

Gail Kahn



PILOT YOUR OWN COMPUTER:



AN INTRODUCTION TO COMPUTER PROGRAMMING IN PILOT



by Rita Liff and Keith Vann
with Treacy Hickok



Graphics: Wendy Warren and Fernando Micheli
Typing: Liz Elliott

FOREWORD

PILOT is the name of a dialogue-oriented programming language for use on a time-sharing computer system. Students and teachers use PILOT to create interactive computer conversations.

PILOT is easier to learn than many other popular computer languages (BASIC, FORTRAN, etc.) because it is not algebraic in nature. It has a less complex structure and can be learned quickly, freeing students with limited mathematical background from the frustrations of trying to begin programming by learning an algebraic language.

Students can interact with programs designed by their teachers, and also learn to write their own. Writing a program requires systematically translating thoughts and ideas into an organized set of instructions for a computer to follow. Within this framework there is plenty of room for creativity. PILOT programming provides students with an opportunity to develop logical thinking skills and imagination simultaneously.

PILOT was first developed by John Starkweather of University of California Medical Center in San Francisco, and many other versions of PILOT have since been created. In 1973 a standard core PILOT language was constructed from similar languages in use at Lawrence Hall of Science, Stanford Research Institute, the University of California Medical Center in San Francisco, the Peoples' Computer Company, Pacific Union College, and others in the area. Implementations of PILOT now exist on several different computers.

In this manual the Lawrence Hall of Science version of PILOT is described. This version is similar to the core PILOT 73' and also includes many extended capabilities.

The authors hope that - -

- For those of you who are new to the use of computers, this manual will provide an exciting and challenging introduction to computer programming.
- For the already-initiated PILOT user, the exercises, examples and explanations within these pages will help to extend your ability to use this language.

use this language.

For the already-familiar PILOT user, the selected exercises and explanations within these pages will help to extend your ability to

write

— for those of you who are new to the use of computers, this manual will

provide an exciting and challenging introduction to computer programs.

For anyone who has

learned programming languages, this manual is similar to the core PILOT 3, and also includes many new

features and exercises. This manual is designed to describe the

operation of PILOT for users of several different computers: Apple II, Commodore 64, and the IBM PC compatible systems. The manual is written for users of the University of California, San Diego, version of PILOT.

The manual was developed by John Stephenson of the University of California, San Diego, and modified by the PILOT Development Team at the University of California, San Diego. The manual is written for users of the PILOT Development Team's version of PILOT.

Within this framework, there is a variety of ways to use the manual. You can use it as a reference for specific topics, or you can use it as a guide to learning the language. The manual is written for users of the PILOT Development Team's version of PILOT. The manual is written for users of the PILOT Development Team's version of PILOT.

PILOT is the name of a dialog-oriented programming language developed by the PILOT Development Team. The manual is written for users of the PILOT Development Team's version of PILOT. The manual is written for users of the PILOT Development Team's version of PILOT.

FOREWORD

BEFORE YOU BEGIN

1. You'll need a terminal hooked up to a computer that understands PILOT.
2. You should know a little about using the terminal. Find out:
 - HOW TO LOG-IN.
 - WHERE THE "RETURN" BUTTON IS AND WHEN TO USE IT.
 - HOW TO ERASE MISTAKES.
 - WHAT ARE ERROR MESSAGES?
 - HOW TO LOG-OFF.
3. Be sure to read the next page - HOW TO USE THIS MANUAL.

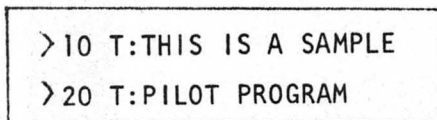
BEFORE YOU BEGIN

1. You'll need a terminal hooked up to a computer that understands ASCII.
2. You should know a little about using the terminal. Find out:
 - HOW TO LOG-IN.
 - WHERE THE "RETURN" BUTTON IS AND WHEN TO USE IT.
 - HOW TO ERASE MISTAKES.
 - WHAT ARE ERROR MESSAGES?
 - HOW TO LOG-OFF.
3. Be sure to read the next page - HOW TO USE THIS MANUAL.

HOW TO USE THIS MANUAL

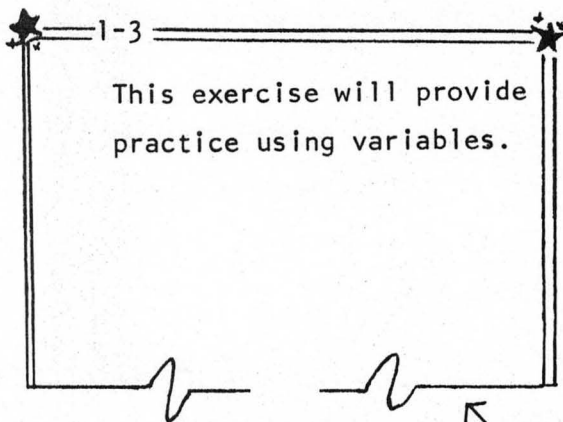
This is a step-by-step guide to programming in PILOT, including exercises to be done at a computer terminal.

WHAT THE SYMBOLS MEAN



```
>10 T:THIS IS A SAMPLE
>20 T:PILOT PROGRAM
```

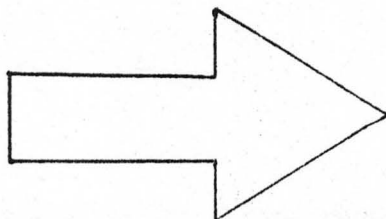
← Boxes surround material shown as it appears when typed on the terminal.



1-3
This exercise will provide
practice using variables.

← Exercises TO BE DONE AT A TERMINAL are surrounded by double-edged boxes.

← This symbol means an exercise is continued on more than one page.

