- |
| **Time** | **Topic/Content** | **Resources** |
| 2:00 – 2:05 | Intros and outline   * Who I am:   + Digital Access Librarian   + Programming around computational literacy and Digital Content creation * What the class is for:   + Learning about coding and why it is considered an important skill   + Trying out a type of hardware that is cheap and popular for those learning to code   + Learning about some basic concepts around coding that will give you an idea of the shape and scope of the field   + Gain a bit of a vocabulary around coding * You won’t be an expert of anything after this class, and we are aiming at beginners. |  |
| 2:05 – 2:15 | **Why code and what is hour of code**  There are more jobs than there are skilled workers in the computational science / tech industry, and there is a push for young people to gain those skills.  But we aren’t kids, so why do we care?  Because in order for there to be young people interested in code, etc. There need to be enough people out there who understand what problems code can solve.  When we talk about activities around coding, around Pis, and around other activities, the skills that we are working on, and the traits we are trying to develop are often referred to as computational thinking.  Let’s unpack this a bit more.  The greatest contribution programs that are about “learning code” bring, aren’t people who get jobs as programs, but it changing the way people think – this is computational thinking part of it.  Computational thinking is a bit like: looking at a number of ingredients in your cupboard and all you can see is- onions, lentils, some spices and thinking “soup!” Instead of “takeout”.  Mentally sketching a path from raw ingredients to a meal is computational thinking. Coding is like that – taking some specific building blocks and seeing a way to a solution. Like being able to use downtime effectively (making a salad while the soup is cooking)  Other skills in the computational toolkit are– iteration: testing to see when things are working, and adjusting and improving all the time. < adding a bit more spice, turning down the heat. When you next try to make soup, adding in another ingredient.  These pieces of computational thinking might be using new words, but shouldn’t sound too different.  And really, part of this little session today and programs like it, aren’t about making more programs, especially as adults, but being able to look at a problem and see that there might be a technological solution to it.   * Example Fire Hydrants from article.   *A few years ago some software engineers were in Boston before a terrible snowstorm for a special program. Their plans changed a bit when the storm hit.*  *They discovered that a problem during big storms is that fire hydrants are buried, and firefighters in an emergency have to spend time digging them out.*  *Boston had a list of the locations of all 13,000 hydrants. So they got out their laptops.*  *Now, Boston has adoptahydrant.org, a simple website that lets residents "adopt" hydrants across the city. The site displays a map of little hydrant icons. Green ones have been claimed by someone willing to dig them out after a storm, red ones are still available—500 hydrants were adopted last winter.*  *This has an ongoing cost of $9 a year.*  This type of thinking – using technology to solve everyday problems can help in other ways as well. People are trying to increase diversity in tech fields – if we are thinking about problems that can be solved with programming that matter to us – as library folk and people in a female-dominated profession, will attract young women to enter the field. Programming skills are more than just games, making money, and “math” but also being able to make a difference in society.  Hour of code is:  “The Hour of Code is a global movement by Computer Science Education Week and Code.org reaching tens of millions of students in 180+ countries through a one-hour introduction to computer science and computer programming.”  “Launched in 2013, Code.org® is a non-profit dedicated to expanding access to computer science, and increasing participation by women and underrepresented students of color. Our vision is that every student in every school should have the opportunity to learn computer science. We believe computer science should be part of core curriculum, alongside other courses such as biology, chemistry or algebra.” |  |
| 2:15 – 2:20 | **What is Python? Why are we using it**  “**Python** is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.”  “use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.  Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.” |  |
| 2:20 – 2:25 | **Turtle Basics**  We’re using Turtle today in order to work on “computational thinking” in a fun, visual way. The activities and visual output makes learning fun.  This module comes from “logo” a programming language invented in the 1960.  We’ll be using commands like:   * Forward * Backwards * Right * Left * Color * Turtle * Shape   The format will primarily be:   * Turtle.forward(100) * Or = object.command(steps/color/etc)   As we make shapes and art. | Handout 1 |
| 2:25 – 2:35 | **Activity Prep**   * Clear existing example code * Ctrl + a * Delete * Type in:   import turtle  wn = turtle.Screen()  tommy = turtle.Turtle()  tommy.shape("turtle")  What this does is create an area for the turtle to draw, and our first turtle – named “tommy”  Note, you can set the shape of your turtle (circle, square, turtle, classic, arrow) | Visit <https://trinket.io/python> |
| 2:35 – 2:40 | **Activity 1: square**  How would we draw a square? What angles are the corners?  tommy.forward(50)  tommy.right(90)  tommy.forward(50)  tommy.right(90)  tommy.forward(50)  tommy.right(90)  tommy.forward(50)  Click arrow to run / test the code! | White board |
| 2:40 – 2:50 | **Activity 2: 2nd turtle and triangle**  We need not limit ourselves to only one turtle! Let’s define a second turtle, called “s” or “mabel” and give it a different shape. Also, let’s have it draw a triangle.  s = turtle.Turtle()  s.shape("arrow")  s.right(180)  s.forward(80)  s.right(120)  s.forward(80)  s.right(120)  s.forward(80)  s.right(120)  s.forward(39) | whiteboard |
| 2:50 – 2:55 | **Debrief and introduce next concept**  So we can have as many turtles making art as we would like!  Can you see how it might get tiring to write out each side of a shape? There is something we can do called a “loop” that allows us to have the turtle repeat its moves. We can also change the colour of the drawing. |  |
| 2:55 – 3:05 | **Activity 3: Colour loops**  Let’s edit tommy – add a line to change his colour, and change the square.  tommy = turtle.Turtle()  tommy.shape("turtle")  tommy.color("purple")  for i in [0,1,2,3]:  tommy.left(90)  tommy.forward(50) |  |
| 3:05 – 3:10 | **Debrief and introduce next concept**  The for i in  Lets us tell Tommy to repeat the steps below four times.  One thing to note about Python (and other similar coding languages) is that when counting you start at 0, not 1. |  |
| 3:10 – 3:25 | **Activity 4: circle & star, fill & pen up**  Let’s make a new turtle and draw something easy like a circle!  d = turtle.Turtle()  d.shape("turtle")  d.color("purple")  d.circle(50)  That was easy. Let’s fill it in now with a new command.  d.begin\_fill()  d.circle(50)  d.end\_fill()  Note that our new turtle draw on top of our old turtle. So for our next turtle, we’re going to have it move ini  e = turtle.Turtle()  e.shape("turtle")  e.color("purple")  e.up()  e.forward(100)  e.down  for i in range(5):  e.forward(50)  e.right(144) |  |
| 3:25 – 3:30 | **Debrief and introduce next concept**  So we learned 2 new commands:   * fill (begin and end) * up and down * circle!   So for the next (20) minutes, we’re going to  To help, here is some graph paper to help work out shapes and distance. |  |
| 3:30 – 3:50 | **Challenge time**   * Olympic Rings * Name * Smiley face * Many sided polygon | Handout: Challenges  Graph paper |
| 3:50 – 3:55 | Follow-up learning resources | Handout 2 |
| 3:55 – 4:00 | Upcoming Library events |  |

