LEARN TO CODE · FIRST STEPS

Chapter 1

Meeting BASIC-256 — Say Hello

This chapter introduces the BASIC-256 environment using the `print` and `say` statements. You will see the difference between commands you send to the computer, strings of text, and numbers used by a program. We will also explore simple mathematics to show off just how talented your computer is. Lastly you will learn what a syntax error is and how to fix one.

hello.kbs BASIC-256

```
say "hello"
```

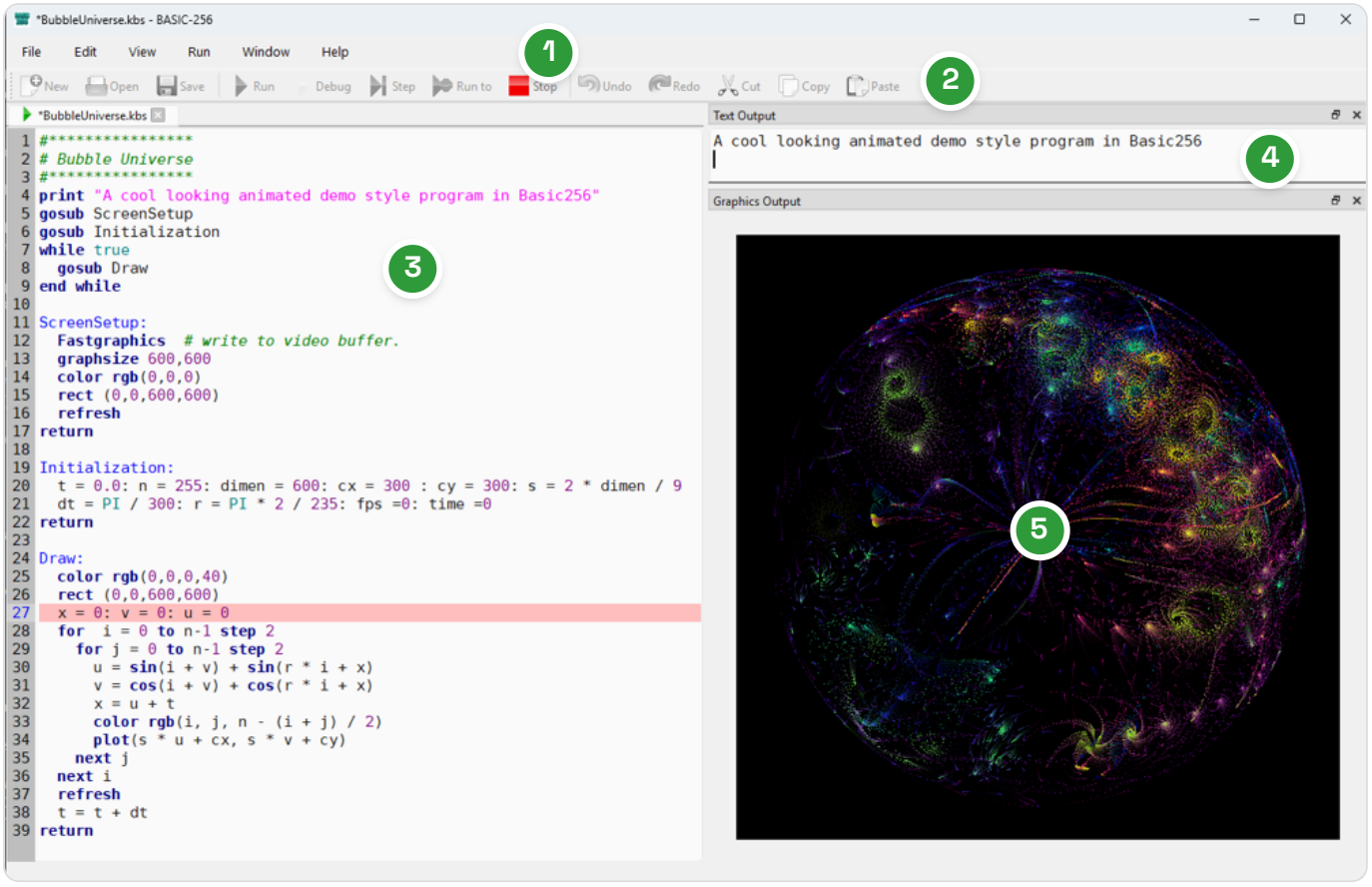


© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 1

The BASIC-256 window

The BASIC-256 window, also known as an **Integrated Development Environment** (IDE), is divided into five parts: the **Menu Bar**, **Tool Bar**, **Program Area**, **Text Output Area** and **Graphics Output Area**.