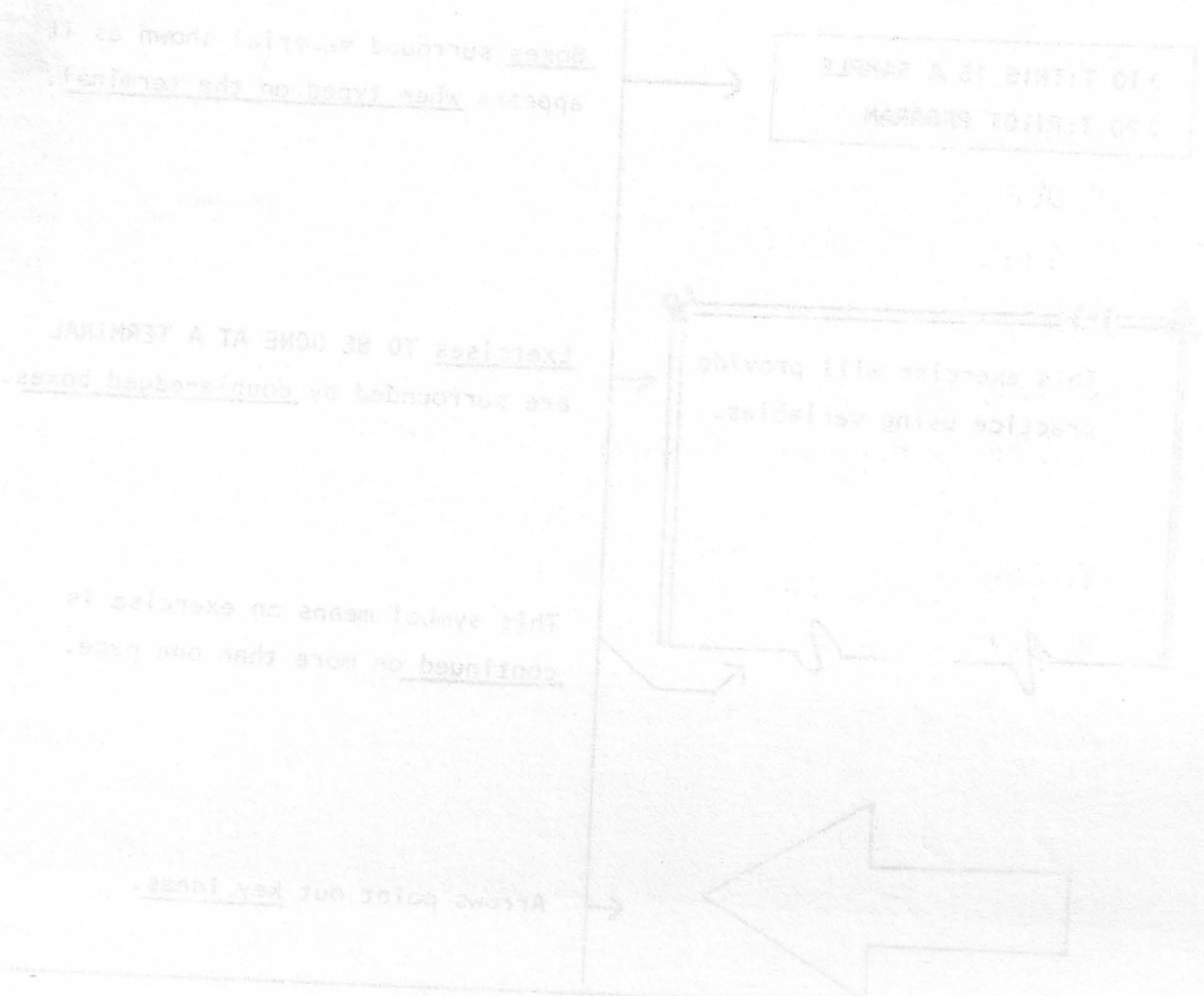


← Arrows point out key ideas.

HOW TO USE THIS MANUAL

This is a step-by-step guide to programming in PL/I, including exercises to be done at a computer terminal.

WHAT THE SYMBOLS MEAN



CONTENTS

| | Page |
|--|------|
| <u>CHAPTER ONE</u> | |
| How to Prepare the Computer for a PILOT Program | 2 |
| A PILOT Program | 3 |
| If You Make a Mistake, You Can Correct It! | 5 |
| Commands | 6 |
| RUN Command | 7 |
| LIS Command | 10 |
| DEL Command | 13 |
| END Command | 14 |
| REN Command | 14 |
| Editing | 15 |
| Creating Pictures with PILOT | 17 |
| Review of Chapter One | 20 |
| <u>CHAPTER TWO</u> | |
| Interactive Programs | 23 |
| TYPE Statement <input type="text" value="T:"/> | 25 |
| ANSWER Statement <input type="text" value="A:"/> | 26 |
| Input and Output | 28 |
| A Special Character "+" | 30 |
| Variables | 31 |
| Variable Names | 33 |
| ANSWER Statement <input type="text" value="A: "/> with Variables | 33 |
| The Value of a Variable | 34 |
| TYPE Statement <input type="text" value="T: "/> with Variables | 37 |
| Mystery Programs with Variables | 39 |

