# Beginnings

## Lesson 1: The First Day

| Lesson | Objectives |
|---|---|
| 0.1: The First Day | Identify the class they are taking. List the high-level goals of the course. Describe classroom procedures, rules, and norms. |
| 0.2: Algorithms | Define “algorithm.” Construct algorithms for performing simple tasks. |
| 03: Data Structure | Demo snap program with List and Set feature, explain concept of data structure of List and Set |
| 0.4: Programming Languages | Talk about Sprite and add behavior to sprite in snap programming environments.  Complete small programs (Ballio) in SNAP with guidance. Explain why computer programs are written in specialized languages. |
| 0.5: Prototype | Talk about Prototype, explain why Snap is a prototype language. Ask students to do Clone snap application |
| 0.6 Variable | Talk about Variable, demo snap application and guide students with variable related snap programs. |
| 0.5: Snap Self Portrait | Create a simple “program” in SNAP to describe themselves |
| 0.6: Snap Coordinate System | Create a drawing using the Snap! Coordinate System |

## Learning Objectives

**Students will be able to...**

Identify the class they are taking.
List the high-level goals of the course.
Describe classroom procedures, rules, and norms.

Welcome to CS101! This is an introductory class designed for students who have never had formal exposure to computer science and are looking for a gentle, but thorough introduction to the wonderful world of computing in a free but yet friendly and supportive homeschooling environment. The class size is only limited to 4 students. This class will not only prepare students for future computer science courses, but also empower them to use computer science in their own field of study.

Throughout the program, students can expect to explore a variety of topics, from core computer science principles/concepts like abstraction, recursion and data structures, to apply these principles by building fun assignments like 2048, Mastermind, and some projects of their own choice! What's more? We'll also be covering various applications and implications of computing, including topics like AI, privacy, and algorithmic bias, to explore the exciting, and frightening ways computers are changing the world as we know it. By the end of the semester, students will have learned two programming languages: Snap!, a friendly graphical language, and Python, an industry-standard programming language.

Introduction to Computer Science is an engaging course that explores a variety of basic computational thinking and programming concepts through a project-based learning environment. Every unit culminates in a comprehensive project and roughly 75% of student time is spent building projects and practicing the skills they are learning.

## Visual and approachable

This course uses Snap!, an approachable visual block-based programming language with a robust tool set, perfect for introducing students to coding for the first time.

GitHub: https://github.com/TEALSK12/introduction-to-computer-science/
Github Page: https://tealsk12.github.io/introduction-to-computer-science/

## Topic of the day

**Algorithms:** a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

**Computer Algorithms:** An **algorithm** is simply a set of steps used to complete a specific task. They're the building blocks for programming, and they allow things like **computers**, smartphones, and websites to function and make decisions. In addition to being used by technology, a lot of things we do on a daily basis are similar to **algorithms**.

Example: create a peanut butter & jelly sandwich (ingredients, utensils, plates, napkins, etc.)

**Computer program**: a sequence of instructions or steps, written in a language that can be understood by a computer, that will be used by the computer to complete a task or solve a problem

**Snap**!: https://snap.berkeley.edu/snap/snap.html
https://en.scratch-wiki.info/wiki/Object-Oriented_Programming#OOP_in_Snap.21

**Data Structure:**

> List: in computer science, a **list** or sequence is an abstract **data** type that represents a countable number of ordered values, where the same value may occur more than once. ... **Lists** are a basic example of containers, as they contain other values.
>
> Set: In computer science, a **set** is an abstract **data** type that can store unique values, without any particular order. It is a computer implementation of the mathematical concept of a finite **set**. ... Some **set data structures** are designed for static or frozen **sets** that do not change after they are constructed.

**Sprite**: the thing created in a programming environment with look & feel and behavior.

**Prototypes**: A prototype is an early sample, model, or release of a product built to test a concept or process or to act as a thing to be replicated or learned from. It is a term used in a variety of contexts, including semantics, design, electronics, and software programming. A prototype is generally used to evaluate a new design to enhance precision by system analysts and users. Prototyping serves to provide specifications for a real, working system rather than a theoretical one
It can act as a prototype for making many many copies so it's like a blueprint or prototype. Prototype means to make more of the same thing. Snap is also called a prototype based language or programming

**Variable**: It is used to store information and keep it up to date.

**Lab 1:**

**List & Set Demo**

https://snap.berkeley.edu/snap/snap.html#present:Username=hesam&ProjectName=lws_list

https://snap.berkeley.edu/snap/snap.html#present:Username=hesam&ProjectName=lws_set

Students spend time on the above program.

