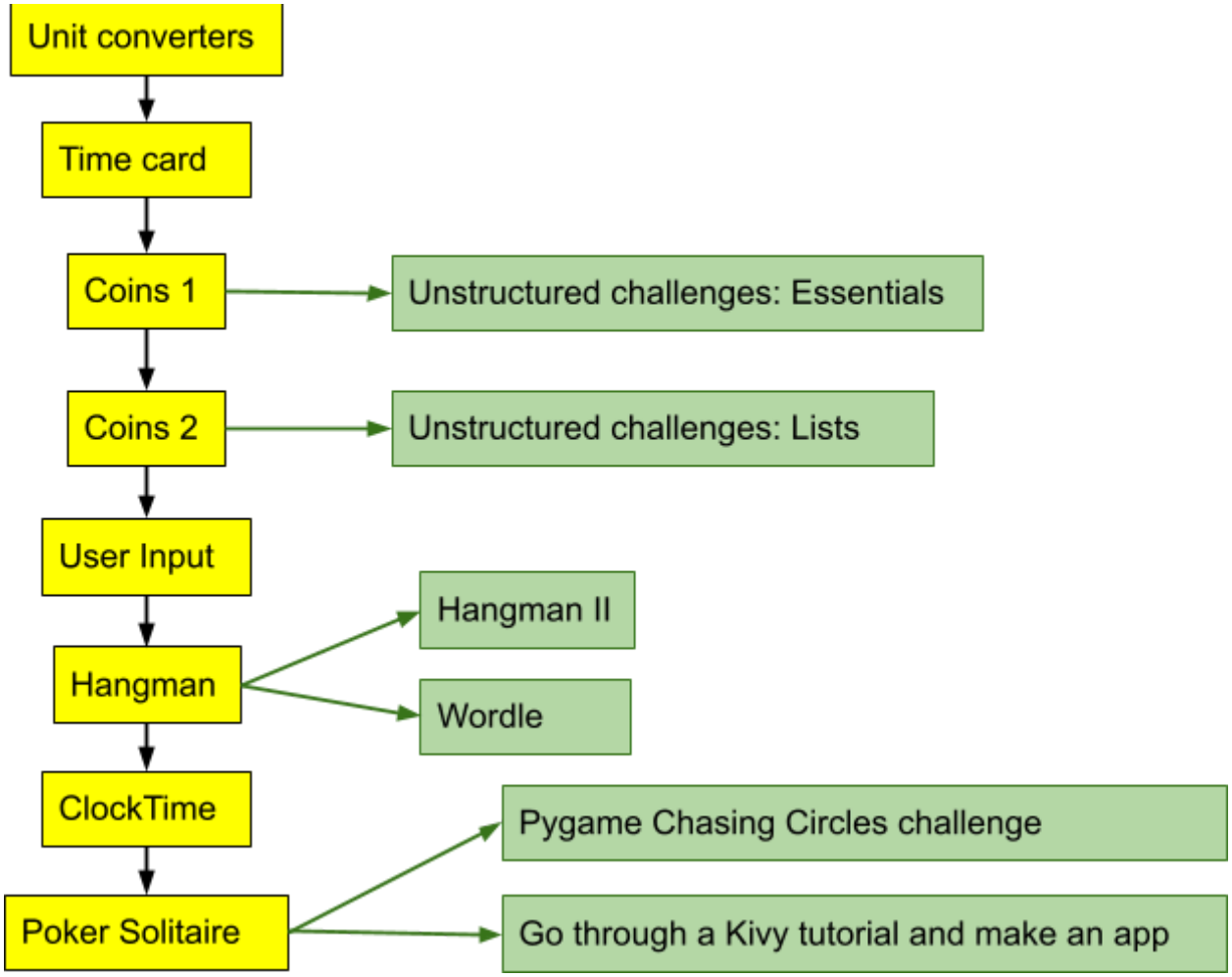
Recommended progression

Recommended progression in yellow, with side-quests in green for more practice.



Unit Converters

Create several simple unit converting apps.

Learning objectives:

- defining and using variables
- doing computations
- using conditional statements (IF statements)
- text input and output

Time Card

Create a fully-functional time-card app.

Prerequisite: Unit Converters

Learning objective: get comfortable creating algorithms with multiple steps and conditions.

Coins 1: WHILE loop

Simulate a piggy bank.

Prerequisite: Unit Converters. (Time Card strongly recommended.)

Learning objective: learn how to use the WHILE loop

Coins 2: lists and FOR loop

Work with a bunch of coins.

Prerequisite: Coins 1

Learning objective: learn how to use lists and the FOR loop

User input - Strings

Foolproof user input.

Prerequisite: Coins 2

Learning objectives: learn how to work with strings, and how to create your own functions.

Hangman

Create a "hangman" game app.

Prerequisite: Coins 2

Learning objectives:

- Writing and using your own functions
- Mastering the essentials of strings
- Mastering loops
- Working with lists, and in particular getting a random element from a list
- Working with a text file

Hangman II

Write a hangman app that can play the game multiple times.

Prerequisite: Hangman

Learning objectives: same as Hangman, but with less scaffolding

Wordle

Create a "wordle" game app.

Warning: the colored text works fine in Jupyter Notebook or Google Collab, but does not work in Terminal or Command.

Prerequisite: Hangman

Learning objectives: same as Hangman, but with less scaffolding

ClockTime - Intro to Classes

Create a ClockTime class of objects.

Prerequisite: Time Card, Hangman

Learning objectives: This is an introduction to creating one's own classes of objects.

Poker Solitaire - Intro OOP

Create a text-based Poker Solitaire game with the Object-Oriented Programming (OOP) approach.