CONTENTS

CHAPTER ONE

How to Prepare the Computer for a PILOT Program 2

A PILOT Program 3

If You Make a Mistake, You Can Correct It 3

Commands 5

REN Command 7

RES Command 8

ALL Command 9

END Command 10

REN Command 11

Editing 12

Creating Pictures with PILOT 14

Review of Chapter One 20

CHAPTER TWO

Interactive Programs 23

TYPE Statement [T] 25

ANSWER Statement [A] 26

Input and Output 28

A Special Character "+" 30

Variables 31

Variable Names 33

ANSWER Statement [A] with Variables 34

The Value of a Variable 34

TYPE Statement [T] with Variables 37

Mystery Programs with Variables 39

CHAPTER ONE

| | |
|---|----|
| How to Prepare the Computer for a PILOT Program | 2 |
| A PILOT Program | 3 |
| If You Make a Mistake, You Can Correct It! | 5 |
| Commands | 6 |
| RUN Command | 7 |
| LIS Command | 10 |
| DEL Command | 13 |
| END Command | 14 |
| REN Command | 14 |
| Editing | 15 |
| Creating Pictures with PILOT | 17 |
| Review of Chapter One | 20 |

Version of Script One

Start/stop symbols with B/G/D

Editing

BEI Command

END Command

DEF Command

FIZ Command

ВЛИ Command

Command

If you work in "mode" you can collect it

A B/G/D Program

How to create the Command for a B/G/D Program

CHAPTER ONE

50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35

HOW TO PREPARE THE COMPUTER FOR A PILOT PROGRAM

(Entry into PILOT)

When you are at a terminal you will follow the steps below to prepare the computer for a PILOT program.

1. Check to see that your terminal has been logged-in.
2. Let the computer know you are going to write a program in PILOT.
To do this, type:

```
XEQ-$PILOT
```

3. The computer will then type:

```
PILOT PROGRAM NAME?
```

IF IT
DOESN'T, TRY
STEP 2 AGAIN



4. Now make up a name for your program which is 5 or less characters (letters or numbers). Type the name.

5. The computer will print :

```
>
```

Then it is ready for you to type your program.

Example of Entry into PILOT:

```
XEQ-$PILOT  
PILOT PROGRAM NAME? PETER  
>
```

MY PROGRAM'S
NAME IS PETER



1-1

Follow the steps to prepare the computer for a PILOT program. When you are finished, the computer should type:

```
>
```

IF NOT, TYPE THE PROGRAM NAME AGAIN

On the same line as ">", type "END", and press "RETURN".
This should happen:

```
>END
```

```
*READY
```

THE *READY MEANS YOU ARE DONE WITH
THIS EXERCISE. IT IS NOT A QUESTION.
PLEASE DON'T ANSWER IT.



What PILOT program name did you select?

Food

A PILOT PROGRAM

What is a program?



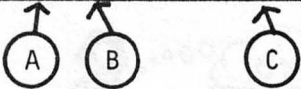
A program is a list of numbered statements. Every statement contains an instruction for the computer to follow.

Let's look at a sample PILOT program:

```
>10 T:WELCOME TO PILOT
>20 T:WHAT'S YOUR NAME
>30 A:$NAME
>40 T:HELLO THERE $NAME
>50 T:HOW DO PEOPLE TEACH COMPUTERS
>60 A:
>70 M:PROGRAM,WRITE A PROGRAM
>80 T Y:YOU KNOW IT!
```

Now look at one of the statements in the program . . .

```
>10 T:WELCOME TO PILOT
```



There are 3 parts to a PILOT statement.

- (A) LINE NUMBER - any number between 1 and 9999
- (B) INSTRUCTION - a single letter followed by a colon (:)
- (C) OPERAND - the information that the PILOT instruction works with . . . not all instructions need operands

Look a little more closely at the PILOT statement above.

- (A) The line number is 10.
- (B) The instruction is **T:** . The **T:** stands for TYPE.
- (C) "WELCOME TO PILOT" is what the computer is instructed to type on the terminal.

And here's something you can think about.

Look at this line from a PILOT program:

>50 T:HOW DO PEOPLE TEACH COMPUTERS

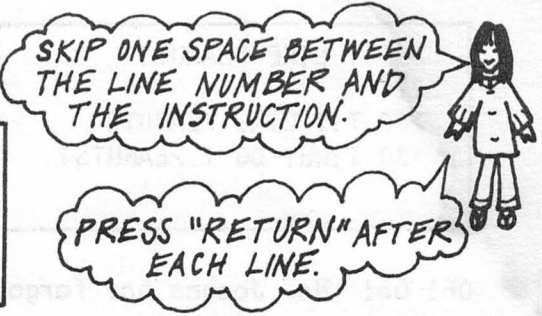
What is the line number? 50
What is the instruction? T:

1-2

Prepare the computer for a PILOT program (enter into PILOT), using the same name you chose in 1-1.

Now, type this:

>10 T:I LIKE PEANUTS
>20 T:WHY DO I LIKE PEANUTS?
>30 T:I'LL TELL YOU WHY
>RUN



!! If you have any problems, read the story on the next page!!

What happened? What did the computer type?

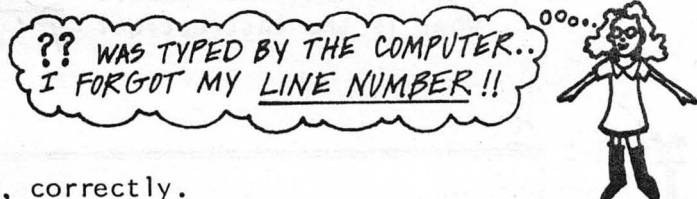
I like peanuts
why do I like peanuts
I'll tell you why

IF YOU MAKE A MISTAKE, YOU CAN CORRECT IT!

This story about Joanne should help you see how to correct mistakes.

Joanne tried to type in her program and the following things happened:

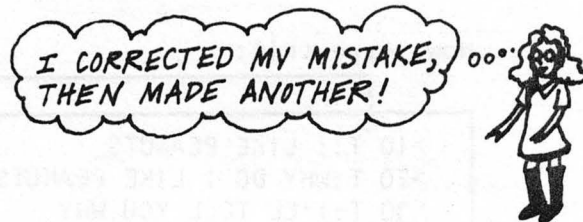
```
> T:I LIKE PEANUTS
??
```



So Joanne typed the line again, correctly.