1 Menu bar 2 Tool bar 3 Program area 4 Text output area 5 Graphics output area

The BASIC-256 window, also called the Integrated Development Environment (IDE).

1 · Menu bar

The menu bar holds six drop-down menus. Click a name to open it.

File	start a new program, open and save programs, close a program's tab, print, and exit BASIC-256.
Edit	cut, copy and paste text and pictures from the program, text output and graphics output areas, and find or replace text.
View	show or hide parts of the window, zoom, and switch between a light and a dark theme.
Run	run your program, or debug it one line at a time.
Window	switch between the programs you have open, each in its own tab.
Help	open the online help and show **About BASIC-256**, with the version you are using.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

2 · Tool bar

The menu options you will use the most are also on the tool bar, one click away.

	**New**	Open a new, empty tab for a new program
	**Open**	Open a saved program in its own tab
	**Save**	Save the program in the current tab to your computer or USB drive
	**Run**	Run the program in the current tab
	Debug	Start running the program one line at a time
	**Step**	While debugging — go to the next line
	**Run to**	While debugging — run to the next line marked as a break point
	**Stop**	Stop the program that is running
	**Undo**	Undo the last change to the program
	**Redo**	Redo the last change that was undone
	**Cut**	Move the highlighted program text to the clipboard
	**Copy**	Put a copy of the highlighted program text on the clipboard
	**Paste**	Insert text from the clipboard where the cursor is

3 · Program area

Programs are made up of instructions that tell the computer exactly what to do and how to do it. You type your programs here, change and fix them, and load saved programs into this area. Each program gets its own **tab** along the top of the area, with its name on it, so you can have several programs open at once.

4 · Text output area

This area shows the words and numbers your program prints. If the program needs to ask you a question, the question (and what you type) appears here too.

5 · Graphics output area

BASIC-256 is a graphical language, as you will see in Chapter 2. The pictures, shapes and graphics you create are drawn here.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 2

Your first program — the say statement

Let's actually write a computer program and see if BASIC-256 will say hello to us. In the Program Area, type the following one-line program. (BASIC-256 shows a line number in front of every line — you don't type it.)

sayhello.kbs — BASIC-256

```
say "hello"
```



Program 1 · Say Hello

Once you have typed it in, click **Run** on the tool bar. Did BASIC-256 say hello to you through the computer's speakers?



NEW CONCEPT

Run on the tool bar — or **Run ▸ Run** on the menu, or press **F5**.

You must tell BASIC-256 when you want it to start running a program. It doesn't know when you have finished typing, so nothing happens until you click Run.



NEW CONCEPT

```
say expression
```

The `say` statement makes BASIC-256 read an expression aloud, through the computer's speakers.

BASIC-256 treats letters, numbers and punctuation that are inside a pair of quotation marks as one block. This block is called a **string**.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout



NEW CONCEPT

```
"letters, numbers 9988, and symbols &%"
'another string with a "quote" inside.'
```

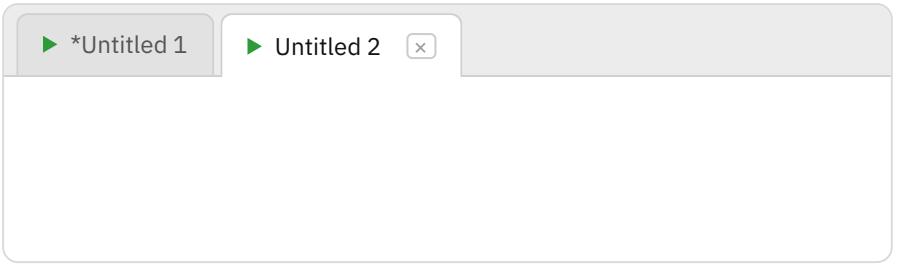
A string may begin with either a single quote mark (`'`) or a double quote mark (`"`), and it ends with the same mark it began with. So a string in single quotes may contain double quotes, and a string in double quotes may contain single quotes.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 3

Starting a new program

When you want to start another program, click the **New** button on the tool bar. BASIC-256 does **not** throw away the program you were working on. Instead it opens a fresh, empty **tab** next to it, called *Untitled 2* (then *Untitled 3*, and so on). Your first program is still there, safe in its own tab — just click its tab to go back to it.



After clicking New: the first program keeps its tab, and the new, empty program opens in a tab of its own.

See the little ***** in front of **Untitled 1*? It means that program has changes that have **not been saved** yet. When you save it, the * disappears.

What can I do with my first program?