Requirement: ClockTime - Intro to Classes

Learning objective: Object-Oriented Programming

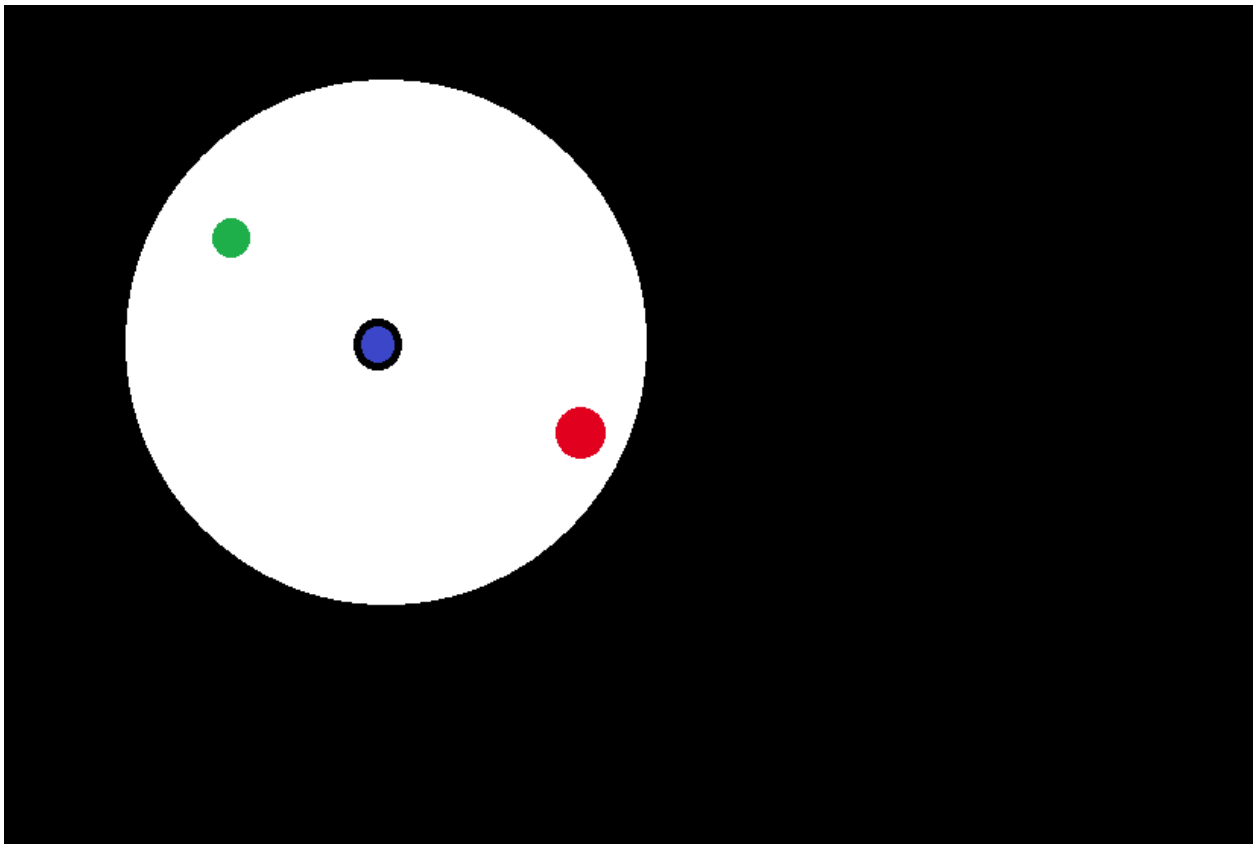
Kivy tutorial suggestion

Go through the Kivy Crash Course: https://inclem.net/pages/kivy-crash-course/
Then write an app of your own.

Pygame Chasing Circles Challenge

This is a challenge for those who are learning the Pygame library:

Create a one-player game of Chasing Circles.



In this game, the player controls a black-outlined circle which starts out blue on the inside. (Blue rgb values (0, 0, 255).) The player's circle can see only so far (six times its body length), it can move up, down, left, and right, and it can "eat" leaves (green circles). Whenever it "eats" a leaf, its blue rgb value goes down by 15 and its red rgb value goes up by 15, so it becomes more red.

The player circle can also be "eaten" by the predator red circle (with rgb values (230, 0, 30)) as long as the predator's red value is bigger than the player circle's red value. But once the player circle's red value is bigger, the player's circle can "eat" the predator.

The goal of the game is to eat the predator.

A leaf appears somewhere at random on the screen, which may be invisible to the player. Once the player circle eats it, another leaf appears somewhere at random.

The predator has the advantage over the player in that it can move in any direction. Its disadvantages are that it doesn't "see" as far (only five body-lengths), and it moves at a speed slightly slower than the player (10% less fast). Whenever it "sees" the player, the predator chases the player. When it doesn't "see" the player, the predator continues going in the same direction, changing direction only when it gets to the edge of the screen or to a green circle.

Once the player becomes red-enough to "eat" the predator, the predator runs away from the player whenever it "sees" the player's circle. The predator also becomes as fast as the player--fear gives it that extra speed. You might need to herd it into a corner to catch it.

Prototypes of Chasing Circles

Prototype 1: Player's circle only, and don't worry about limiting the vision

This simple prototype has only the player's circle that the player can control with up, down, left and right arrow keys.

Prototype 2: Now add the leaf

Add the green circle (the leaf) to Prototype 1. The green circle should appear in a random place on the screen. When the player's circle touches the green circle, the green circle re-appears elsewhere, and the player's circle color changes a bit (more red, less blue, by 15 rgb units).

Prototype 3: Now add the predator

Add the predator circle to Prototype 2. Implement all the behaviors of the predator circle.

Prototype 4: Complete Chasing Circles

Change the visuals on Prototype 3 so that only the part that the player's circle can "see" is visible.