She put a line number at the beginning of her statement, and got this far:

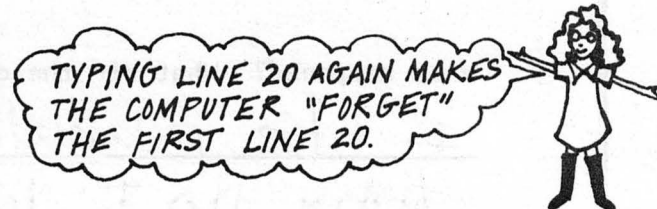
```
> T:I LIKE PEANUTS
??
>10 T:I LIKE PEANUTS
>20 T:WHY DO I PEANUTS?
>
```



Oh! Oh! Now Joanne has forgotten a word! Which word did Joanne forget?

Joanne did not know what to do; so her teacher suggested she try this after line 20:

```
>10 T:I LIKE PEANUTS
>20 T:WHY DO I PEANUTS?
>20 T:WHY DO I LIKE PEANUTS?
```



Joanne finished her program with no further problems. After line 20 she typed 'RUN' and waited for the results!

* * *

Here are some "ERROR MESSAGES" you might see:

```
??
```

```
NO COLON
```

```
NO INSTRUCTION
```



* * *

COMMANDS

This PILOT program contains three **T:** instructions.

```
>10 T:I LIKE PEANUTS  
>20 T:WHY DO I LIKE PEANUTS?  
>30 T:I'LL TELL YOU WHY  
>
```

EACH INSTRUCTION TELLS THE COMPUTER TO TYPE A MESSAGE ON THE TERMINAL.



Does the computer follow each instruction as soon as you type it in?

No, never! The computer waits until you command it to follow all of the instructions in the program.

Typing 'RUN' after line 30 commands the computer to follow the instructions.

```
>10 T:I LIKE PEANUTS  
>20 T:WHY DO I LIKE PEANUTS?  
>30 T:I'LL TELL YOU WHY  
>RUN
```

RUN COMMANDS THE COMPUTER TO FOLLOW THE INSTRUCTIONS.



*Did you notice that RUN does not have a line number? Do you know why?

The reason is because RUN is not a statement. It is not an instruction either. RUN is not even part of the program. RUN is a command!

What are commands ??



Commands tell the computer what to do with a program. Since they are not statements and are not part of the program, commands need no line numbers!

On the next few pages you'll learn how several PILOT commands work.

RUN Command



RUN tells the computer to follow all the instructions in a program.

Example:

```
>10 T:I LIKE PEANUTS
>20 T:WHY DO I LIKE PEANUTS?
>30 T:I'LL TELL YOU WHY
>RUN
I LIKE PEANUTS
WHY DO I LIKE PEANUTS?
I'LL TELL YOU WHY

*READY
```

"RUN" COMMANDS THE COMPUTER TO FOLLOW THE 3 MESSAGES IN THE PROGRAM.



"*READY" IS A SPECIAL MESSAGE THE COMPUTER TYPES AFTER 'RUNNING' YOUR PROGRAM.

1-3

Enter into PILOT. Use the same name you used in 1-1 .
If the computer types:

(OLD PROGRAM)

after your program's name, don't worry. That message means that your program's name has been used before (by you!). For example,

PILOT PROGRAM NAME? WILLY (OLD PROGRAM)
>

THE COMPUTER TELLS YOU THE PROGRAM 'WILLY' IS ALREADY STORED IN ITS MEMORY.



A) Now type this:

```
>DEL
>10 T:MY NAME IS FRED
>20 T:I AM A STAR
>LIS
```

"DEL" CAUSES THE COMPUTER TO FORGET THE OLD PROGRAM.

What does LIS cause the computer to type?

10 T:my name is Fred
20 T:I am a star
7

IF YOU MAKE A MISTAKE REMEMBER YOU CAN FIX IT!



- B) Your name might be Sandra instead of Fred. Naturally you want to use your own name, but how can you do this?

Type this:

```
>10 T:MY NAME IS SANDRA
>LIS
```

USE YOUR OWN NAME
INSTEAD OF SANDRA.



Now, what does LIS cause the computer to type?

```
10 T:my name is Gail
20 T: I am a star
>
```

The old line 10 should be forgotten by the computer. The line you just typed should have replaced it.

Try to replace line 20 with this line:

```
>20 T:I AM A HORSE
```

LIS the program.

```
>LIS
10 T:MY NAME IS SANDRA
20 T:I AM A HORSE
>
```

YOUR PROGRAM SHOULD
LOOK LIKE THIS, THOUGH
YOUR OWN NAME MIGHT
REPLACE 'SANDRA'.



If your program looks like this then continue at (D).

1-3

C) To repair your program follow the steps below.

1. Replace your lines 10 & 20 with the correct lines 10 & 20. If just one of your lines is different, then replace only that line.
2. LIS your program. If your program looks correct, then continue at (D).
3. If LIS types more than 2 lines, then type this:

```
>DEL  
>10 T:MY NAME IS SANDRA  
>20 T:I AM A HORSE  
>LIS
```

REMEMBER YOU CAN
USE YOUR OWN NAME.



D) What if you want to add a new line to this program and put it between lines 10 & 20?

Type the following and see what happens:

```
>15 T:I WALK THROUGH GREEN FIELDS  
>LIS
```

Is line 15 in the correct place in your program?

You have just inserted line 15 into your program.

```
>LIS  
10 T:MY NAME IS SANDRA  
15 T:I WALK THROUGH GREEN FIELDS  
20 T:I AM A HORSE  
>
```

YOUR PROGRAM
SHOULD LOOK LIKE
THIS.