**Sprite**: the thing created in a programming environment with look & feel and behavior.

http://52.88.187.27/cs/courses/computing-math/2.html

https://snap.berkeley.edu/snap/snap.html

**Sprite url**

**Sprite areas:**

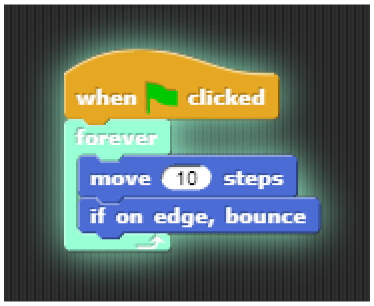
Stage: create/edit sprite),

Script: tell sprite when and what to do

Tool box: tiles which define functionality and behavior

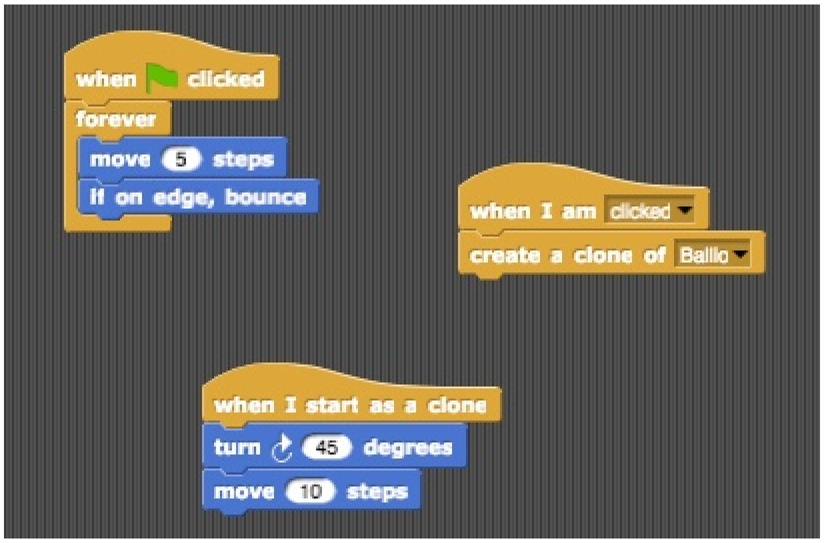
**Lab 2:** Create a sprite with customized costume and save/export the application/program

**Lab 3:** write simple snap program



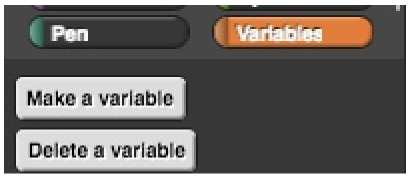
```
when ⚑ clicked
forever
  move 10 steps
  if on edge, bounce
```

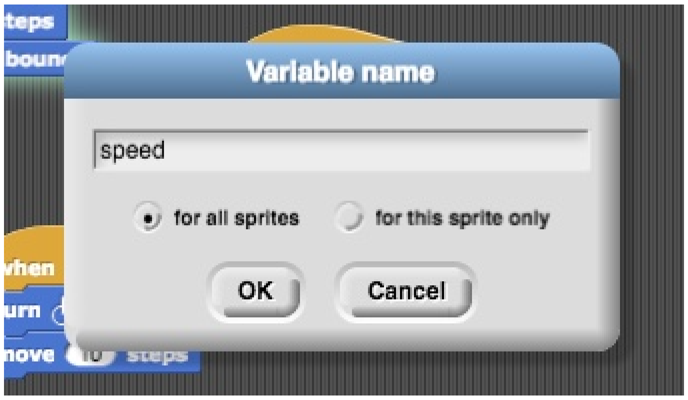
**Lab 4:** write simple snap program with Prototype concept



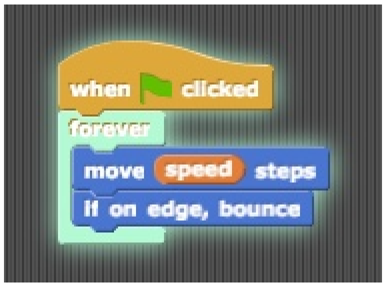
```
when [flag] clicked
forever
    move (5) steps
    if on edge, bounce

when I am [clicked]
create a clone of [Ballio]

when I start as a clone
turn ↻ (45) degrees
move (10) steps
```

**Lab 5:** write simple snap program with Variable concept





Variable name: speed
[● for all sprites] ○ for this sprite only



```
when [flag] clicked
forever
    move (speed) steps
    if on edge, bounce
```

**Lab 6: Spend more time on Snap to try out [https://github.com/TEALSK12/introduction-to-computer-science/blob/master/Unit%200/lab_04.pdf](https://github.com/TEALSK12/introduction-to-computer-science/blob/master/Unit%200/lab_04.pdf)**

**Lab 7: Write dog chasing snap program (Bonus)**