CL1002
Programming Fundamentals

Lab 01
Problem solving in the context of programming fundamentals, Problem Analysis Charts, Input Process Output, Introduction to flowcharts, Pseudocode, Algorithms, Introduction to scratch

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Fall 2025

AIMS AND OBJECTIVES

The aim of this lab is to equip students with the foundational knowledge and skills necessary to understand and apply fundamental programming concepts. The objectives of this lab are as follows.

1. Equip students with foundational knowledge of fundamental programming concepts.
2. Develop problem-solving skills through real-world examples.
3. Introduce students to the PAC (Problem Analysis Chart) methodology.
4. Familiarize students with IPO (Input-Process-Output) Charts.
5. Teach students how to create and interpret Flowcharts.
6. Understand the concept of pseudocode and its role in program design.
7. Learn the specific keywords and operations used in pseudocode.
8. Develop the ability to write algorithms that outline the logic of a program.
9. Learn scratch programming

INTRODUCTION

In this lab we will be covering the following topics

1. Problem solving in the context of programming fundamentals.
2. Introduction to Problem Analysis Charts.
3. Introduction to Input Process Output Table.
4. Introduction to flowcharts.
5. Pseudocode
6. Algorithms
7. Introduction to scratch

OBJECTIVE: 1 PROBLEM SOLVING

What is Problem solving?

•**Definition**: Problem-solving in programming is identifying a problem, planning a solution, and executing that solution to achieve the desired result.

•**Importance**: It's fundamental in programming because it helps create efficient, effective, and scalable solutions.

EXAMPLE

Imagine that you have the following image, which is a map of a road leading to the building shown in the picture.

- There is a car and trees.
- The car cannot cross the trees
- The road is divided into squares to calculate the steps of the car.
- Each square is considered as one step.
- The car can be moved omni-directional.

How can the car arrive at the building?