Insert this line into your program:

```
>17 T:ON FOUR LEGS
```

YOU CAN LIS THE PROGRAM
TO SEE THAT IT WAS
INSERTED CORRECTLY.



1-3

Compare your listing with this one:

```
>LIS
10 T:MY NAME IS SANDRA
15 T:I WALK THROUGH GREEN FIELDS
17 T:ON FOUR LEGS
20 T:I AM A HORSE
>
```

By now you know that LIS tells the computer to copy your program.

E) It is now time for the computer to follow all of the instructions in your program.

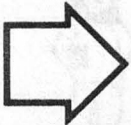
Which command will cause the computer to do this? Run

Type that command and press "RETURN".

IF YOU'RE NOT SURE, SEE P. 6



LIS Command



LIS tells the computer to list all the lines in a program in numerical order.

```
>20 T:I AM 9 YEARS OLD
>30 T:I WAS BORN IN WEST OAKLAND
>40 T:MY BIRTHDAY IS AUGUST 1
>LIS
20 T:I AM 9 YEARS OLD
30 T:I WAS BORN IN WEST OAKLAND
40 T:MY BIRTHDAY IS AUGUST 1
>10 T:MY NAME IS FIFI
>LIS
10 T:MY NAME IS FIFI
20 T:I AM 9 YEARS OLD
30 T:I WAS BORN IN WEST OAKLAND
40 T:MY BIRTHDAY IS AUGUST 1
>
```

NOTICE THAT LIS IS DONE IMMEDIATELY BY THE COMPUTER.

THESE ARE BOTH 'LISTINGS' OF THE PROGRAM AT DIFFERENT TIMES.



Prepare the computer for the same PILOT program you worked on in exercise 1-3 .

Then type:

```
>LIS
```

USE THE SAME
PILOT PROGRAM NAME



What did you discover?

- A) Now try to insert these two lines into your program between lines 17 and 20.

```
T:I CLIMB TREES AND  
T:I CAN RUN VERY FAST
```

REMEMBER, INSERTING
MEANS ADDING NEW
LINES TO YOUR
PROGRAM.



Your program should look like this:

```
>LIS  
10 T:MY NAME IS SANDRA  
15 T:I WALK THROUGH GREEN FIELDS  
17 T:ON FOUR LEGS  
18 T:I CLIMB TREES AND  
19 T:I CAN RUN VERY FAST  
20 T:I AM A HORSE  
>
```

LIS IT AND SEE!



Oops! Line 18 looks strange in this program!
Horses can't climb trees!
How can you get rid of line 18?

You can erase line 18.

Type:

```
>18
```

JUST TYPE 18 AND PRESS "RETURN"



Now LIS your program. What happened to line 18?

You can erase any line in your program by typing the line number and pressing "RETURN". Of course we don't wish to erase any more lines yet.


1-4

```

>LIS
10  T:MY NAME IS SANDRA
15  T:I WALK THROUGH GREEN FIELDS
17  T:ON FOUR LEGS
19  T:I CAN RUN VERY FAST
20  T:I AM A HORSE

```

YOUR PROGRAM SHOULD LOOK LIKE THIS. IF NOT, THEN YOU CAN EITHER REPLACE, INSERT OR ERASE STATEMENTS UNTIL IT DOES.



B) Now, what if you want to insert these two statements

```

T:WHEN I AM HAPPY
T:OR FRIGHTENED

```

between lines 19 and 20?

There is no room there for new lines, so now what do we do?

To solve the problem, let's try something new.


Type this:

```

>REN
>LIS

```

REN IS A COMMAND. IT MEANS 'RENUMBER'



Which new line numbers did REN create?

What were the old line numbers?

10
20
30
40
50

10
15
17
19
20


```

10  T:MY NAME IS SANDRA
20  T:I WALK THROUGH GREEN FIELDS
30  T:ON FOUR LEGS
40  T:I CAN RUN VERY FAST
50  T:I AM A HORSE
>

```

YOUR PROGRAM SHOULD NOW LOOK LIKE THIS.

REN HAS CAUSED ALL THE LINES TO BE NUMBERED BY 10'S.

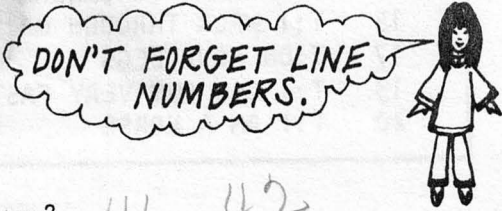


The old lines 19 and 20, are now called lines 40 and 50. Now you can insert statements between them!

1-4

C) Insert these two lines

```
T:WHEN I AM HAPPY
T:OR FRIGHTENED
```



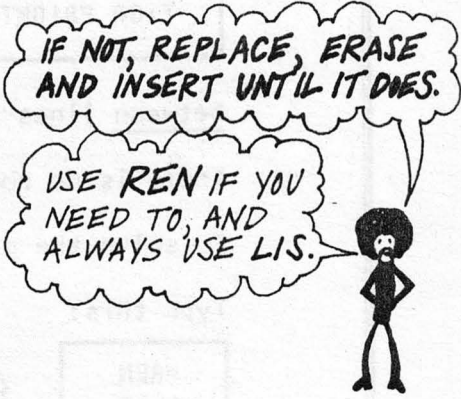
between lines 40 and 50.

What line numbers did you use? 41 42

LIS your program and then type REN.
Type LIS again.

Your program should now look like this:

```
>LIS
10 T:MY NAME IS SANDRA
20 T:I WALK THROUGH GREEN FIELDS
30 T:ON FOUR LEGS
40 T:I CAN RUN VERY FAST
50 T:WHEN I AM HAPPY
60 T:OR FRIGHTENED
70 T:I AM A HORSE
>
```



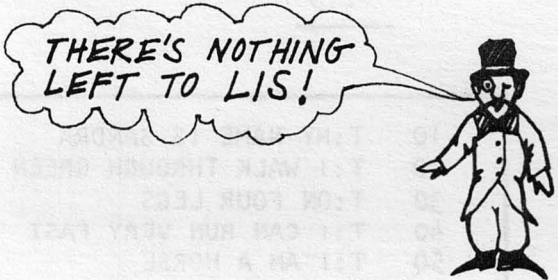
Now you can RUN the program.

