CPT 121: INTRODUCTION TO PROGRAMMING

2 Units (PART ¼)

COURSE OUTLINE

1. Brief survey of programming paradigms: Classification; procedural, object-oriented, functional.

2. Introduction to Visual Basic (VB): History, versions.

3. Fundamental programming constructs in VB: Syntax and semantics; variables, types, expressions, and assignment; simple I/O; conditional and iterative control structures, testing and debugging strategies.

CPT 121: INTRODUCTION TO PROGRAMMING

2 Units (PART ¼)

COURSE OUTLINE

1 Brief survey of programming paradigms

- **Classification**
  - ❑ Procedural
  - ❑ Object-oriented
  - ❑ Functional

RECAP *(what you should know by now)*

- Before we go any further lets make sure we have a proper foundation that can answer the following questions

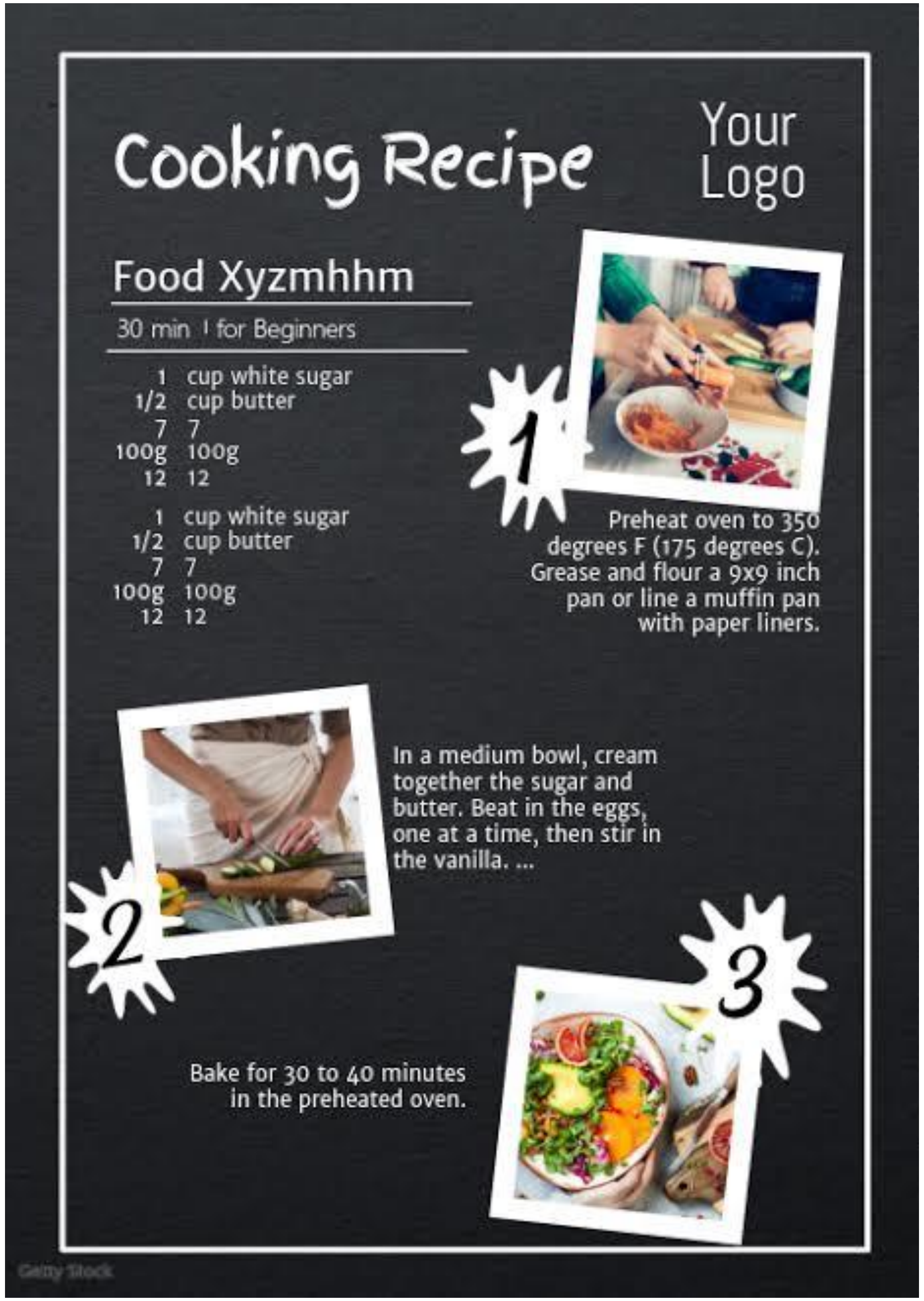
1. What is a program?
2. Who is a programmer?
3. What is a programming?
4. What is a Programming Language?
5. What is a paradigm?
6. What are programming Paradigms?

1 WHAT IS A COMPUTER PROGRAM?

i A computer program is a collection of instructions that performs a specific task when executed by a computer *(Knuth, Donald E. (1997))*. A computer requires programs to function and typically executes the program's instructions in a central processing unit.

ii. An organized list of instructions that, when executed, causes the computer to behave in a predetermined manner. Without programs, computers are useless.

iii A computer program is very similar to a cooking recipe, which can be defined as a set of instructions for preparing a particular dish, including **A LIST** of the ingredients required **INSTRUCTIONS** on what to do



Cooking Recipe

Your Logo

Food Xyzmhhm

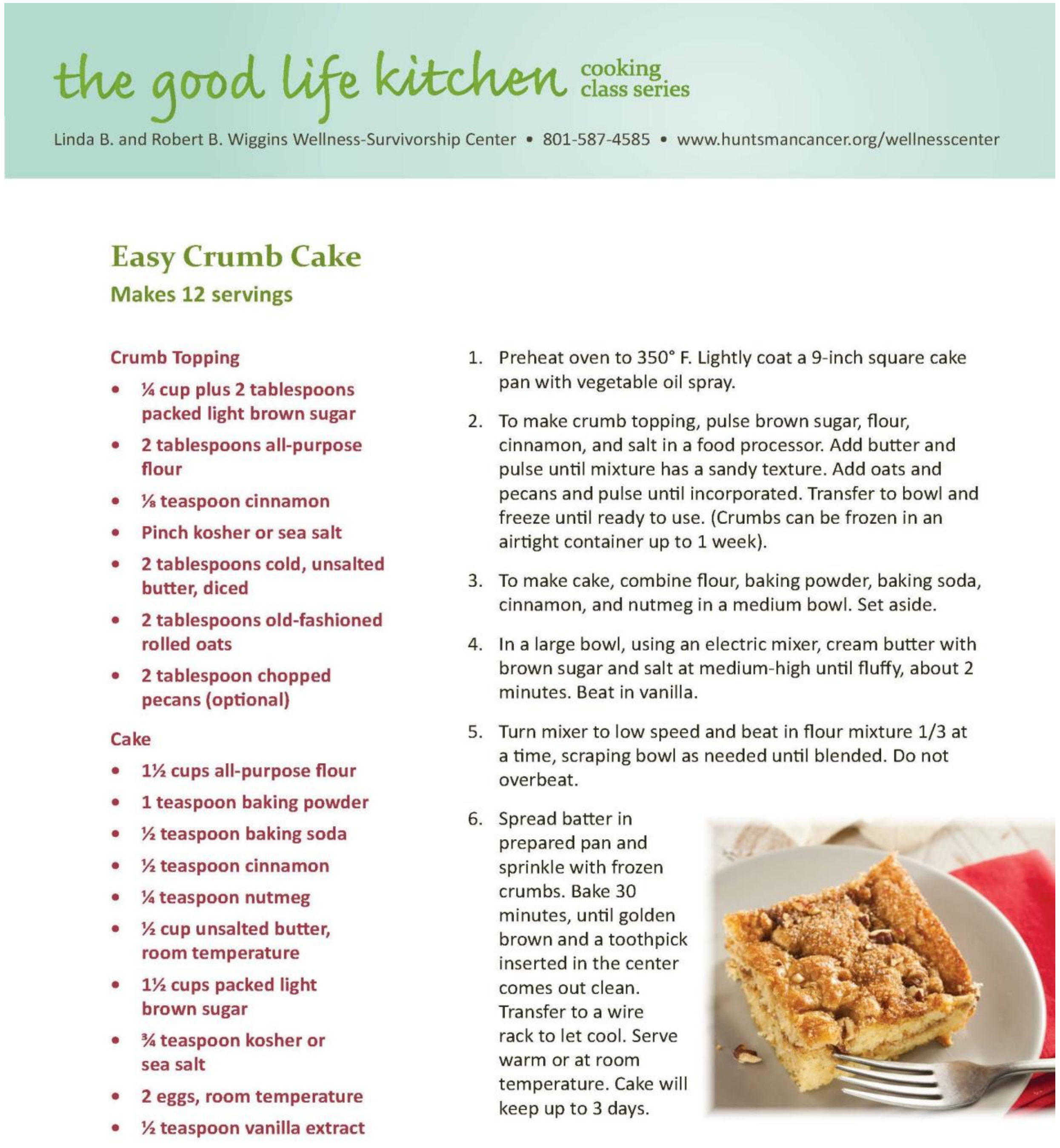
30 min | for Beginners

1	cup white sugar
1/2	cup butter
7	7
100g	100g
12	12
1	cup white sugar
1/2	cup butter
7	7
100g	100g
12	12