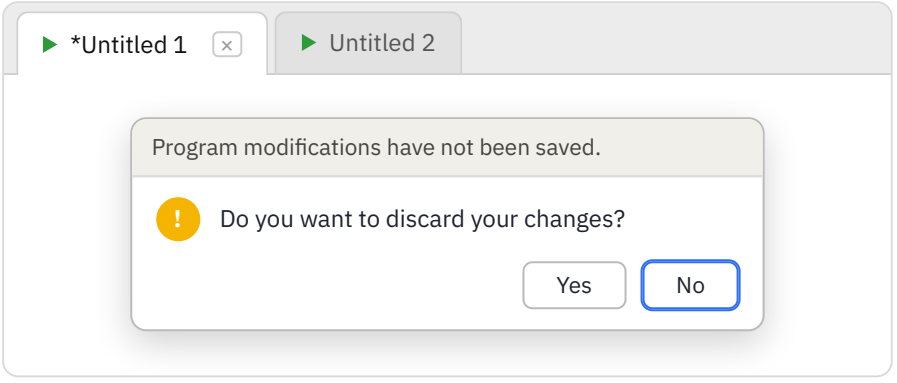
Click its tab first, so that it is the program you see. Then you can choose:

Keep it — save it

1. Click the **Save** button.
2. Type a name for your program, for example *hello*, and click **Save**.
3. The tab now shows the new name, without a *. You can close the tab and open the program again later — Chapter 2 tells you more about saving.

Throw it away — close it

1. Click the small **×** on its tab, or use **File ▸ Close**.
2. If the program has unsaved changes (the *), BASIC-256 asks first, so you can't lose your work by accident.
3. Click **Yes** to throw the changes away and close the tab, or **No** to keep it open.



Closing a tab with unsaved changes. **No** is the safe choice: the tab stays open so you can save it first.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout



NEW CONCEPT

New on the tool bar — or **File ▸ New** on the menu.

New opens a new, empty tab for a new program. The programs you already have open stay just as they are. To get rid of a program, close its tab; to keep it, save it. You can also simply leave it open: when you exit BASIC-256 it reminds you about every program that has not been saved and lets you **Save** it, **Discard** it, or **Cancel** and go back.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 4

Your second program — saying something else

The `say` statement can also speak numbers. Click **New** to get a fresh tab, and try this program:

saynumber.kbs BASIC-256

```
say 123456789
```



Program 2 · Say a Number

Click **Run**. Did BASIC-256 say what you were expecting?



NEW CONCEPT

numbers

BASIC-256 lets you enter numbers in decimal format. Do not use commas when you type large numbers. If you need a number less than zero, just put a minus sign in front of it.

Examples: `1.56` , `23456` , `-6.45` and `.5`

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 5

BASIC-256 is really good with numbers — simple arithmetic

The brain of the computer, called the **Central Processing Unit** (CPU for short), works only with numbers. Everything it does — graphics, sound and all the rest — is done by working with numbers.

The four basic operations of addition, subtraction, multiplication and division are written with the operators in this table.

BASIC MATHEMATICAL OPERATORS

OPERATOR	OPERATION	EXAMPLE
+	Addition	`expression1 + expression2`
-	Subtraction	`expression1 - expression2`
*	Multiplication	`expression1 * expression2`
/	Division	`expression1 / expression2`

Try this program and listen to the talking super calculator.

sayanswer.kbs — BASIC-256

```
say 12 * (2 + 10)
```



Program 3 · Say the Answer

The computer should have said “144” to you.

sayanother.kbs — BASIC-256

```
say 5 / 2
```



Program 4 · Say Another Answer

Did the computer say “2.5”?

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout



NEW CONCEPT

```
+
-
*
/
( )
```

The four basic operations — addition (`+`), subtraction (`-`), multiplication (`*`) and division (`/`) — work with numbers to make calculations. A number is needed on both sides of the operator. You can use parentheses to group operations together.

Examples: `1 + 1` , `5 * 7` , `3.14 * 6 + 2` , `(1 + 2) * 3` and `5 - 5`

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 6

Concatenation

Concatenation joins two strings together to make a longer string. If the strings “abcd” and “xyz” are concatenated, the result is the string “abcdxyz”. This operation is called concatenation, or “cat” for short.

BASIC-256 has three operators that concatenate strings, but they behave differently when both sides are numbers. The `;` operator turns both sides into strings and **always** concatenates. The `+` operator **adds** two numbers, but concatenates if either side is a string. The `&` operator does a “bit-wise and” if both sides are numbers, and otherwise concatenates.