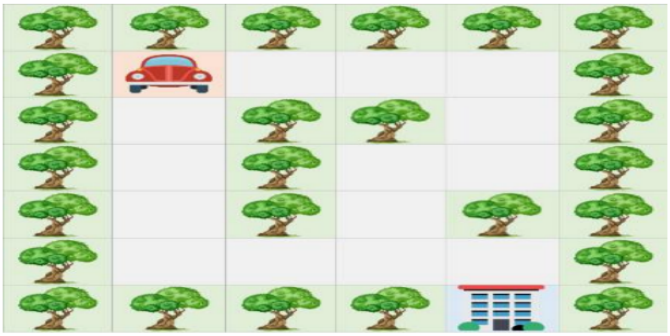


Figure: 1(a) Problem solving

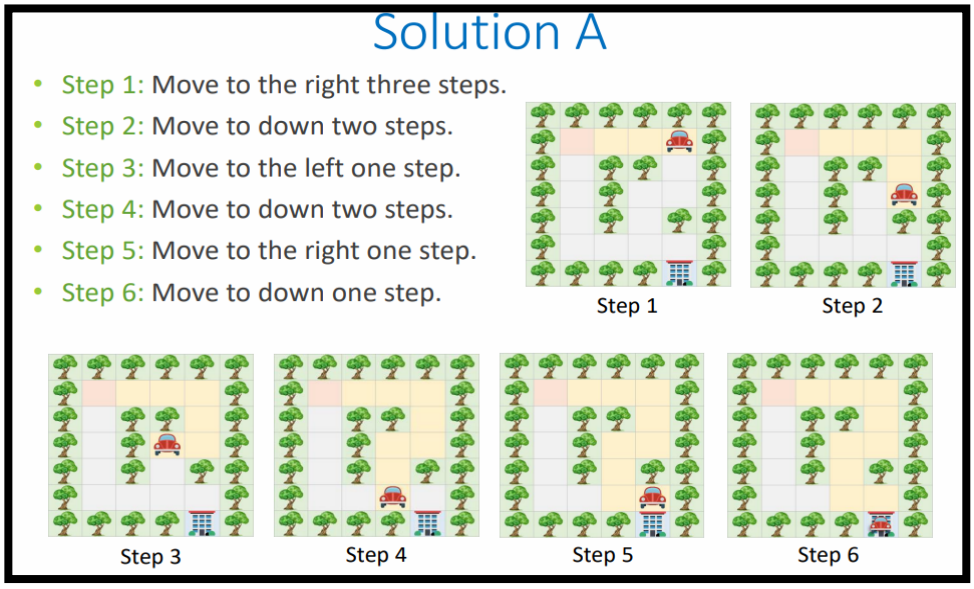


Figure: 1(b) Problem Solving

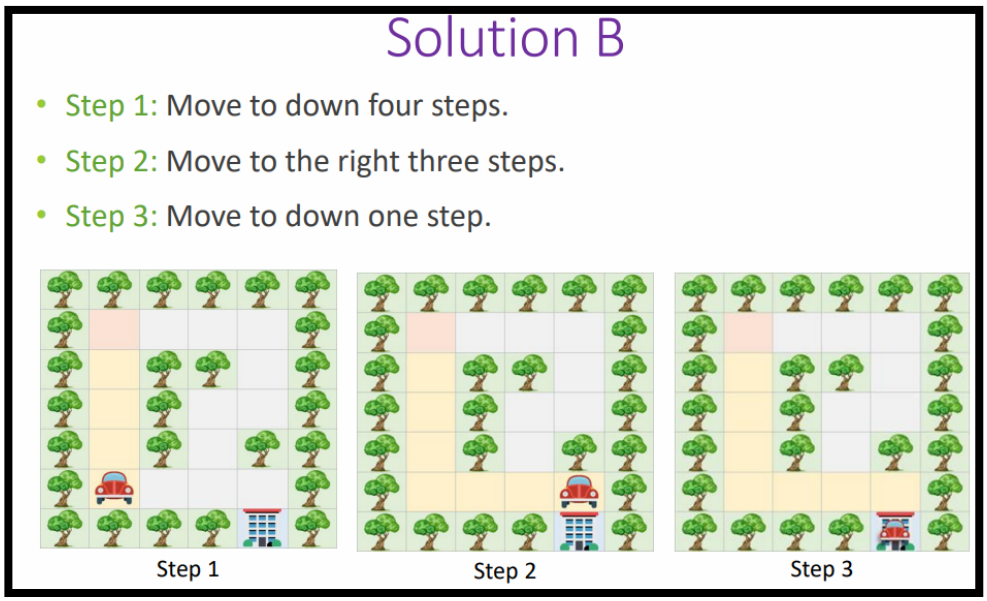


Figure: 1(c) Problem Solving

As we can see **Solution A** and **Solution B** are both correct solutions to the same problem, but there are differences in the complexity and efficiency of the solutions.

The cost of Solution A is 10 steps while Solution B is 8, so we consider Solution B a better solution based on the number of steps.

Reducing the number of steps in the previous example means reducing the fuel needed by the vehicle and speeding up the arrival time.

OBJECTIVE: 2 PROBLEM ANALYSIS CHART

The Problem Analysis Chart (PAC) is a dependent tool used in troubleshooting problems, especially in programming and systems development. It allows problem to be broken down into smaller, more manageable parts and is often used to plan and document the steps needed to solve a problem using a programming language like C.

Given Data	Required Results
Section 1: Data given in the problem or provided by the user. These can be known values or general names for data, such as price, quantity, and so forth.	Section 2: Requirements for the output reports. This includes the information needed and the format required.
Processing Required	Solution Alternatives
Section 3: List of processing required. This includes equations or other types of processing, such as sorting, searching, and so forth.	Section 4: List of ideas for the solution of the problem.

Figure: 2 An Example of a Problem Analysis Chart (PAC)

In a Problem Analysis Chart there are 4 sections that we need to concentrate on. The Top Left section is the data for the problem we are currently trying to solve, so it will include things such as variables, numerical data, percentile and more. The Top Right section is the desired output we require which is what the problem is based on. The bottom left section is the processing section where the necessary steps are enlisted to ensure the correct output for the required results, and the bottom right section are alternate solutions as a problem can have multiple solutions to achieve the same result.