**Resources**

* <https://docs.python.org/3/library/turtle.html>
* <http://interactivepython.org/runestone/static/thinkcspy/PythonTurtle/OurFirstTurtleProgram.html>
* <http://openbookproject.net/thinkcs/python/english3e/hello_little_turtles.html>
* <http://knuth.luther.edu/~leekent/SamplingCS/lesson5.html>
* <http://www.blog.pythonlibrary.org/2012/08/06/python-using-turtles-for-drawing/>
* <https://michael0x2a.com/blog/turtle-examples>
* my sampler: <https://trinket.io/python/b749bba39a>

**Challenge:** **Spirals**

import turtle

import math

import random

wn = turtle.Screen()

wn.bgcolor('black')

Albert = turtle.Turtle()

Albert.speed(0)

Albert.color('white')

rotate=int(360)

def drawCircles(t,size):

for i in range(10):

t.circle(size)

size=size-4

def drawSpecial(t,size,repeat):

for i in range (repeat):

drawCircles(t,size)

t.right(360/repeat)

drawSpecial(Albert,100,10)

Steve = turtle.Turtle()

Steve.speed(0)

Steve.color('yellow')

rotate=int(90)

def drawCircles(t,size):

for i in range(4):

t.circle(size)

size=size-10

def drawSpecial(t,size,repeat):

for i in range (repeat):

drawCircles(t,size)

t.right(360/repeat)

drawSpecial(Steve,100,10)