Let’s try it out:

sayhellomary.kbs — BASIC-256

```
say "Hello " ; "Mary."
```



Program 5 · Say Hello to Mary

The computer should have said hello to Mary. Try another one.

onemoretime.kbs — BASIC-256

```
say 1 ; " more time"
```



Program 6 · Say It One More Time

In this last example, concatenation was done with a number and a string. The number was first turned into the string “1”, and then BASIC-256 could concatenate.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

threeways.kbs BASIC-256

```
say 1 + 2
say '1' + 2
say 1 ; 2
```



Try it: the same numbers, three operators

The computer should have said “three”, “twelve” and “twelve”. On the first line, `+` adds the numbers 1 and 2. On the second line, `+` concatenates the string “1” and the number 2 (which is turned into a string). On the last line, `;` turns both numbers into strings and concatenates them.



NEW CONCEPT

```
; (concatenate)
+ (concatenate)
& (concatenate)
```

The semicolon (`;`) tells the computer to concatenate (join) strings together. If one or both sides are numbers, they are turned into strings first.

The `+` and `&` operators concatenate if either or both sides are strings. If both sides are numbers, they do something else.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 7

The text output area — the `print` statement

Programs that use the Text to Speech (TTS) `say` statement are useful and fun, but often you also want to write words and numbers on the screen, so they can be read. The `print` statement does just that. In the Program Area, type this two-line program:

printhello.kbs — BASIC-256

```
print "hello"
print "there"
```

Text Output

```
hello
there
```

Program 7 · Print Hello There

Click **Run**. The text output area should now show “hello” on the first line and “there” on the second line.



NEW CONCEPT

```
print expression
print expression;
```

The `print` statement displays text and numbers in the text output area of the BASIC-256 window.

After printing, `print` normally moves down to the next line. If you put a `;` (semicolon) at the end, it stays on the same line, so the next print continues right after it.

Here is that semicolon in action:

manyprints.kbs — BASIC-256

```
cls
print "Hello ";
print "there, ";
print "my friend."
```

Text Output

```
Hello there, my friend.
```

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

Program 8 · Many Prints One Line



NEW CONCEPT

```
cls
```

The `cls` statement clears everything that was displayed in the text output area.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout

SECTION 8

What is a “syntax error”?

Programmers are human and sometimes make mistakes. A **syntax error** is one kind of mistake: BASIC-256 reports it when it does not understand the program you typed. Syntax errors are usually caused by misspellings, missing commas, wrong spaces, quotes that are never closed, or parentheses that don’t match.

BASIC-256 tells you which line the error is on, and even tries to tell you roughly where on the line. Here the closing parenthesis is missing:

oops.kbs — BASIC-256

```
say 12 * (2 + 10
```

Text Output

```
COMPILE ERROR on line 1: Syntax error around character 17.
```

A syntax error, reported in the text output area



TIP

Don’t worry when you see an error — every programmer gets them all the time. Read the message, look at the line it names, fix the mistake and click **Run** again.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout



PRACTICE

Exercises

WORD SEARCH

z a h d g p b a n n q m c
j g j r o i q l o c q o x
r u n t u u n i l c n s z
v w s y o b s s k c y l l
e n a t i s s p a n p a x
r s e p e q r t t f r p t
r b k r y o e a r m m r a
o r p i g n x d o i f n i
r x n r a y t i h l n a f
e g a t m d w n v e d g i
t m i a c v c e i j f d n
b o t c c a u s o r c i s
n a m z i z i g n c p r u

Find: cls, concatenation, error, expression, print, program, quote, run, say, stop, string, syntax



Problems

Try these — they only use what this chapter covered.

1. Write a one-line program to say the tongue twister “Peter Piper picked a peck of pickled peppers.”
2. Add a second line to Problem 1 to also display that sentence on the screen.
3. Use the computer as a talking calculator to solve this problem and say the answer: Bob has 5 pieces of candy and Jim has 9. If they share the candy evenly, how many would they each have (the average)?
4. Use the computer as a talking calculator to solve this problem and say the answer: you want 5 model cars that cost $1.25 each and one model boat that costs $3.50. How much money do you need to buy them all?
5. Write a one-line program to say “one plus two equals three” without using the word *three* or the number 3.

Chapter 1 · Meeting BASIC-256 — Say Hello. Modernised layout of the BASIC-256 beginner’s guide, updated for the current BASIC-256 window, its tabs and its menus. Original text © 2019 James M. Reneau, licensed under CC BY-NC-SA 3.0 US. This adaptation is shared under the same licence. Mascot: BitBot.

© 2019 James M. Reneau · CC BY-NC-SA 3.0 US · adapted layout