DEL Command



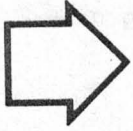
DEL tells the computer to forget (delete) all the statements in a program.

```
>10 T:MY NAM IS STEVE
>20 T:I SIX YEARS OLD
>30 T:I LIKE CONPOOTERS
>DEL
>LIS
>
```



Why was there nothing to LIS?

END Command



END tells the computer that you wish to stop working on a PILOT program. You can always work on the program again at a later time.

```
XEQ-$PILOT
PILOT PROGRAM NAME? RICK
>10 T:MY NAME IS RICK
>20 T:I AM 15 YEARS OLD
>30 T:I THINK COMPUTERS ARE FUN
>END

*READY
```

I JUST HAVE TIME TO WRITE PART OF MY PROGRAM NOW. I'LL FINISH IT LATER.



When you want to begin working on the program again, you must re-enter PILOT.

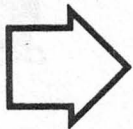
```
XEQ-$PILOT
PILOT PROGRAM NAME? RICK (OLD PROGRAM)
>LIS
 10 T:MY NAME IS RICK
 20 T:I AM 15 YEARS OLD
 30 T:I THINK COMPUTERS ARE FUN
>40 T:I AM A STUDENT AT GARFIELD SCHOOL
>
```

MY PROGRAM IS STILL IN THE COMPUTER.

I TYPED LIS TO SEE MY PROGRAM BEFORE GOING ON.



REN Command



REN tells the computer to renumber all of the statements in your program by tens starting with 10.

```
>LIS
 5 T:THIS IS A STORY
 7 T:ABOUT A VERY SPOOKY
10 T:HOUSE ON A HILL
>REN
>LIS
10 T:THIS IS A STORY
20 T:ABOUT A VERY SPOOKY
30 T:HOUSE ON A HILL
>
```

REN RENUMBERS THE LINES BY 10'S. IT DOES NOT LIST THE PROGRAM. ONLY LIS DOES THAT.



EDITING



Any time that you change the lines in a program, you are editing that program. Making changes in lines, adding new lines, or removing lines are all forms of editing.

Here is how some editing is done in PILOT.

Replacing a Line (making changes in a single line)

```
>10 T:HULLO
>20 T:WHAT IS YOUR NAME
>10 T:HELLO
>LIS
  10 T:HELLO
  20 T:WHAT IS YOUR NAME
>
```

'HULLO' IS CHANGED TO 'HELLO' BY TYPING LINE 10 AGAIN.

IF YOU TYPE LIS YOU CAN SEE THAT THE NEW LINE 10 HAS REPLACED THE OLD LINE 10.



Inserting a Line (adding a line to the program)

```
>10 T:ONE POTATO
>20 T:TWO POTATO
>30 T:FOUR
>25 T:THREE POTATO
>LIS
  10 T:ONE POTATO
  20 T:TWO POTATO
  25 T:THREE POTATO
  30 T:FOUR
>
```

LINE 25 IS TYPED FOR THE FIRST TIME.

THE "LISTING" SHOWS THAT LINE 25 WAS INSERTED IN THE PROPER PLACE IN THE PROGRAM.



Erasing a Line (removing a line from the program)

```
>10 T:THIS IS A VERY FUN GAME
>20 T:FUN GAME IF
>30 T:IF YOU STICK TO IT.
>20
>LIS
  10 T:THIS IS A VERY FUN GAME
  30 T:IF YOU STICK TO IT.
>
```

TYPING 20 AND PRESSING "RETURN" CAUSES THE COMPUTER TO ERASE LINE 20.



1-5

Prepare the computer for the same PILOT program you worked on in exercise 1-4 .

Type:

```
>DEL
>LIS
```

What happened?

NO Program

Now type this program:

```
>1 T:THERE WAS A MOUSE
>2 T:WHO WAS A LOUSE
>3 T:THEIR WUZ A KAT
>4 T:THERE WAS A CAT
>5 T:WHO ATE A PICKLE
```

A) Erase the extra line in the program.
LIS the program.

B) Change the last line to make a rhyme.
Then type LIS.

*REPLACE LINE 5 WITH
A DIFFERENT LINE 5.*



C) Now insert this line into your program:

```
T:IN A HOUSE
```

*HINT: YOU CAN 'RENUMBER' AND
LIS THE PROGRAM FIRST.*



LIS the program.

D) Insert a new line into the program. Make up one you like. Insert more lines if you want. LIS the program. Each time there is a mistake, correct it and LIS the program. When finished, ask your teacher to RUN the program.

Remember: If you want to work on the program after you have RUN it, just re-enter PILOT, using the same name.

CREATING PICTURES WITH PILOT

You can teach the computer to draw designs and pictures!

A good way to start is by drawing a design on graph paper.


Then you can write a program, instructing the computer to draw you design, one line at a time.

Look at the program Scott wrote:

```
>10 T:THIS IS A SMILING FACE
>20 T:
>30 T:      * *
>40 T:      *
>50 T:      * * * * *
>
??
>50 T:      * *
>60 T:      ***
>LIS
10 T:THIS IS A SMILING FACE
20 T:
30 T:      * *
40 T:      *
50 T:      * *
60 T:      ***
>RUN
THIS IS A SMILING FACE

      * *
      *
      * *
      ***

*READY
```



OOPS! NOT A VERY GOOD SMILE. I THINK THE SMILE SHOULD USE 2 T: STATEMENTS.

