

Lesson Plan

- Lesson 1: Learning Sequence
- Lesson 2: Learning Branch, Jump (), goto ()
- Lesson 3: Making decisions, Conditional – if ()
then else ()
- Lesson 4: Fixing Errors, Bug and Debugging
- Lesson 5: Looping with repeat, bounded loops
- Lesson 6: Understanding Functions**

Bonus lessons

- Lesson A: Introducing operations, greater, less than. Boolean - TRUE, FALSE
- Lesson B: Introducing Variable, string and numeric
- Lesson C: Nested repeat – Loop in Loop

Lesson 6: Functions

Lesson Overview

Students will do a concept introductory worksheet. Then they will play an advance game level and will be introduced to the function concepts in coding. Finally, they will write their code as an algorithm.

Lesson Objective

- First students will do an introduction worksheet to introduce the concept of functions. In the process they will create functions from series of multiple actions. A function could be used over and over
- Then they will play a level of CoderBunnyz. Based on where they are located in relevance to the carrot, they will create a function and then use that and other code cards for the game play.
- Finally they will count the number of code cards used as instructions and write/draw the sequence of cards taken. This is called algorithm writing.

Materials needed

- Function worksheet (on the next page), game, algorithm sheet 6.1, pencil

Getting Started

- Instructor will explain the worksheet. Players will then do the exercise.
- After the worksheet is complete, arrange the board and cards based on Rules book level 2.2(page 13), explain the cards, especially the function cards to the student.
- Choose the destination
- The first step is to create a function. Review the maze and let the students think, what code they would need to use again and again. It could be combination of 2 cards or 3 cards or may be

even 4 cards (see example in rulebook). Then create that sequence as a function.

- Once function is created, get ready to start the game.

Activity

- Play level 2.2 of CoderBunnyz to program your bunny to eat their colored carrot, jump over the puddles, find their way around the maze and reach the destination.
- Use the function card (f) wherever applicable or till you run out of function cards. Continue till all players reach the destination.
- Then each player will review their code cards. That's the sequence of code (including the functions) that they will write on their algorithm sheet.
- Count the number of cards used to reach the destination and write those on the sheet 6.1. Also, write the algorithm of the game played.

Fun Fact

The first high-level (very close to real English that we use to communicate) programming language was FORTRAN, invented in 1954 by IBM's John Backus.

Read aloud

6. Function

A block of re-useable code you can use to do a certain task many times



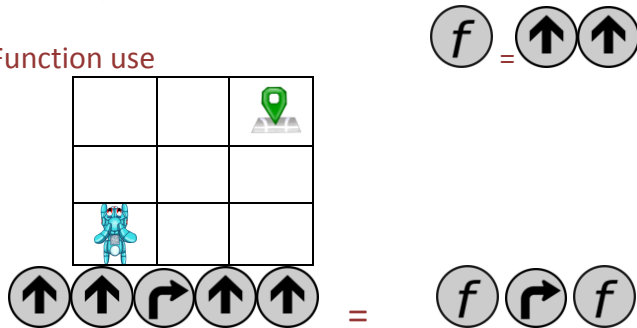
What is a function?

Read aloud

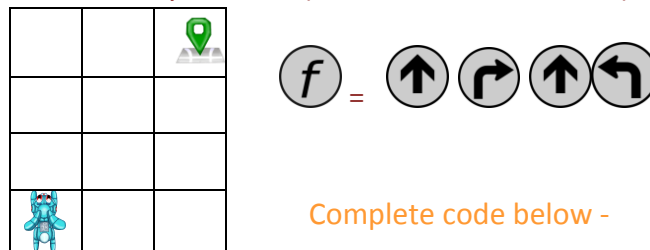
- A piece of code that can be called over and over.
- Function helps write efficient code

Read aloud

Function use



Create code for the bunny to reach the destination using the function already defined (Use : stair case function)



Complete code below -



Find a few case in which you use functions in your day to day life.

(Hint – Think of a routine of something that you do over and over)

1)

2)

Practice Exercise

Practice Exercise





6.1 Function



Look at your code : Count how many code-cards you use in your coding today:

_____ MoveForward ();

_____ TurnLeft ();

_____ TurnRight ();

_____ Jump ();


_____ Function ();



Add these three, You use _____ Total_Code_Cards();

-----ADVANCE-----

Write your Code for today's game. (Use back of your page to continue)

Hint: Use arrows ( ,  ,  ,  , f) to write your moves or use codewords (MoveForward (); Function();..) or

both! **How to use function? First** write your $f = \uparrow \curvearrowright$ **OR** Function() = { MoveForward(); TurnLeft (); }

Then use "f" or "Function ()" in your code.

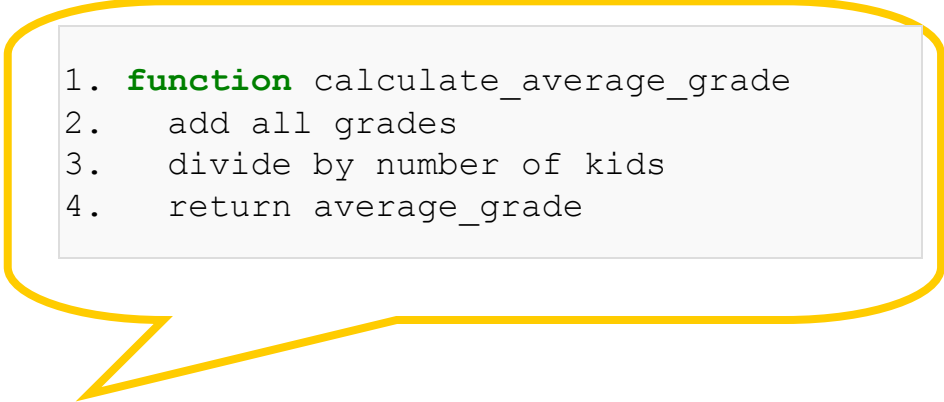
How are functions used in real programming languages?

Functions are used by all the programming languages to perform multiple steps of computation or action. Many programming languages provide inbuilt functions.

Why Functions ?

- They allow us to see our program as a bunch of smaller steps. Each step - a function - can break the code into smaller steps making it easier for us to understand and write more efficient code.
- They allow us to reuse the code, instead of rewriting them.

A code example in many programming languages using function -



```
1. function calculate_average_grade
2.   add all grades
3.   divide by number of kids
4.   return average_grade
```

Now if a sixth grade teacher needs to calculate the classes' average grade in the first quarter of the year, they just call function `calculate_average_grade`.

It's easy to create a list of many different functions that could be used by all languages. Many of these languages now come with a list of inbuilt functions that makes the programming easier.