1 Preheat oven to 350 degrees F (175 degrees C). Grease and flour a 9x9 inch pan or line a muffin pan with paper liners.

2 In a medium bowl, cream together the sugar and butter. Beat in the eggs, one at a time, then stir in the vanilla. ...

3 Bake for 30 to 40 minutes in the preheated oven.



the good life kitchen cooking class series

Linda B. and Robert B. Wiggins Wellness-Survivorship Center • 801-587-4585 • www.huntsmancancer.org/wellnesscenter

Easy Crumb Cake

Makes 12 servings

Crumb Topping

- ¼ cup plus 2 tablespoons packed light brown sugar
- 2 tablespoons all-purpose flour
- ⅛ teaspoon cinnamon
- Pinch kosher or sea salt
- 2 tablespoons cold, unsalted butter, diced
- 2 tablespoons old-fashioned rolled oats
- 2 tablespoon chopped pecans (optional)

Cake

- 1½ cups all-purpose flour
- 1 teaspoon baking powder
- ½ teaspoon baking soda
- ½ teaspoon cinnamon
- ¼ teaspoon nutmeg
- ½ cup unsalted butter, room temperature
- 1½ cups packed light brown sugar
- ¾ teaspoon kosher or sea salt
- 2 eggs, room temperature
- ½ teaspoon vanilla extract

1. Preheat oven to 350° F. Lightly coat a 9-inch square cake pan with vegetable oil spray.
2. To make crumb topping, pulse brown sugar, flour, cinnamon, and salt in a food processor. Add butter and pulse until mixture has a sandy texture. Add oats and pecans and pulse until incorporated. Transfer to bowl and freeze until ready to use. (Crumbs can be frozen in an airtight container up to 1 week).
3. To make cake, combine flour, baking powder, baking soda, cinnamon, and nutmeg in a medium bowl. Set aside.
4. In a large bowl, using an electric mixer, cream butter with brown sugar and salt at medium-high until fluffy, about 2 minutes. Beat in vanilla.
5. Turn mixer to low speed and beat in flour mixture 1/3 at a time, scraping bowl as needed until blended. Do not overbeat.
6. Spread batter in prepared pan and sprinkle with frozen crumbs. Bake 30 minutes, until golden brown and a toothpick inserted in the center comes out clean. Transfer to a wire rack to let cool. Serve warm or at room temperature. Cake will keep up to 3 days.

2 Who is a Computer Programmer?

A person who writes computer programs.

3 What is a programming?

Computer programming (often shortened to programming) is a process that leads from an original formulation of a computing problem to executable computer programs. Or simply put the art of writing Computer programs.

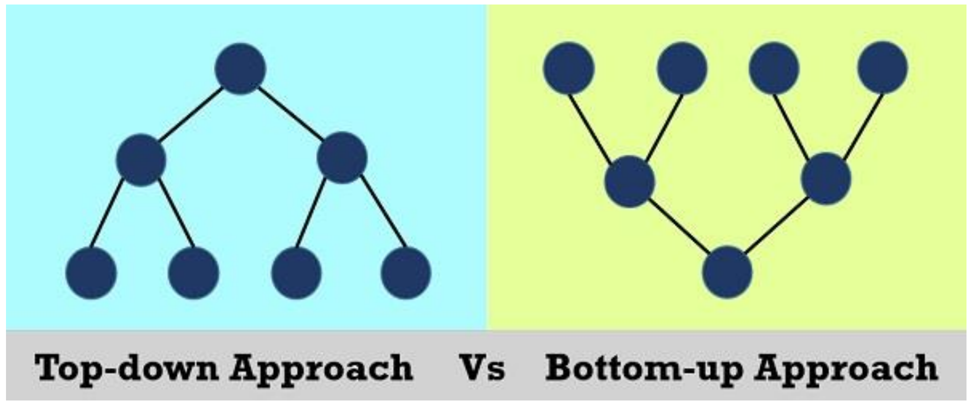
* So a **Computer Program** is to **A Recipe** as **Computer Programmer** is to **The Author** of the Recipe and **Programming** as to the **Art of writing** a recipe

4 What is Programming Language?

1. A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output.

2. A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks

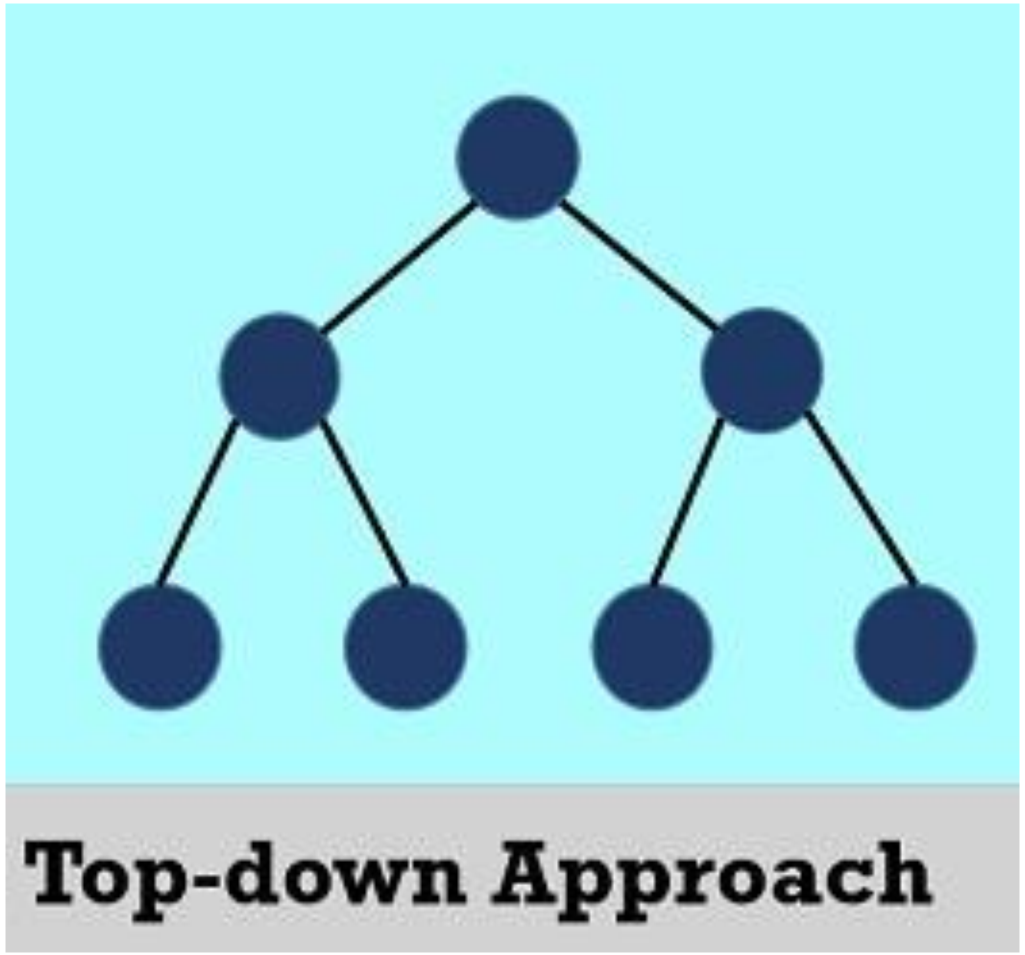
TOP DOWN VS BOTTOM UP DESIGN APPROACH



- They are both strategies of information processing and knowledge ordering, used in a variety of fields including software development
- A top-down approach (also known as stepwise design and in some cases used as a synonym of decomposition)