I REPLACED LINE 50 AND INSERTED LINE 60.



It is easy to make mistakes when creating a computer picture so don't give up.

Count spaces carefully! And remember, you can edit your program as much as you need to.

Important! Some characters have special meaning in PILOT. When drawing a design, use any characters except: (+) (↑) (↩)

Enter PILOT. Use the same program name you used before.

Delete the old program.
What command did you use? _____

CHECK P. 12
IF YOU DON'T
REMEMBER



Type in this program.

```
>10 T:   X
>20 T:  X  X
>30 T: X X X X X
>LIS
```

Then RUN the program.

```
>RUN
  ^  X  X
  ^ X  X X
  X X X X X

*READY
```

IT SHOULD LOOK LIKE THIS.
IF NOT, RE-ENTER PILOT
AND EDIT THE PROGRAM.



- B) Can you turn this triangle into a house?
Try it!
When you are finished, go on to part (C).

HINT: ADD SOME
LINES TO THE PROGRAM.

- C) This part of the exercise can be done away from the terminal.



Draw a design on the graph paper on the next page.
Place a line number and a **T:** in front of each line.

The graph paper might look something like this:

| | | | | | | | | | | |
|----|----|---|---|---|---|---|---|---|--|--|
| 10 | T: | X | X | X | X | X | | | | |
| 20 | T: | | X | | | | | | | |
| 30 | T: | | X | X | X | X | X | X | | |
| 40 | T: | | X | X | X | X | X | X | | |
| 50 | T: | | X | X | X | X | X | X | | |

10 T: X X X X X X

20 T: X X X X X

30 T: X X X X X

40 T: X X X X X

50 T: X X X X X X

- D) Return to a terminal.
Enter PILOT using the same program name you used before.

'DELeTe' the old program.
Type in your picture program.

Edit the program, if needed, until it is correct.

RUN the program when it's finished. *STOP*

- E) You have just finished your very own program! It is now saved in the computer's memory. You know that you can RUN it again by re-entering PILOT and typing:

```
>RUN
```

You can also RUN the program, without re-entering PILOT, by typing:

```
XEQ-
```

followed by the name of your program.

For example:

```
XEQ-SMILE
```

SMILE IS MY PROGRAM NAME.

causes the computer to RUN the program SMILE.

Try to RUN your program without re-entering PILOT.

You can also RUN your program on other terminals!

For example, if:

```
XEQ-SMILE
```

is typed on another terminal, program SMILE will be RUN there.

Show your program to a friend!



REVIEW OF CHAPTER ONE

PILOT is a computer language developed for interactive conversation.

Entry into PILOT

Be sure you are logged in.
Follow these steps:

```
(A) XEQ-$PILOT
(B) PILOT PROGRAM NAME? 
(D) > (C)
```

- (A) - you type this and press "RETURN"
- (B) - computer types this
- (C) - you type a name (5 or less letters or numbers)
- (D) - computer types this when it is ready for your program

Sample PILOT Program

```
>10 T:HOW ARE YOU
>20 A:
>30 M:FINE,GOOD,TERRIFIC,FAR OUT
>40 T Y:I'M HAPPY TO HEAR THAT
>50 T N:THAT'S A SHAME!
>60 T:HAVE A NICE DAY
```

A PILOT program is a list of numbered PILOT statements. Each statement contains an instruction for the computer to follow.

The PILOT Statement

```
>10 T:WELCOME TO PILOT
```

(A) (B) (C)

- (A) - line number
- (B) - PILOT instruction (always followed by a ":")
- (C) - the part the instruction operates with (operand)

Commands

>RUN tells the computer to follow all instructions in a program.

>LIS lists all of the lines in your program in order.

>REN renumbers all of the lines by 10's, starting with 10.

>DEL deletes all the lines in a program.

>END tells the computer you want to stop working on your program.

The computer cannot follow the instructions in a program until told to do so.

COMMANDS tells the computer what to do with the instructions or the lines in a program.

REVIEW OF CHAPTER ONE (continued)

Editing (Correcting) Your Program

```

>10 T:THIS IS A GUESSING GAME
>5 T:HELLO THERE!
>20 T:I WILL ASK YOU QUESTIONS
>20 T:I WILL ASK YOU QUESTIONS
>30 T:DO YOU WANT THE RULES
>40 T:ABOUT DIFFERENT ANIMALS
>30
>50 T:DO YOU WANT THE RULES
>LIS
>5 T:HELLO THERE!
>10 T:THIS IS A GUESSING GAME
>20 T:I WILL ASK YOU QUESTIONS
>40 T:ABOUT DIFFERENT ANIMALS
>50 T:DO YOU WANT THE RULES
    
```

Line 5 is inserted

The second line 20 will replace the first

Line 30 will be erased

LIS shows that all changes have been made

PILOT Picture Program

```

>10 T:X X X X X
>20 T:X X
>30 T:X X X
>40 T:X X
>50 T:X X X X X
    
```

Any characters may be used in a design except:



Exit from PILOT

```

>END
*READY
    
```

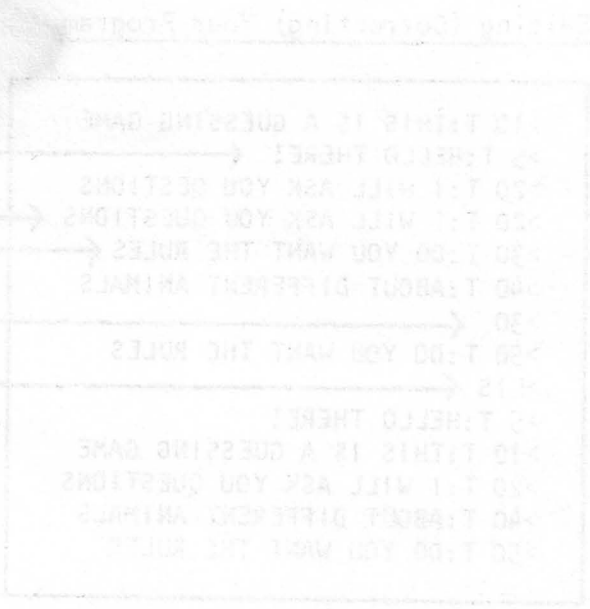
OR

```

>RUN
____
____
____
____
*READY
    
```