EXAMPLE

We have a problem in which we want to calculate the gross pay of an employee at the department store that you were previously working as a cashier (assume you are the general manager of that department store). Since you studied well you remember the formula for calculating the formula for gross pay. But I will write it down anyway for ease of use.

GrossPay = Hours*PayRate
Payrate = 20$/HR.
Hours = 12

We need to develop a PAC chart for the above-mentioned problem.

Given Data	Required Results
Hours Pay Rate	Gross Pay
Processing Required	Solution Alternatives
*GrossPay = Hours * PayRate*	1. Define the hours worked and pay rate as constants. *2. Define the hours worked and pay rate as input values.

Figure: 3 Example of PAC chart

In the problem listed we identified that Hours and Pay Rate are the data that we are given so it is fed into the top left section of the chart. We require the gross pay of the employee so that goes to our required results section. A formula is required to calculate the gross pay so we can put it in the processing section, finally we can have multiple solutions to the problem as we can have an adaptive pay scale and hour scale.

Some more problems to solve—

Create PAC Charts of the following problems

Problem #1

Imagine you are designing a student details record system. Create a PAC chart that collects information from the user about a student and calculates the obtained marks and percentage based on the marks entered. The program should include the following steps:

- ✓ Ask the user to enter the student's ID, student's Name, marks for Subject 1, marks for Subject 2, and marks for Subject 3.
- ✓ Calculate the obtained marks by adding the marks for all three subjects.
- ✓ Calculate the percentage obtained by dividing the obtained marks by the maximum possible marks (assuming each subject has a maximum of 100 marks) and multiplying by 100.
- ✓ Display the student's details (ID and Name), the obtained marks, and the calculated percentage.

Problem #2

Create PAC chart that calculates the area of a rectangle. Ask the user for the length and width, then display the calculated area?

Problem #3

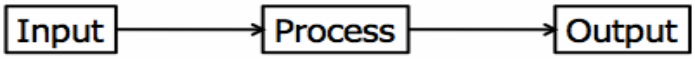
Design a PAC chart that converts Celsius to Fahrenheit. Ask the user for a temperature in Celsius and display its equivalent in Fahrenheit using the formula: Fahrenheit = (Celsius * 9/5) + 32.

OBJECTIVE: 3 INPUT PROCESS OUTPUT

The IPO (Input-Process-Output) model is used in programming and system design for several important reasons. It provides a clear, structured way to conceptualize and develop software by breaking down the process into three fundamental steps: Input, Process, and Output.

IPO Chart shows:

- What data item are input.
- What processing takes place on that data.
- What information will be the result, the output.
- Where in the solution the processing takes place.



EXAMPLE

Input	Processing	Module Reference	Output
Hours Worked Pay Rate	1. Enter Hours Worked 2. Enter Pay Rate 3. Calculate Pay 4. Print Pay 5. End	*Read* *Read* *Calc* *Print* *PayRollControl*	Gross pay

Figure: 5 An example IPO Chart of a Payroll System

In an IPO Chart we have 4 sections. Input, Processing, Module Reference and Output. Like the PAC chart we need to identify what are the input variables in our problem. The processing section has the same effect as the PAC chart but now in a more detailed layout where each step is enlisted. The Module Reference is one change from the PAC chart where each step of the processing section is converted into a command to which a computer will interpret. Such as Read, Calculate, Print, Loop, Write and Program control.