Barry = turtle.Turtle()

Barry.speed(0)

Barry.color('blue')

rotate=int(80)

def drawCircles(t,size):

for i in range(4):

t.circle(size)

size=size-5

def drawSpecial(t,size,repeat):

for i in range (repeat):

drawCircles(t,size)

t.right(360/repeat)

drawSpecial(Barry,100,10)

Terry = turtle.Turtle()

Terry.speed(0)

Terry.color('orange')

rotate=int(90)

def drawCircles(t,size):

for i in range(4):

t.circle(size)

size=size-19

def drawSpecial(t,size,repeat):

for i in range (repeat):

drawCircles(t,size)

t.right(360/repeat)

drawSpecial(Terry,100,10)

Will = turtle.Turtle()

Will.speed(0)

Will.color('pink')

rotate=int(90)

def drawCircles(t,size):

for i in range(4):

t.circle(size)

size=size-20

def drawSpecial(t,size,repeat):

for i in range (repeat):

drawCircles(t,size)

t.right(360/repeat)

drawSpecial(Will,100,10)

**Challenge: Smiley Face**

import turtle

smiles = turtle.Turtle()

smiles.penup()

smiles.goto(-75,150)

smiles.pendown()

smiles.begin\_fill()

smiles.circle(10) #eye one

smiles.end\_fill()

smiles.penup()

smiles.goto(75,150)

smiles.pendown()

smiles.begin\_fill()

smiles.circle(10) #eye two

smiles.end\_fill()

smiles.penup()

smiles.goto(0,0)

smiles.pendown()

smiles.circle(100,90) #right smile

smiles.penup()

smiles.setheading(180) # <-- look West

smiles.goto(0,0)

smiles.pendown()

smiles.circle(-100,90)

smiles.penup()

smiles.goto(150,100)

smiles.pendown()

smiles.circle(150)

**Challenge: Olympic Rings**

import turtle

wn=turtle.Screen()

black = turtle.Turtle()

black.speed(25)

black.pensize(6)

black.circle(50)

red = turtle.Turtle()

red.speed(25)

red.pensize(6)

red.color("red")

red.penup()

red.forward(128)

red.pendown()

red.circle(50)

b = turtle.Turtle()

b.speed(25)

b.pensize(6)

b.color("blue")

b.penup()

b.backward(128)

b.pendown()

b.circle(50)

g = turtle.Turtle()

g.speed(25)

g.pensize(6)

g.color("green")

g.penup()

g.forward(14)

g.right(90)

g.pendown()

g.circle(50)

y = turtle.Turtle()

y.speed(25)

y.pensize(6)

y.color("yellow")

y.penup()

y.backward(114)

y.right(90)

y.pendown()

y.circle(50)

**Challenge: Write your name** \*\*need graph paper\*\*

import turtle

wn=turtle.Screen()

s = turtle.Turtle()

s.color("green")

s.pensize(11)

s.speed = (5)

s.up()

s.goto(-175,0)

s.down()

s.forward(50)

s.left(90)

s.forward(50)

s.left(90)

s.forward(50)

s.right(90)

s.forward(50)

s.right(90)

s.forward(50)

a1 = turtle.Turtle()

a1.color("blue")

a1.pensize(11)

a1.speed = (5)

a1.up()

a1.goto(-50,0)

a1.down()

a1.backward(50)

a1.left(90)

a1.forward(50)

a1.right(90)

a1.forward(50)

a1.right(90)

a1.forward(65)

r = turtle.Turtle()

r.color("violet")

r.pensize(11)

r.speed = (5)

r.up()

r.goto(-25,0)

r.down()

r.left(90)

r.forward(50)

r.right(45)

r.forward(25)

r.right(90)

r.forward(25)

a2 = turtle.Turtle()

a2.color("red")

a2.pensize(11)

a2.speed = (5)

a2.up()

a2.goto(80,0)

a2.down()

a2.backward(50)

a2.left(90)

a2.forward(50)

a2.right(90)

a2.forward(50)

a2.right(90)

a2.forward(65)

h = turtle.Turtle()

h.color("light blue")

h.pensize(11)

h.speed = (5)

h.up()

h.goto(105,0)

h.down()

h.left(90)

h.forward(100)

h.backward(50)

h.right(90)

h.forward(50)

h.right(90)

h.forward(55)