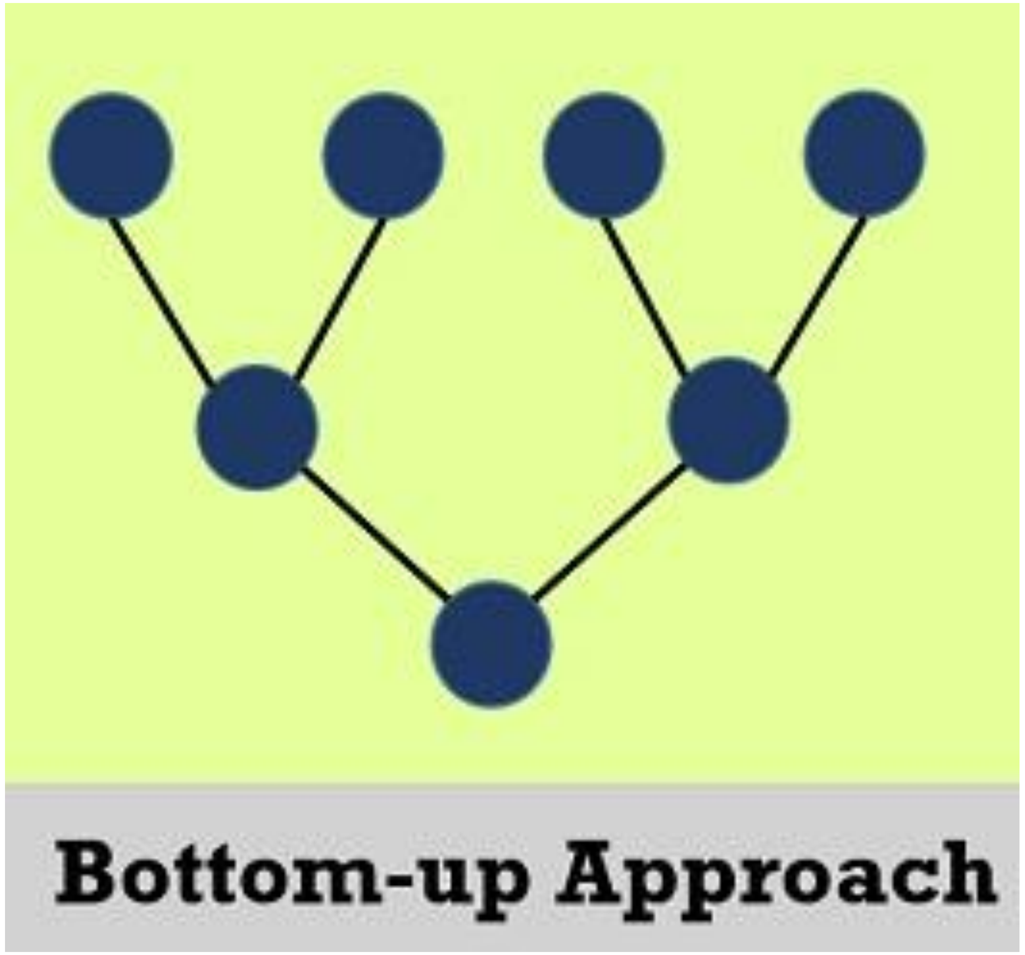
TOP DOWN

- Design begins by specifying complex pieces and then dividing them into successively smaller pieces.
- The main procedure is then broken down into sub functions to gain insight into its compositional sub-systems



BOTTOM UP

- In a bottom-up approach the individual base elements of the system are first specified in great detail.
- These elements are then linked together to form larger subsystems
- They then link sometimes in many levels, until a complete top-level system is formed



6 Steps of the Programming Process by Peter Bruce

- Define problem.
- Develop algorithm.
- Write code.
- Enter and compile.
- Execute and debug (run, fix errors).
- Document and maintain.

STEP 1: DEFINE THE PROBLEM

- Break the problem down into 3 parts:
- **INPUT** – What data do you need to find the answer? (And what data types and identifier names will you need?)
- **PROCESS** – What are the steps needed to arrive at the answer?
- **OUTPUT** – What is the result/answer you are looking for?

STEP 2: DEVELOP THE ALGORITHM

- Give more step-by-step details of how you are going to solve the problem.
- This step can be written in “psuedo-code”.
- This part of the planning does not get typed into the computer!

STEP 3: WRITE PROGRAM CODE

- Translate the steps of your algorithm into program code.
- *Example.*

Q Basic

```
PRINT "This program adds 2 numbers together"
PRINT "Enter the 1st number: "
INPUT , number1
PRINT "Enter the 2nd number: "
INPUT, number2
sumTotal=number1+number2
PRINT "The sum total is: "; sumTotal
```

STEP 4: ENTER AND COMPILE

- This is when you finally go to the computer!
- Type in you’re a programing language and compile to make sure there are no syntax errors and that the computer understands what you have entered.

STEP 5: EXECUTE AND DEBUG (RUN, FIX ERRORS)

- Once you have entered and compiled your program, you are ready to see if it works.
- When you run the program, you will see if it works as you want it to. If it does not, you make adjustments until it does!

STEP 6: DOCUMENT AND MAINTAIN

- Once the program is working correctly, you want to make sure you have documented / added comments to make the program easy for someone to understand and follow.

- If additional information is to be added, make sure you keep your program up-to-date with the assignment.

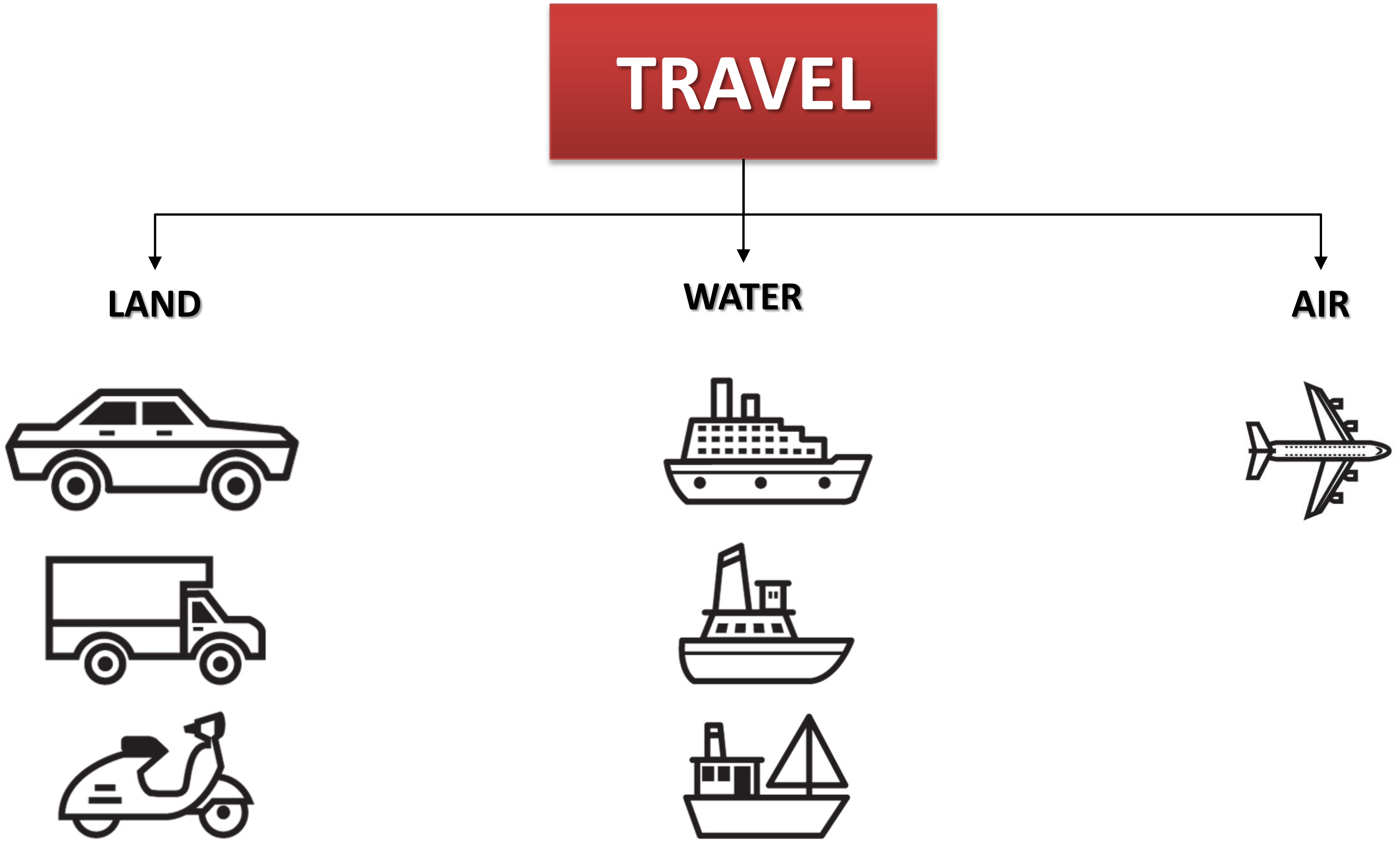
- This is the time to take a little pride in your work and make sure the program "looks nice!" Use formatting to make the work easy to read and understand!

BRIEF SURVEY OF PROGRAMMING PARADIGMS

- Programs as know are written to solve problems
- However, there exists different types of problem; (simple/complex, small/large, etc)
- For that reason, different problem would require different solutions; (simple problems/ simple solutions, complex problems, complex solution)
- This led to the creations of different programming paradigms
- Some more suitable for the solution of a given group of problems and others better fitting for another given group of problems
- That is why we have different programming paradigms for different programming language, at different points in time

5 What is a paradigm?

- A paradigm means a pattern or model (Oxford dictionary)
- Other word often used to describe paradigms are style of going about a particular task



What are programming Paradigms?

