# **AP Computer Science Principles ’19-20 Summer AssignmentAssignment 0**

#

# Watch and take notes on the 1:10 video called:

# **CS50 2018 - Lecture 0 - Computational Thinking, Scratch**

# **(**[**https://www.youtube.com/watch?v=5azaK2cBKGw**](https://www.youtube.com/watch?v=5azaK2cBKGw)**)**

# **before the first day of class, Tuesday, August 13th , 2019.**

#

# There are **TWO** parts to this assignment:

#

# **Part I** is watching the first part of the video (through time stamp 32:35), taking notes and studying the questions below in anticipation of an entrance ticket that I will administer on the first day of class (Day 0). This should take 1-2 hours. The entrance ticket will be graded for correctness.

#

# **Part II** is a project you will build in Scratch. Plan on spending 2-4 hours watching the rest of the video and putting together your Scratch project. The project will be graded by meeting the specifications listed in the appropriate section below.

**Please note:**

If you have trouble completing parts of this assignment, email me at jchamberlain@stfrancishs.org **as soon as you are aware of the problem.** Extensions and/or makeups will only be allowed if I have heard about the issue early enough to help diagnose the problem with time remaining for you to complete it. In other words, watch the video and get started on the Scratch project **before** Wednesday, August 7th. Emails after August 7th will be considered but there is no guarantee that I will award full credit for the assignment if it is not completed by Tuesday, August 13th.

**Part I - Entrance Ticket:**

Be prepared to answer the following questions on an entrance ticket on Day 0:

1. The numbering system that uses only 0’s and 1’s is called \_\_\_\_\_\_\_\_\_.
	1. ASCII
	2. binary
	3. decimal
	4. hexadecimal
2. ASCII translates letters into \_\_\_\_\_\_\_\_\_\_\_.
	1. bits
	2. bytes
	3. numbers
	4. pixels
3. Individual dots on a screen are called \_\_\_\_\_\_\_\_\_\_. (pixels)
	1. bits
	2. bytes
	3. numbers
	4. pixels
4. The three colors that are used to represent color on most computer screens are red, green and \_\_\_\_\_\_\_\_\_\_.
	1. blue
	2. black
	3. cyan
	4. magenta
5. In Computer Science, step-by-step instructions are called \_\_\_\_\_\_\_\_\_\_\_\_.
	1. advanced directives
	2. algorithms
	3. binary code
	4. derivatives
6. Writing down the steps in an algorithm without any kind of programming syntax is called \_\_\_\_\_\_\_\_\_\_.
	1. summarizing
	2. outlining
	3. pseudocode
	4. pseudoscience
7. Making a yes or no decision in a computer program is called a \_\_\_\_\_\_\_\_\_\_.
	1. function
	2. condition
	3. loop
	4. decider

**Part II - Itching to Program:**

Head to <https://scratch.mit.edu/> and sign up for an account on MIT’s website by clicking **Join Scratch** atop the page. Your username should be your email address without the ampersand. For example, if your email address is 22gjones@stfrancishs.org, your username would be “22gjones”. Pick a good password and remember it.

**Make sure that your web browser has Flash enabled. If you’re using Google Chrome, you can go to** chrome://settings/content**, click on Advanced → Content settings, and make sure you "allow" Flash!**

Then head to <https://scratch.mit.edu/help/> and take note of the resources available to you before you dive into Scratch itself. In particular, at least skim the [Getting Started Guide](https://cdn.scratch.mit.edu/scratchr2/static/__95f8025b5d5663c8eca07b96a66ef8d6__/pdfs/help/Getting-Started-Guide-Scratch2.pdf).

Next try your hand at *Pikachu’s Pastry Catch* by (former student) Gabe Walker! Click the green flag and then, per Gabe’s instructions, hit your keyboard’s space bar, at which point the game will begin! Feel free to procrastinate a bit.

If curious, Gabe’s source code can be seen at <http://scratch.mit.edu/projects/26329354/>. (You can also full-screen the game at that same URL, as full-screening the embedded game here might not work.)

Next, be sure you know what’s recyclable and compostable these days by trying out this remix of *Oscartime* by Jordan Hayashi!

Jordan’s source code can be found at <https://scratch.mit.edu/projects/71161586/>. (You can also full-screen that game at that same URL.)

If you’ve no experience (or comfort) whatsoever with programming, rest assured that Gabe’s and Jordan’s projects are more complex than what we expect for this first problem set. (Click **See inside** in Scratch’s top-right corner to look at each project’s underlying "implementation details.") But they do reveal what you can do with Scratch.

In fact, for a gentler introduction to Scratch (and programming more generally), you might want to review some of the examples that can be found at <https://scratch.mit.edu/studios/3003963/>. Once you can say to yourself, "Okay, I think I get this," you’re ready to proceed.

Now it’s time to choose your own adventure! Your mission is, quite simply, to have fun with Scratch and implement a project of your choice (be it an animation, a game, interactive art, or anything else), subject only to the following requirements.

* Your project must have at least two sprites, at least one of which must resemble something other than a cat.
* Your project must have at least three scripts total (i.e., not necessarily three per sprite).
* Your project must use at least one condition, one loop, and one variable.
* Your project must use at least one sound.
* Your project should probably use at least 20 puzzle pieces overall.

Feel free to peruse [some of last year’s projects](https://scratch.mit.edu/studios/3009443/) for inspiration, but your own project should not be terribly similar to any of them. Try to think of an idea on your own, and then set out to implement it. But don’t try to implement the entirety of your project all at once: pluck off one piece at a time. Gabe, for instance, probably implemented just one pastry first, before he moved onto the game’s other sprites. In other words, take baby steps: write a bit of code (i.e., drag and drop a few puzzle pieces), test, write a bit more, test, and so forth.

If, along the way, you find it too difficult to implement some feature, try not to fret; alter your design or work around the problem. If you set out to implement an idea that you find fun, you should not find it hard to satisfy this problem set’s requirements.

Alright, off you go. Make us proud!

Incidentally, if you don’t have the best Internet access, you’re welcome to download Scratch’s "offline editor" at <https://scratch.mit.edu/scratch2download/>. But when done with your project offline, be sure to upload it to your account at <http://scratch.mit.edu/> via **File > Share to website** in the offline editor.

Once finished with your project, click **See project page** in Scratch’s top-right corner. Ensure your project has a title (in Scratch’s top-left corner), some instructions (in Scratch’s top-right corner), and some notes and/or credits (in Scratch’s bottom-right corner). Then click **Share** in Scratch’s top-right corner so that others can see your project. Finally, take note of the URL in your browser’s address bar. That’s your project’s URL on MIT’s website, and you’ll need to know it later.

There will be a space on the multiple choice entrance ticket to input your URL, so be ready to share it on Day 0.