Some more problems to solve

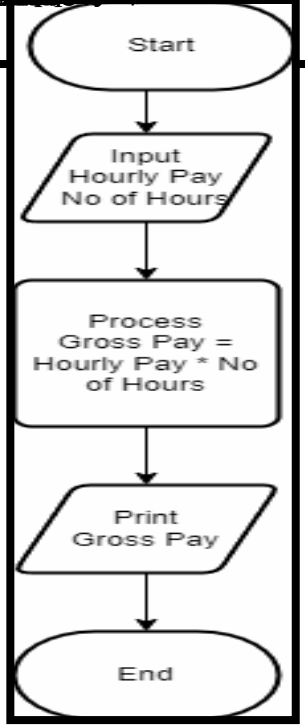
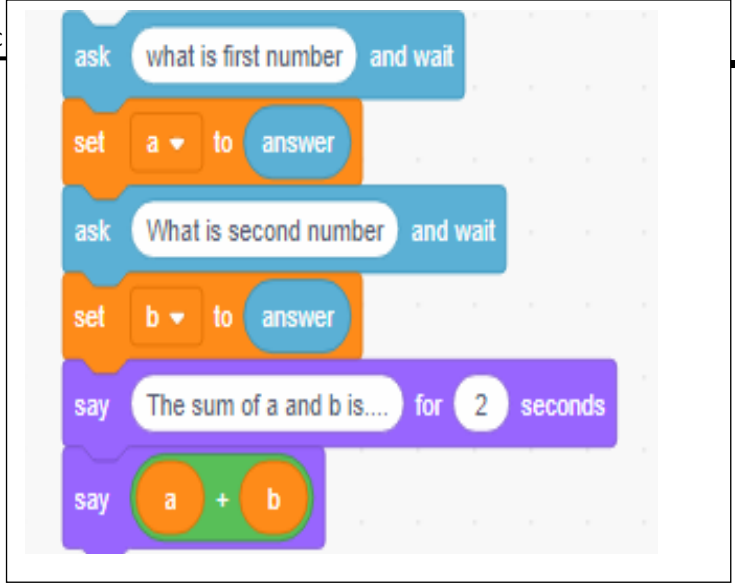


TABLE: 1 BASIC



OBJECTIVE: 6 ALGORITHM

An algorithm is a finite set of instructions that, if followed, accomplishes a particular task.

OR

An algorithm is a set of obvious, logical, and sequential steps that solve a specific problem.

Characteristics of a Good Algorithm:

- Clear and Unambiguous: Each step is precisely defined.
- Finite: It has a definite end.
- Efficient: It accomplishes the task using minimal resources.

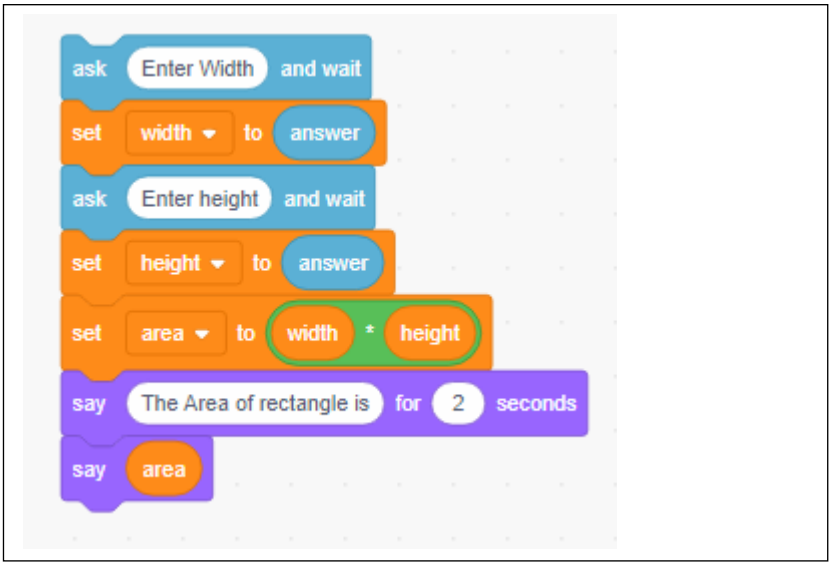
EXAMPLE

Write an algorithm that can calculate the area of a rectangle. The width and the height of the rectangle should be taken from the user.