- Programming paradigms are a way to classify programming languages based on their features, structures and programming styles

- Languages can be classified into multiple paradigms

- Popular Programming paradigms are

- ❑ Procedural
- ❑ Object-oriented
- ❑ Functional
- ❑ Symbolic programming
- ❑ Knowledge-based programming
- ❑ Declarative programming

Classification of programming paradigms

PROCEDURAL: involves the execution of series of computational steps to be carried out (**Procedure;** routines, subroutines)

- **Procedure** are the building blocks of PPL. EG of PPL include Fortran, ALGOL, COBOL, BASIC, Pascal ,**C** etc

OBJECT-ORIENTED: involves the use of objects that interact with each other by passing messages that transform their state.

- Its building blocks are object modelling, classification and inheritance. EG of OOP PLs are JAVA, Python, C++, VB etc

FUNCTIONAL: Involves writing programs in a style that treats computation as the evaluation of mathematical functions and avoids data changing-state.

- Examples are Haskell, Lisp, Scheme etc

Procedural Programming

- Procedural programming (PP) is a style of programming in which instructions are executed one after another in sequence
- In PP the main emphasis on solving a problem is on the **procedure**
- The program can be divided into functions (routines, subroutines; functions; not to be confused with mathematical functions)
- The design approach is "Top to Down"
- Data here is globally available to all procedure/functions
- This makes PP less secure
- Lets see what a PP PL code will look like

A simple program that Adds Tow Numbers

Q Basic

```
INPUT , number1
INPUT, number2
sumTotal=number1+number2
PRINT "The sum total is: "; sumTotal
```

A simple program that Adds Tow Numbers

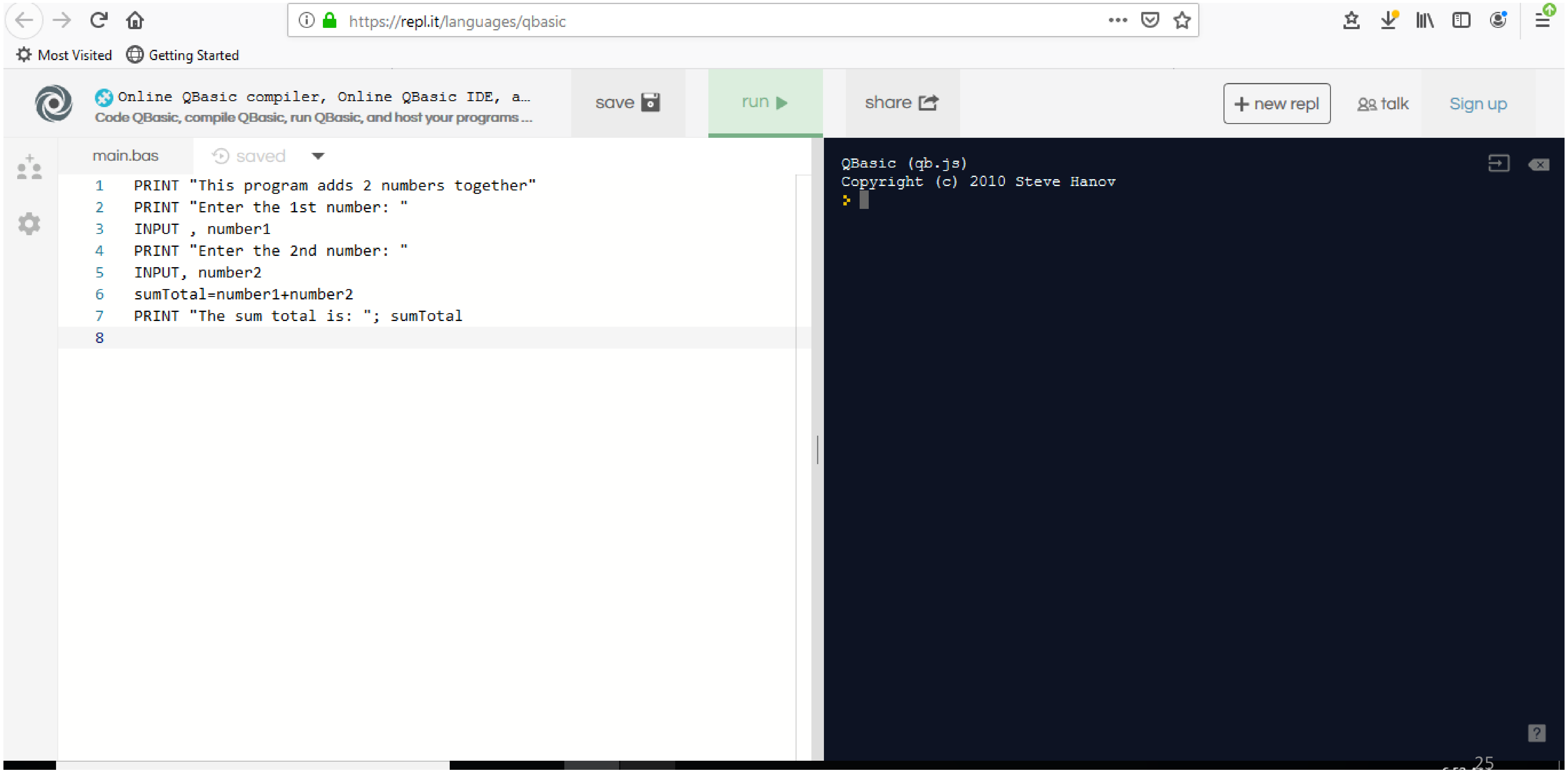
Q Basic

```
PRINT "This program adds 2 numbers together"
PRINT "Enter the 1st number: "
INPUT , number1
PRINT "Enter the 2nd number: "
INPUT, number2
sumTotal=number1+number2
PRINT "The sum total is: "; sumTotal
```

C

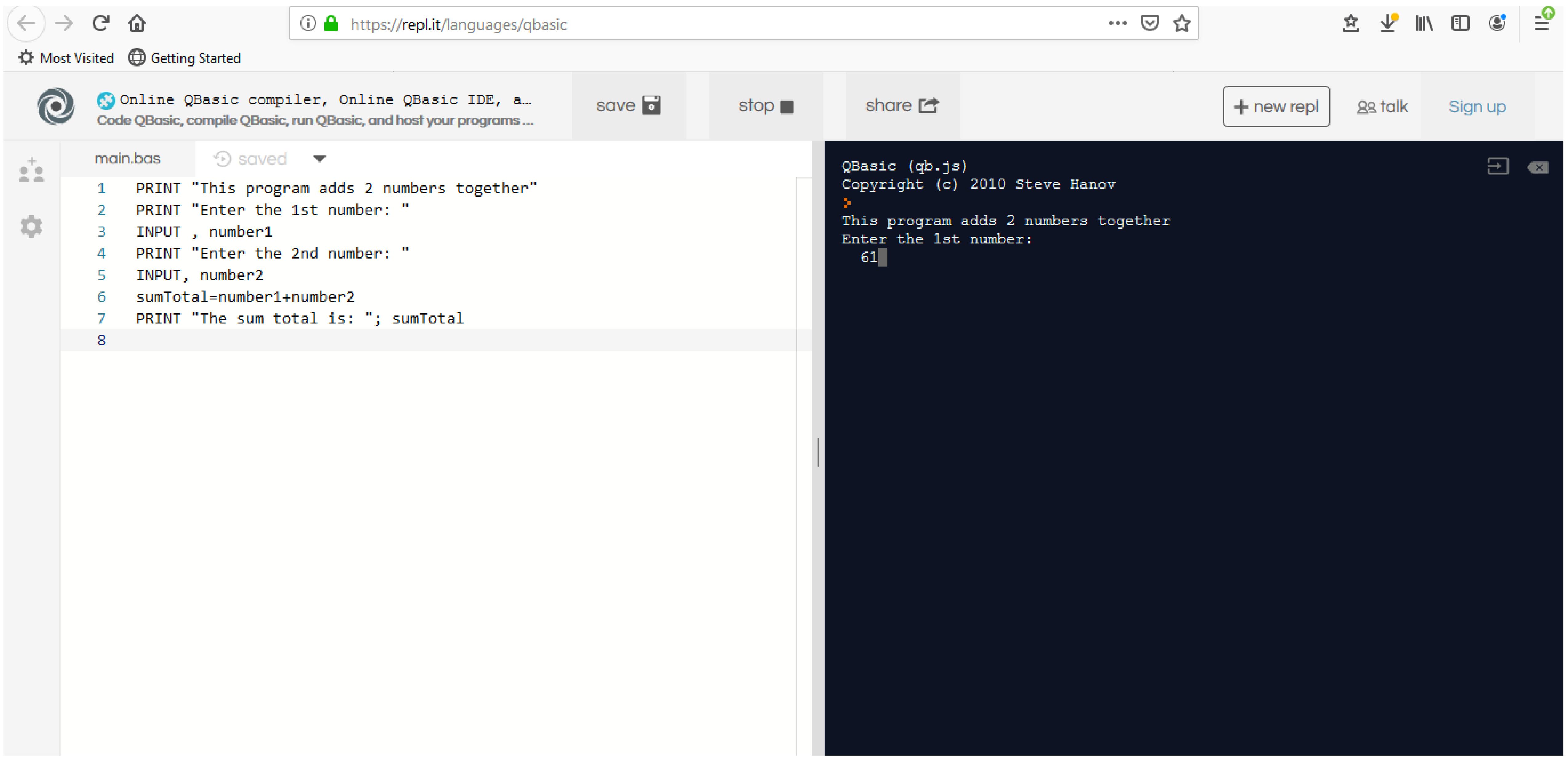
```
#include<stdio.h>
      int main() {
            int a, b, sum;

            printf("Enter first number:");
            scanf("%d",&a);

            printf("Enter second number:");
            scanf("%d",&b);

            Sum = a + b;
            printf("\nSum=%d", sum);
    return 0;
}
```

