Chapter 6: Your Program Asks for Advice.

This chapter shows how BASIC-256 asks the user to enter strings and numbers, and how to use this in a program.

InputString – Getting Text From the User:

So far we have told the program everything it needs to know in the programming code. The next statement to introduce is *inputstring*. The *inputstring* statement captures a string that the user types into the text area and stores that value in a variable.

Let's take Program 22 and modify it so that it will ask you for a name and then say hello to that person.


```
1     # ilike.kbs
2     # using input to ask for a name
3
4     inputstring "enter your name>", name
5     message1 = name + " is my friend."
6     message2 = "I like " + name + "."
7
8     print message1
9     say message1
10    print message2
11    say message2
```

Program 29: I Like fill in the blank

```
enter your name>Vance
Vance is my friend.
I like Vance.
```

Sample Output 29: I Like fill in the blank


```
inputstring "prompt", variable
inputstring variable
```

The **inputstring** statement will retrieve a string that the user types into the text output area of the screen. The result will be stored in a variable that may be used later in the program.

A prompt message, if specified, will display on the text output area and the cursor will directly follow the prompt.

InputInteger and InputFloat – Getting Numbers

The "Math-wiz" program shows an example of input with numbers.

```
1     # mathwiz.kbs
2     # show several mathematical operations
3
4     inputfloat "a? ", a
5     inputfloat "b? ", b
6
7     print a + "+" + b + "=" + (a+b)
8     print a + "-" + b + "=" + (a-b)
9     print b + "-" + a + "=" + (b-a)
10    print a + "*" + b + "=" + (a*b)
11    print a + "/" + b + "=" + (a/b)
12    print b + "/" + a + "=" + (b/a)
```

Program 30: Math-wiz

```
a? 7.9
b? 6
7.9+6.0=13.9
```

```
7.9-6.0=1.9
6.0-7.9=-1.9
7.9*6.0=47.4
7.9/6.0=1.31666666667
6.0/7.9=0.759493670886
```

Sample Output 30: Math-wiz

.


```
inputinteger "prompt", variable
inputinteger variable
inputfloat "prompt", variable
inputfloat variable
```

The **inputinteger** and **inputfloat** statements will allow a user to enter either an integer or float value and store that into a variable.

If the user enters a value that is not numeric, an error or warning will be displayed. If the "Runtime handling of bad type conversions" in the Preferences is set to either "warn" or "ignore" a zero (0) will be assigned to the variable.

The **inputfloat** statement will allow for a user to enter a number with a thousands separator (1,234,567.89) and will accept the number. The **inputinteger** statement only allows the numbers 0-9 and an optional leading minus sign.

A prompt message, if specified, will display on the text output area and the cursor will directly follow the prompt.

Here is another example using **inputinteger** and **inputstring**.

```
1     # sayname.kbs
2
3     inputstring "What is your name?", name
4     inputinteger "How old are you?", age
5
```

```
6     greeting =  "It is nice to meet you, " + name + "."
7     print greeting
8     say greeting
9
10    greeting =  "In 8 years you will be " + (age + 8) + "
      years old. Wow, that's old!"
11    print greeting
12    say greeting
```

Program 31: Fancy – Say Name

```
What is your name?Jo
How old are you?13
It is nice to meet you, Jo.
In 8 years you will be 21 years old. Wow,
that's old!
```

Sample Output 31: Fancy – Say Name

Input – Automatic Type Conversion

The last style of the input statement we will discuss is the plain *input*. This statement will ask the user for something and automatically convert it to either a string, integer or floating-point value. This may be the behavior you wish but may cause problems in other places

New Concept

```
input "prompt", variable
input variable
```

The **input** statement will allow a user to enter a string, integer, or a floating-point number. After the input is complete, if the entry can be converted to an integer or a floating-point number it will and be stored that way. If the user enters a value that is not numeric, it will be stored as a string.

This automatic type assignment may cause some confusion as spaces, leading zeros, and trailing zeros after a decimal point will be stripped from numbers and they will be stored as integer or float values.

A prompt message, if specified, will display on the text output area and the cursor will directly follow the prompt.

Big Program

This chapter's "Big Program" is a silly story generator. Answer the questions with words and the computer will tell you a story.


```
1     # sillystory.kbs
2
3     print "A Silly Story."
4
5     inputstring "Enter a noun? ", noun1
6     inputstring "Enter a verb? ", verb1
7     inputstring "Enter a room in your house? ", room1
8     inputstring "Enter a verb? ", verb2
9     inputinteger "Enter an integer 2 or larger?", howmany
```

```
10    inputstring "Enter a plural noun? ", noun2
11    inputstring "Enter an adjective? ", adj1
12    inputstring "Enter a verb? ", verb3
13    inputstring "Enter a noun? ", noun3
14    inputstring "Enter Your Name? ", name
15
16    sentence = "A silly story, by " + name + "."
17    print sentence
18    say sentence
19
20    sentence = "One day, not so long ago, I saw a " +
      noun1 + " " + verb1 + " down the stairs."
21    print sentence
22    say sentence
23
24    sentence = "It was going to my " + room1 + " to " +
      verb2 + " " + string(howmany) + " " + noun2
25    print sentence
26    say sentence
27
28    sentence = "The " + noun1 + " became " + adj1 + "
      when I " + verb3 + " a " + noun3 + "."
29    print sentence
30    say sentence
31
32    sentence = "The End."
33    print sentence
34    say sentence
```

Program 32: Big Program - Silly Story Generator

```
A Silly Story.
Enter a noun? car
Enter a verb? drive
Enter a room in your house? bathroom
Enter a verb? walk
Enter an integer 2 or larger?5
Enter a plural noun? cows
```

```
Enter an adjective? big
Enter a verb? lifted
Enter a noun? hippo
Enter Your Name? Mary
A silly story, by Mary.
One day, not so long ago, I saw a car drive
down the stairs.
It was going to my bathroom to walk 5 cows
The car became big when I lifted a hippo.
The End.
```

Sample Output 32: Big Program - Silly Story Generator

Exercises:


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

float, input, inputfloat, inputstring, integer, inutinteger, prompt, string

1. Write a program to ask for three names. Store them in string variables. Once the user enters the third name have the computer recite the classic playground song using the names:

```
[Name One] and [Name Two]
sitting in a tree,
K-I-S-S-I-N-G.
First comes love,
then comes marriage,
then comes [Name Three]
in a baby carriage!
```

2. Write a program to ask for an adjective, noun, animal, and a

	sound. Once the use enters the last one, build a single string variable (using concatenation) to say a verse of Old MacDonald. Print the result out with a single statement and say it with a single statement. (Adapted from The Old Macdonald Mad Lib from http://www.madglibs.com) `[Adjective] MacDonald had a [Noun], E-I-E-I-O and on that [Noun] he had an animal, E-I-E-I-O with a [Sound] [Sound] here and a [Sound] [Sound] there, here a [Sound], there a [Sound], everywhere a [Sound] [Sound], [Adjective] MacDonald had a [Noun], E-I-E-I-O.`