Note: Area = Width × Height

Solution A – Good:

1. Ask the user to enter **Width**
2. Ask the user to enter **Height**
3. Set **Area** to **(Width × Height)**
4. Display Area for the user



• You can describe the steps in your way, but your description of the steps should be obvious, logical, and sequential.

Solution B - Bad:

1. Ask the user to enter **Width**
2. Ask the user to enter **Height**
3. Calculate **Area**
4. Display Area for the user

The reason for considering Solution B as a bad solution:
Step 3 is not clear because it does not explain how we can calculate Area. So, this algorithm is bad because its steps are not obvious.

Solution C - Bad:

1. Set Area to (**Width × Height**)

2. Ask the user to enter **Width**

3. Ask the user to enter **Height**

4. Display Area for the user

The reasons for considering Solution C as a bad solution:

We don't know what Width and Height at Step 1 are. In other words, Width and Height have not been defined before Step 1, so we cannot use them because they do not exist yet.

What about Step 2 and Step 3? Width and Height are defined there! After Step 2, Width does exist, but Height does not. After Step 3, Height does exist. Both Width and Height are available to be used at or after step 4.

So, this algorithm is bad because its steps are not correctly sequential.

Solution D - Bad:

1. Set Area to (**Width × Height**)

2. Display Area for the user

The reasons for considering Solution D as a bad solution:

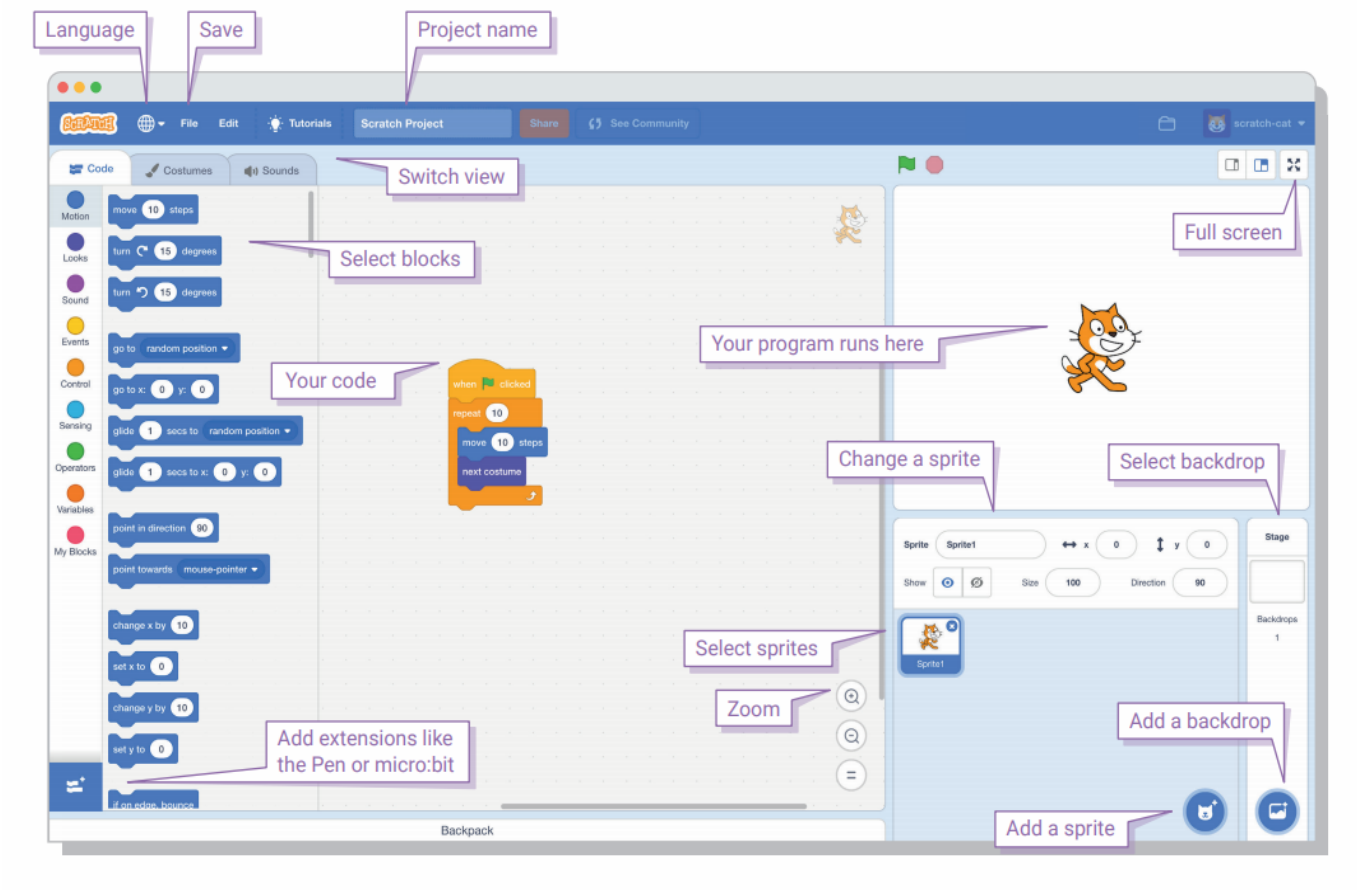
• Step 1 tells us to multiply Width and Height, but we don't know what Width and Height are. Even, they have not been defined in any steps of the algorithm.