http://repl.it/languages/qbasic


https://repl.it/languages/qbasic
Most Visited
Getting Started
Online QBasic compiler, Online QBasic IDE, a…
Code QBasic, compile QBasic, run QBasic, and host your programs …
save
run
share
new repl
talk
Sign up
main.bas
saved

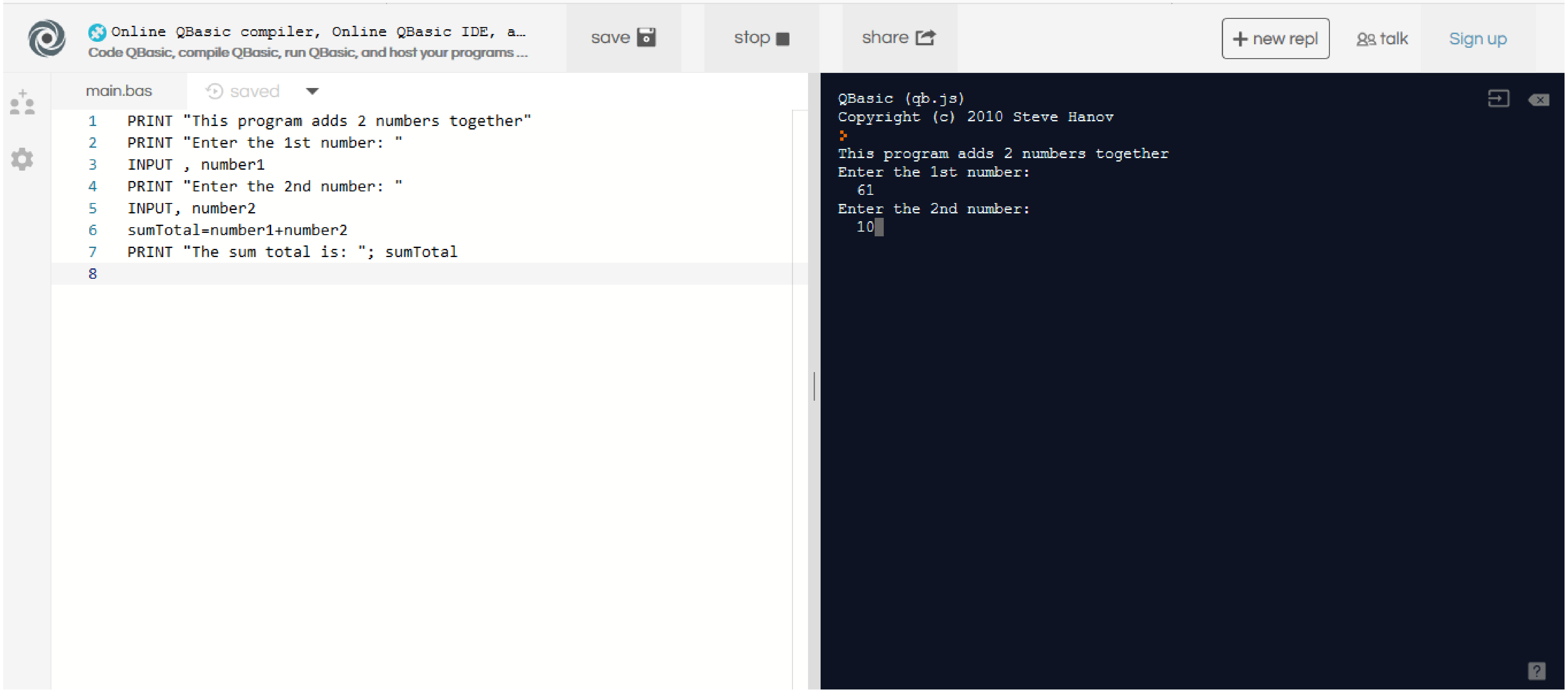
```
PRINT "This program adds 2 numbers together"
PRINT "Enter the 1st number: "
INPUT , number1
PRINT "Enter the 2nd number: "
INPUT, number2
sumTotal=number1+number2
PRINT "The sum total is: "; sumTotal
```

```
QBasic (qb.js)
Copyright (c) 2010 Steve Hanov
>
```


https://repl.it/languages/qbasic
Most Visited
Getting Started
Online QBasic compiler, Online QBasic IDE, a…
Code QBasic, compile QBasic, run QBasic, and host your programs …
save
stop
share
new repl
talk
Sign up
main.bas
saved

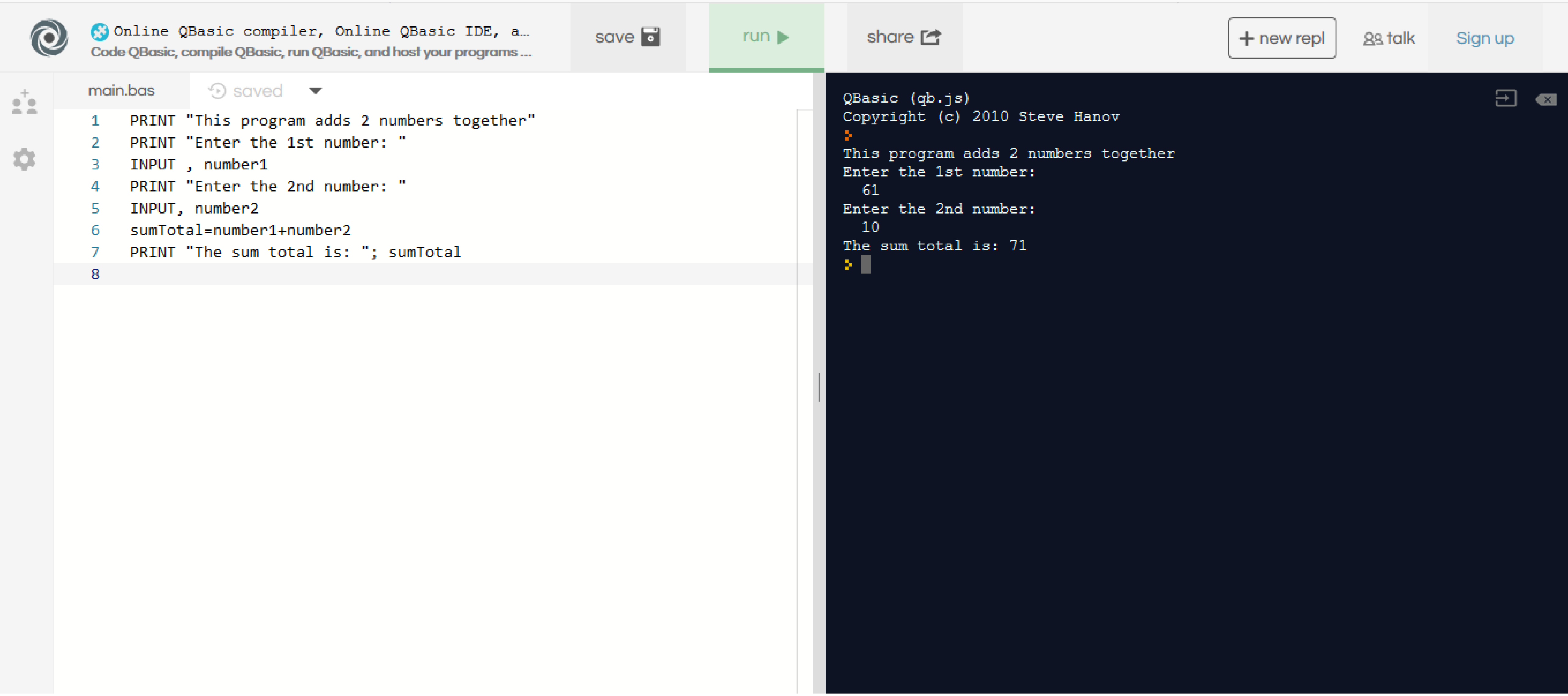
```
PRINT "This program adds 2 numbers together"
PRINT "Enter the 1st number: "
INPUT , number1
PRINT "Enter the 2nd number: "
INPUT, number2
sumTotal=number1+number2
PRINT "The sum total is: "; sumTotal
```

```
QBasic (qb.js)
Copyright (c) 2010 Steve Hanov
>
This program adds 2 numbers together
Enter the 1st number:
  61
```


Online QBasic compiler, Online QBasic IDE, a…
Code QBasic, compile QBasic, run QBasic, and host your programs ...
save
stop
share
+ new repl
talk
Sign up
main.bas
saved

```
PRINT "This program adds 2 numbers together"
PRINT "Enter the 1st number: "
INPUT , number1
PRINT "Enter the 2nd number: "
INPUT, number2
sumTotal=number1+number2
PRINT "The sum total is: "; sumTotal
```

```
QBasic (qb.js)
Copyright (c) 2010 Steve Hanov
>
This program adds 2 numbers together
Enter the 1st number:
  61
Enter the 2nd number:
  10
```


Online QBasic compiler, Online QBasic IDE, a…
Code QBasic, compile QBasic, run QBasic, and host your programs …
save
run
share
+ new repl
talk
Sign up
main.bas
saved

```
PRINT "This program adds 2 numbers together"
PRINT "Enter the 1st number: "
INPUT , number1
PRINT "Enter the 2nd number: "
INPUT, number2
sumTotal=number1+number2
PRINT "The sum total is: "; sumTotal
```

```
QBasic (qb.js)
Copyright (c) 2010 Steve Hanov
>
This program adds 2 numbers together
Enter the 1st number:
  61
Enter the 2nd number:
  10
The sum total is: 71
>
```

- In procedural applications, you create names for computer memory locations that can hold values—for example, numbers and text—in electronic form
- The named computer memory locations are called variables because they hold values that might vary
- Some may remain constant
- A procedural program defines the variable memory locations and then calls a series of procedures to input, manipulate, and output the values stored in those locations

OOP: Object-Oriented Programming

- **OOP** stands for Object Oriented Programming

- **OOP** evolved to ease the solving of complex

- OOP software development is a paradigm that started in the 1980s to help the development of programs in ways that further

 Reduce production cost
 Develop reusable software modules
 Reduce maintenance cost
 Quicken the completion time for software development

-**Simula** (developed by Kristen Nygaard and Ole-Johan) was the first OOP language

-Other are Java, Python, C++, Visual Basic .NET and Ruby

- OOP is an extension of procedural programming
- It takes a slightly different approach to writing computer programs
- Data here is not globally available
- This makes OOP more secure
- The design approach is "Bottom up"
- In OOP the main emphasis on solving a problem is **on the Data**
- It involves creating **classes**, which are **blueprints** for **objects**; creating **objects** from those **classes**; and creating applications that use those **objects**
- After classes are created, they can be reused repeatedly to develop new programs

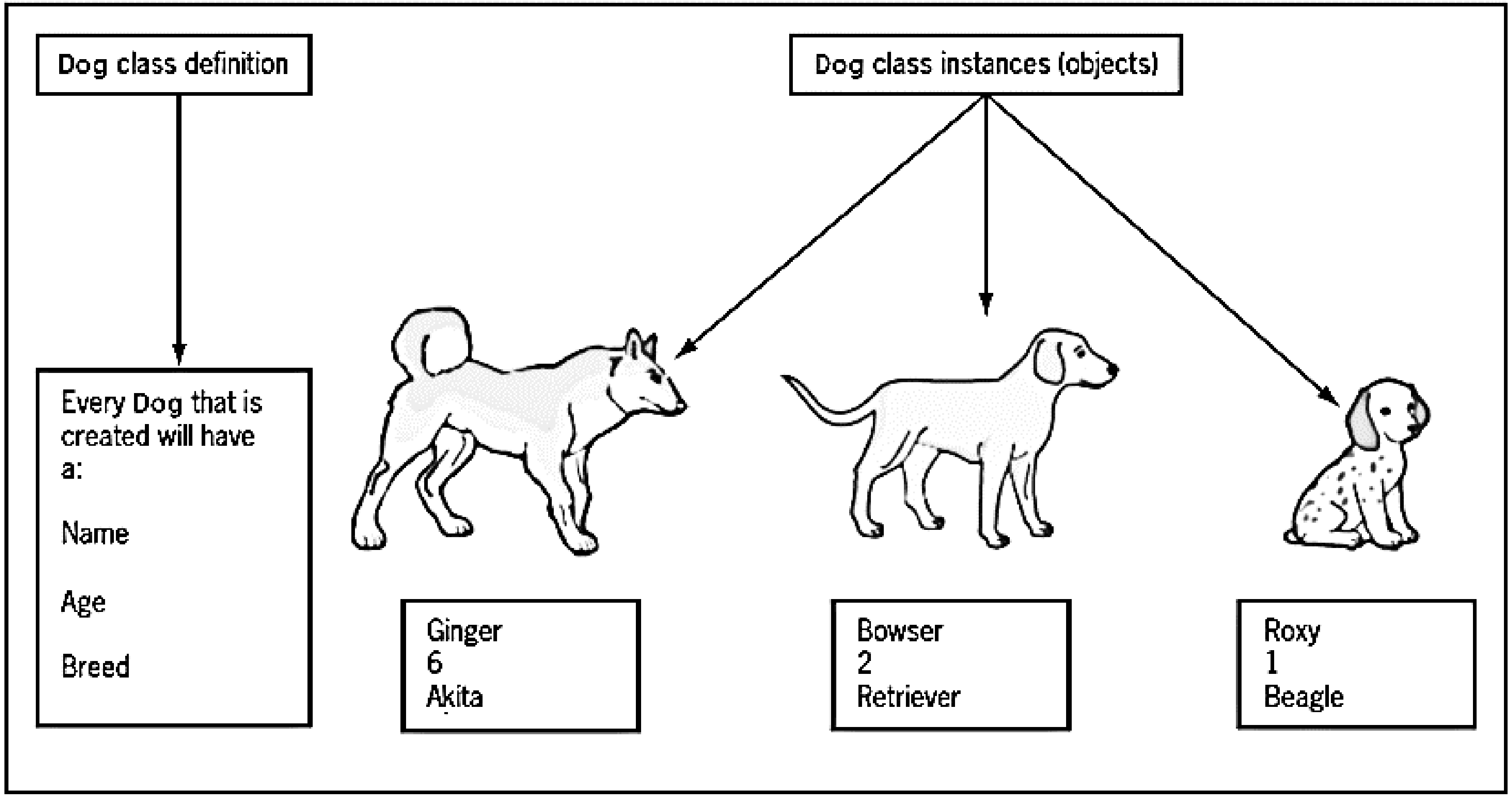
Originally, object-oriented programming was used most frequently for two major types of applications:

1 **Computer simulations**, which attempt to mimic real-world activities so that their processes can be improved or so that users can better understand how the real-world processes operate