• So, this algorithm is bad because of the illogical step, which is using unknown things (Width and Height).

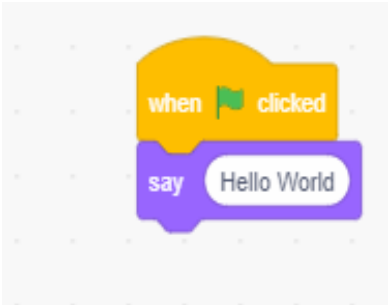
OBJECTIVE: 7 INTRODUCTION TO SCRATCH

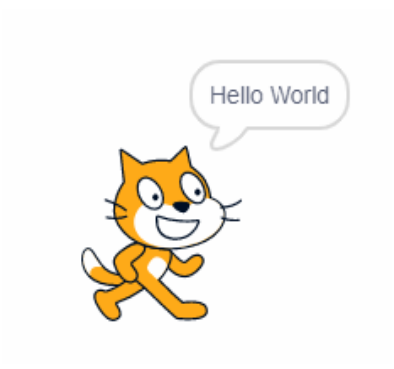
Visual Programming Language (VPL) is an application development environment designed on a graphical block-based programming model.

Scratch Interface



Example#1: Writing our first script in a scratch to print "Hello" to the world.

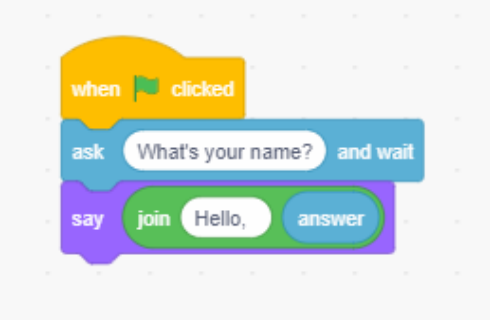




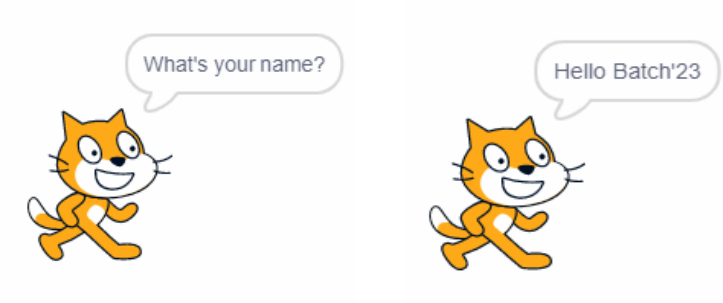

```
#include <stdio.h>
    int main()
{
    printf("Hello World");
}
```

Example#2: Prompting your Name and printing "Hello" to your Name.

Script:

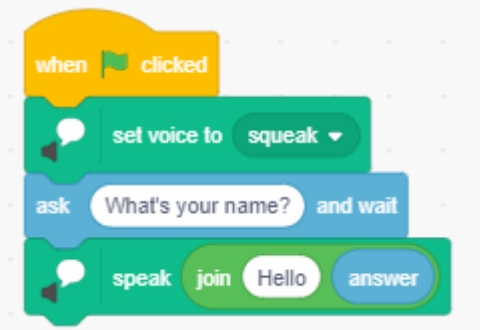


Output:



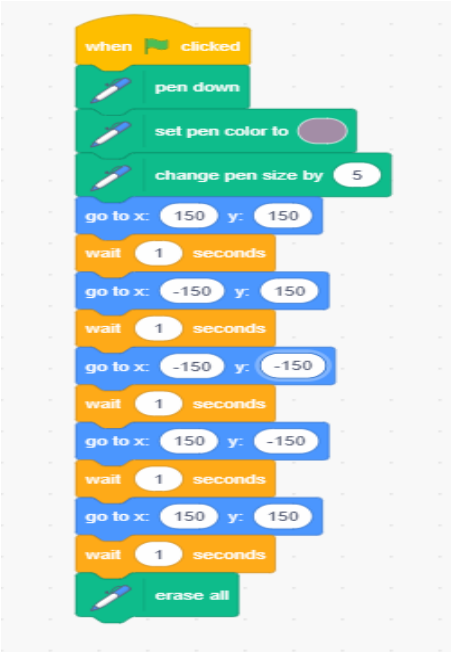
Example#3: Asking your Name and saying by speaking "Hello" to your Name.

Script:

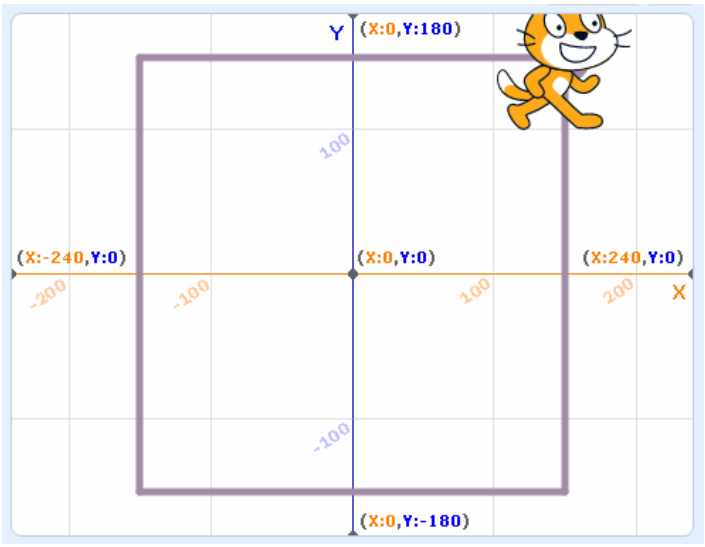


Example#4: Setting Grid Backdrop and moving spirit onto the xy-axis and draw square on the screen.

Script:

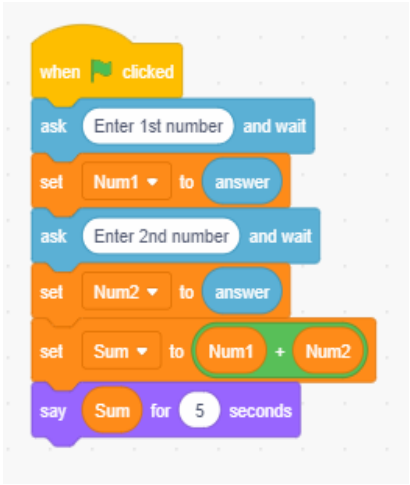


Output:



Example#5: Write a script of input two numbers, add two numbers and print the sum of numbers.