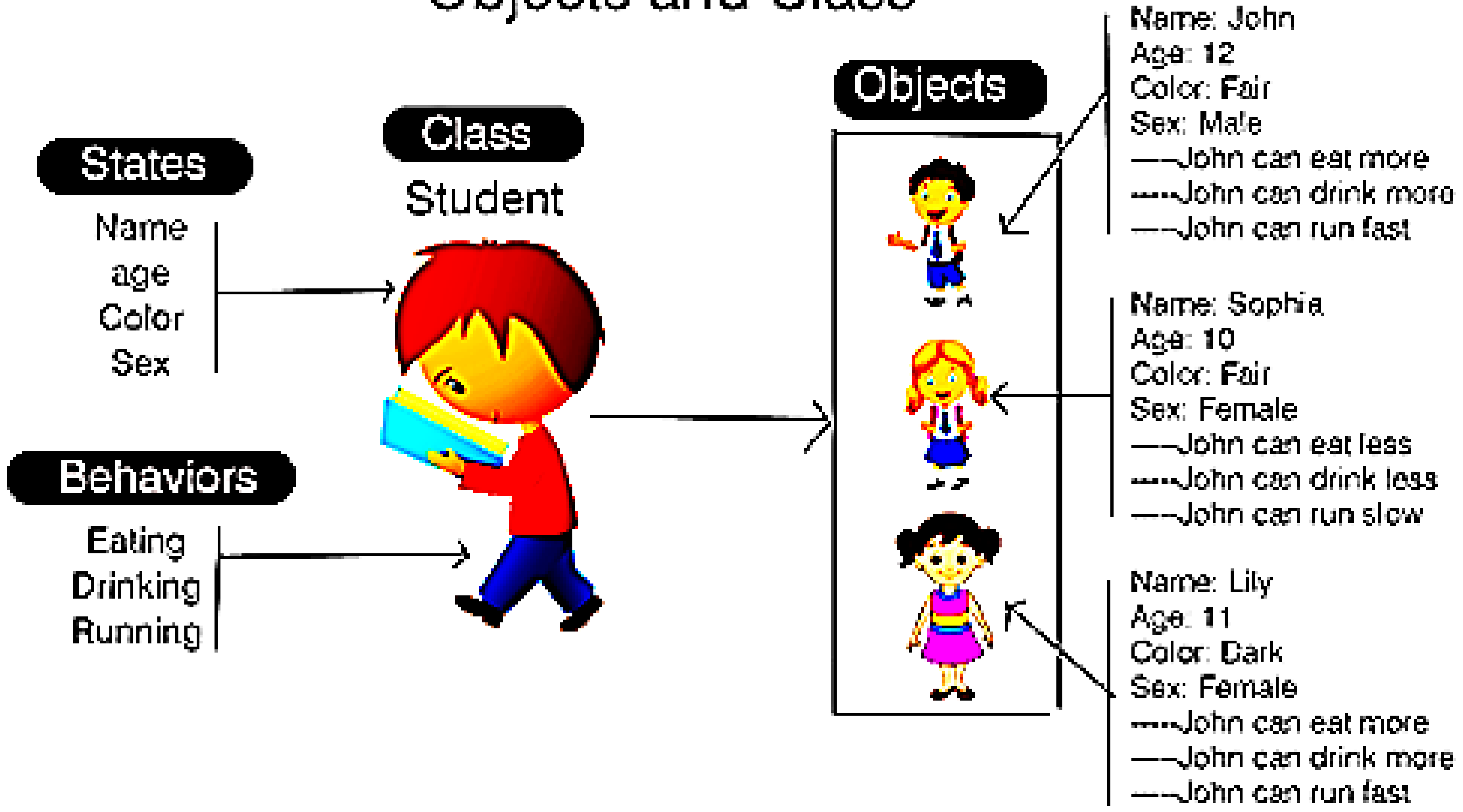
2 **Graphical user interfaces,** or GUIs (pronounced "gooeys"), which allow users to interact with a program in a graphical environment

- A class in Java can either be created by the programmer or the by the language creator

- An object is a specific, concrete instance of a class



Objects and Class



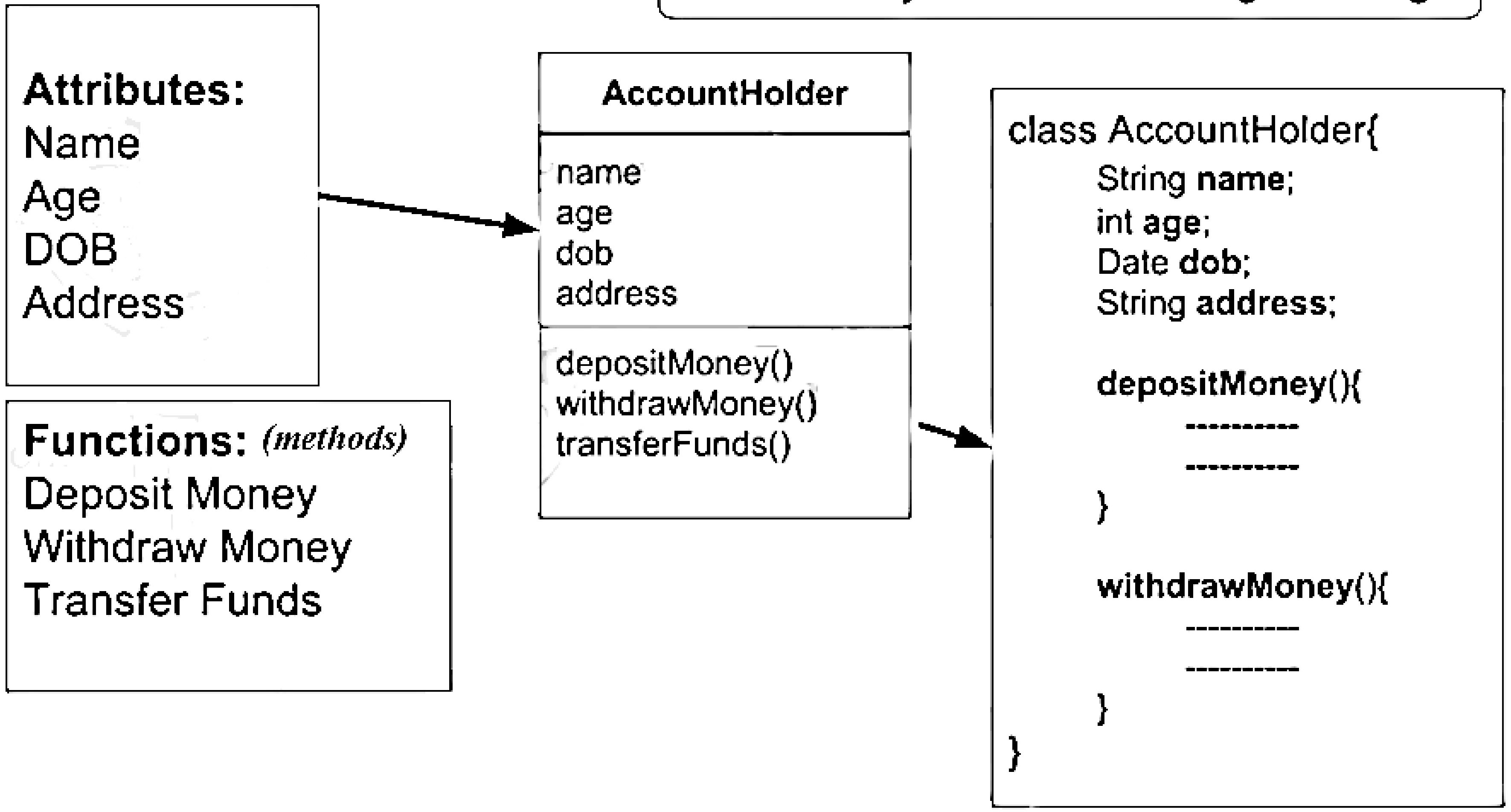
Understanding Classes, Objects, and Methods

OOP is a method of programming where code is designed and based on <u>functions</u> and <u>attributes</u> of the <u>object</u>

What are functions and attributes of an object?

- In OOP, a object is an instance of an class
- A class could be seen as a blueprint (of objects)
- It exist before any objects are created from
- Classes describes the attributes its objects will posses and what those objects will be able to do.
- Attributes are the characteristics that define an object; they are properties of the object

OOP - Object Oriented Programming



```
class AccountHolder{
        String name;
        int age;
        Date dob;
        String address;

        depositMoney(){
                ----------
                ----------
        }

        withdrawMoney(){
                ----------
                ----------
        }
}
```

Class Automobile would describes what Automobile objects
With attributes such as make, model, year, and colour etc

Each Automobile object would possesses the same attributes but not, the same values

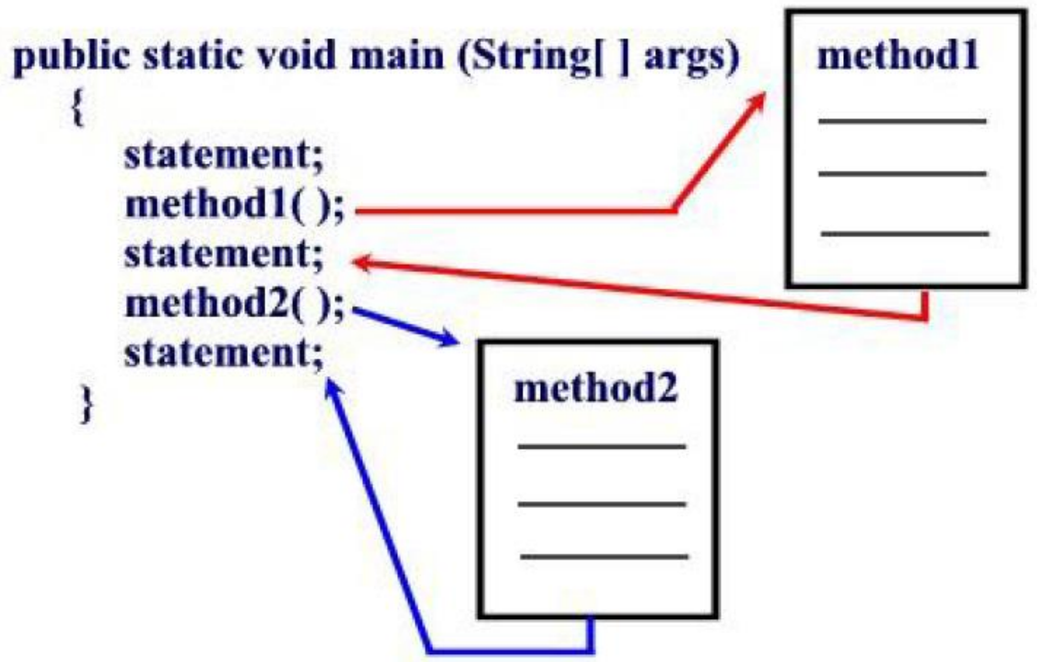
-The values **(Data)** of the properties of an object could also referred to as the **object's state**, EG of classes are below:

```
Class  Automobile {        Class  SmartPhone{          Class  box{
       make;                      make;                       height;
       Model;                     screenSize;                 weight;
       year;                      memory;                     breadth;
       Colour;                    input;                      input;
}                          }                           }
```

```
Class  Automobile {        Class  SmartPhone{
       String make;               String make;
       String Model;              Double screenSize;
       Int year;                  Double memory;
       String Colour;             String input;
}                          }
```

Methods

- A method is a set of instructions that describe a functions that can be performed on a object. It can be called (invoked) at any point in a program
- Think of a method as a subprogram that acts on data



There are two basic types of methods:

Built-in: Build-in methods are part of the compiler package, such as System.out.println( ) and System.exit(0).

User-defined: these are created the programmer. These
methods take-on names that you assign to them and perform tasks that you create

- All java programs are written in a class
- All classes must have a main method
- Every line of instruction expected to be executed must be contained in the main method
- Structurally any group of codes/computer instructions must lie within { and }
- every statement within the main method must also end with a ;

A JAVA program that Adds Tow Numbers

1

```
class AddNumbers {
        public static void main(String args[]) {
                int x, y, z;
                System.out.println("Enter two integers to calculate their sum ");
                x = 10;
                y = 190;
                z = x + y;
                System.out.println("Sum of entered integers = "+z);
        }
}
```

2

```
import java.util.Scanner;
class AddNumbers {
        public static void main(String args[]) {
                int x, y, z;
                System.out.println("Enter two integers to calculate their sum ");
                Scanner in = new Scanner(System.in);
                x = in.nextInt();
                y = in.nextInt();
                z = x + y;
                System.out.println("Sum of entered integers = "+z);
        }
}
```

- The first java program takes constants as input, in other words the input are already predefined

- This give you an idea on how to use constants

- The second java program takes variables as input, in other words the input are to be provide by who so ever runs the program at that point

- This give you an idea on how to use “scan” accept variables in Java

ANOTHER PERSPECTIVE TO CLASS, OBJECTS, ATTRIBUTES AND METHODS

Given the table Dog Register:

- A is class, blue print (class Dog register)
- B is an Object (object Lion)
- C is to be calculated from Reg date and is called a METHOD (sub function, sub routine)

C

A →	**ID**	**Name**	**Age**	**Breed**	**Reg. date**	**Vac. Date**	**…**
	001	Bingo	5	Akita	1/2/2018		
B →	002	Lion	2	Retriever	5/3/2018		
	003	Bullet	3	Beagle	9/6/2018		

Other Java programs:

```
//This program calculates the Area of a Circle
//Area =(22*r*r)/7 ;
class AreaOfCircle {
  public static void main(String args[]) {
             Scanner s= new Scanner(System.in);
               System.out.println("Enter the radius:");
              double r= s.nextDouble();
              double  area=(22*r*r)/7 ;
              System.out.println("Area of Circle is: " + area);
  }
}
```

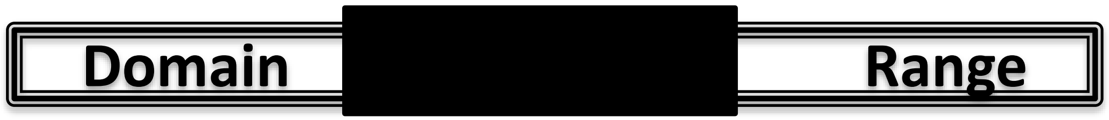
```
//This program calculates the Area of a Triangle
//Area = (width*height)/2
class AreaOfTriangle {
  public static void main(String args[]) {
        Scanner s= new Scanner(System.in);
       System.out.println("Enter the width of the Triangle:");
       double b= s.nextDouble();
       System.out.println("Enter the height of the Triangle:");
       double h= s.nextDouble();
       double area=(b*h)/2;
     System.out.println("Area of Triangle is: " + area);
  }
}
```

```
//This program sums up any number of integer
class sum{
  public static void main(String arg[]) {
              int n,sum=0;
              Scanner sc=new Scanner(System.in);
              System.out.println("enter how many numbers you want sum");
              n=sc.nextInt();
              int a[]=new int[n];
              System.out.println("enter the "+n+" numbers ");
              for(int i=0;i<n;i++){
                          System.out.println("enter  number  "+(i+1)+":");
                          a[i]=sc.nextInt();
             }
             for(int i=0;i<n;i++){
                          sum+=a[i];
             }
             System.out.println("sum of "+n+" numbers is ="+sum);
          }
}
```

Functional Programming

Involves writing programs in a style that treats computation as the evaluation of mathematical functions

- A FP program is a collection of mathematical functions, each with an input(domain) and a result (range), thus it is **rooted in mathematics**
- Because of the above it is **language independent**
- Functions can be imagined as a rail way track with a tunnel

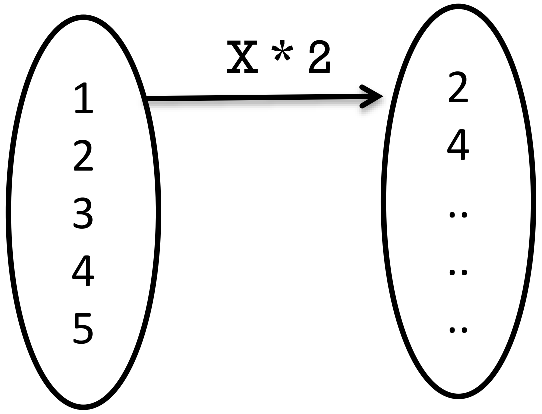


For eg take hot water as a function, it takes in garri and give you………, a blender take in lemons and give you……..

Major FP languages are Lisp, Scheme, Haskell, and ML.

Mathematical Functions *(another angle)*

*Are simply mappings from input to output, for eg f(x) = x*2*

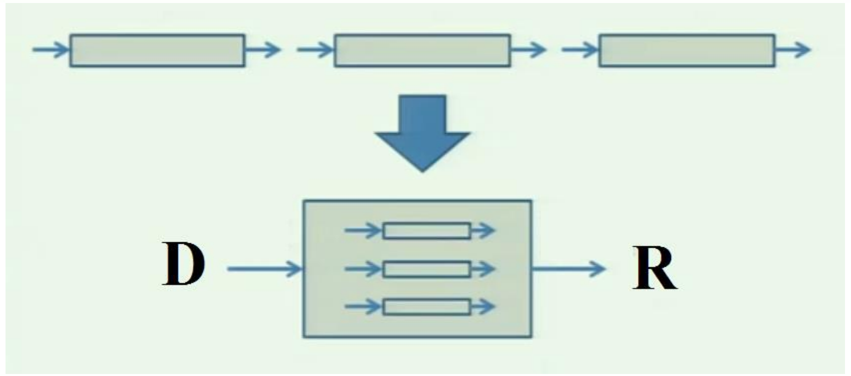


f(x) =x*2 ; p(y) = y + 10
G(f,p) = f(x) + p(y)
g(10,5) = f(10) + f(5)
= 20 + 10
= 30

- Functions in FP Interact and can be combined through a process called functional compositions (like legos) and (conditionals and recursion)
- In FP when a function I/O other functions as input, its called **Higher Order Functions (HOF)**

Functional Compositions (*another angle*)

New function



- FPP avoids changing-state and mutable data. Egs are Haskell, Lisp, Scheme etc

- Functions can’t change their inputs, and should also not be passed data that can change. This make FP **immutable**

- This property of FP makes its not possible for FP to have loops, counters and any other .... that updates variables

- Immutability of FP make it less error prone, easier to debug

- This is achieved by the use of **Pure Functions**, factions that strictly use data passed to them directly and generates output strictly from those inputs

- FP is language independent

Pure:

```
function greet(name) {
  return "Hi, i'm"   + name;
}
```

In-Pure:

```
var name = ""Abu"
function greet() {
  return "Hi, i'm"   + name;
}
```

A Haskell program that Adds Tow Numbers

```
main:: IO()
 main = do putStrLn "Insert the first value: "
       one <- getLine
       putStrLn "Insert the second value: "
       two <- getLine
       putStrLn "The result is:"
   print ((read one) + (read two))
```

A Haskell program that Adds Tow Numbers

```
(defun add()
(format t "Enter 1st Value ")
(setf a(read))
(format t "Enter 2nd Value ")
(setf b(read))
(setf c(+ a b))
(format t "Sum=~d" c)
)
```