Script:

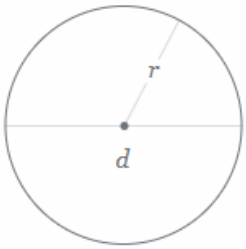


Output:

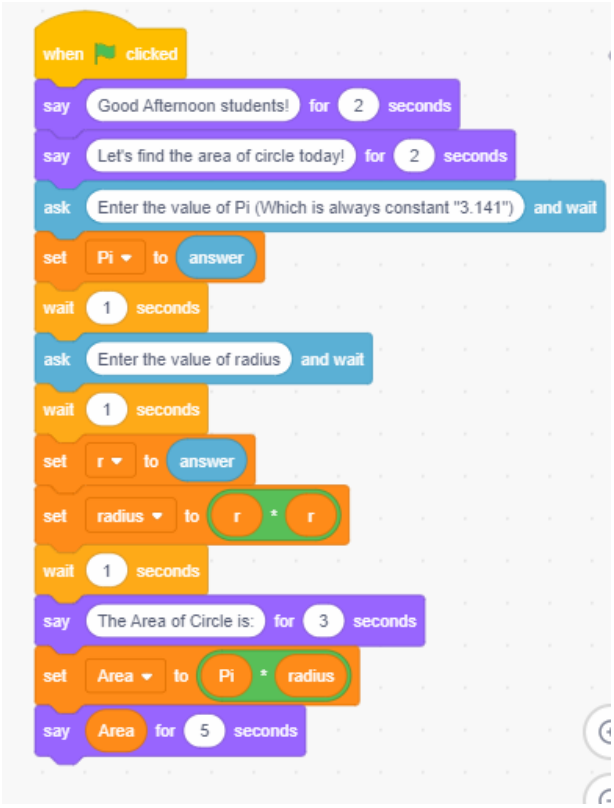
Example#6: Write a script to find an area of circle

Formula:

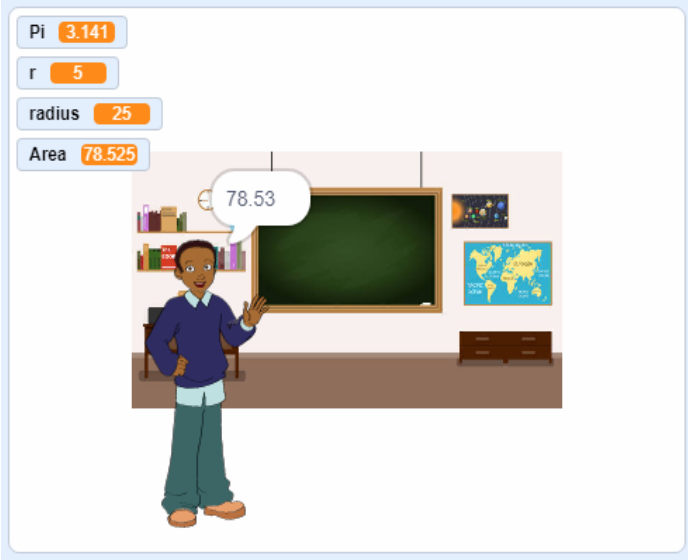
$$A = \pi r^2$$



Script:

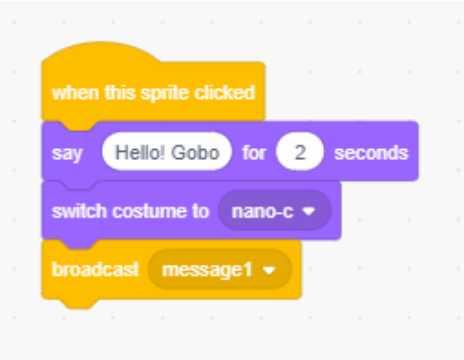


Output:



Example#7: Conversation between two scripts when clicked on the spirit.

Spirit#1



Spirit#2