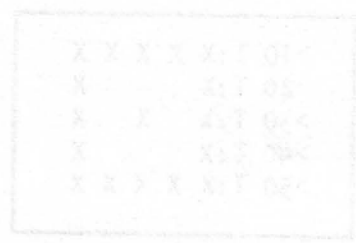
Typing END or 'RUNning' a program brings you "out of PILOT". To work on a PILOT program again, re-enter PILOT.

REVIEW OF CHAPTER ONE (continued)



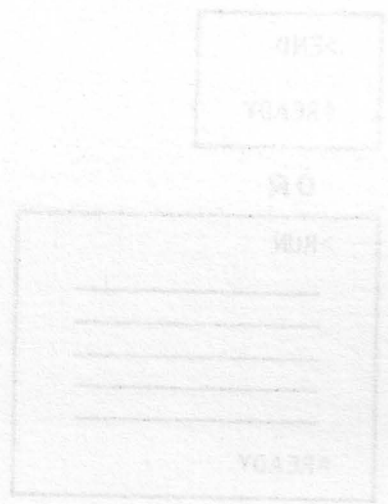
Line 2 is inverted
 The second line 30 will
 will replace the first
 line 30 will be erased
 It shows that all changes
 have been made

First picture program



Any character may be used
 in a design

EXIT type pilot



Typing END or typing a
 program brings you
 out of the
 to work on a pilot
 program again, reenter
 pilot.

CHAPTER TWO

| | |
|---|----|
| Interactive Programs | 23 |
| TYPE Statement <input type="text" value="T:"/> | 25 |
| ANSWER Statement <input type="text" value="A:"/> | 26 |
| Input and Output | 28 |
| A Special Character "+" | 30 |
| Variables | 31 |
| Variable Names | 33 |
| ANSWER Statement <input type="text" value="A:"/> With Variables | 33 |
| The Value of a Variable | 34 |
| TYPE Statement <input type="text" value="T:"/> With Variables | 37 |
| Mystery Programs With Variables | 39 |

CHAPTER TWO

| | |
|----|-------------------------------------|
| 23 | Interactive Programs |
| 25 | TYPE Statement [T] |
| 26 | ANSWER Statement [A] |
| 28 | Input and Output |
| 30 | A Special Character "+" |
| 31 | Variables |
| 32 | Variable Names |
| 33 | ANSWER Statement [A] With Variables |
| 34 | The Value of a Variable |
| 37 | TYPE Statement [T] With Variables |
| 39 | Mystery Programs With Variables |

INTERACTIVE PROGRAMS

What is an interactive program?

Here's one (during a RUN).

```
>RUN  
HELLO. WHAT'S ON YOUR MIND  
?
```

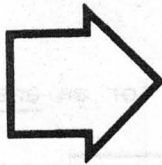
AFTER PRINTING THIS "?" THE COMPUTER STOPS AND WAITS.



The program cannot continue, until someone types-in a response:

```
>RUN  
HELLO. WHAT'S ON YOUR MIND  
? A NEW SONG  
TELL ME MORE ABOUT THAT  
?
```

AFTER SOMEONE TYPES-IN A RESPONSE AND PRESSES "RETURN" THE PROGRAM CONTINUES.



An INTERACTIVE program is one which instructs the computer not only to type messages, but also to wait for responses.

The person who types-in responses is said to "interact" with the program.

To write an interactive program you need at least 2 kinds of statements:

- ① TYPE `T:` statements
- ② ANSWER `A:` statements

★ 2-1

Enter PILOT using a new program name.
Type in this program:

```
>10 T:HI, MY NAME IS ALEX  
>20 T:WHAT IS YOUR NAME  
>30 A:  
>40 T:I LIKE PIES  
>50 T:DO YOU  
>60 A:  
>RUN
```

THIS PROGRAM CONTAINS 2 KINDS OF PILOT STATEMENTS. DO YOU SEE THEM?



2-1

The computer should print:

```
HI, MY NAME IS ALEX
WHAT IS YOUR NAME
?
```

THE COMPUTER STOPS AND WAITS FOR YOU TO TYPE-IN AN ANSWER.



Type in something right after the "?". Press "RETURN".

Then you should see this on your terminal:

```
I LIKE PIES
DO YOU
?
```

THE COMPUTER AGAIN WAITS FOR YOU TO RESPOND.



Type-in a response. Press "RETURN".
The computer will type:

```
*READY
```

Which instruction caused the computer to stop and wait for an answer?



```
A:
```

What did you name your program? Games

* * *

Let's look more closely at the TYPE **T:** statement.

TYPE STATEMENT T:

WHAT IT DOES

GENERAL FORM

T: Instructs the computer to TYPE a message on the terminal.
The message is whatever appears after the T:

[line number] [T:] [something to be typed]


Example 1 -

```
>10 T:PROGRAMMING IS FUN
>20 T:WHAT IS YOUR NAME
```

THESE STATEMENTS CAUSE THE COMPUTER TO TYPE:

```
PROGRAMMING IS FUN
WHAT IS YOUR NAME
```

WHEN THE PROGRAM IS RUN



Example 2 -

```
>100 T:THIS SENTENCE IS TOO
>110 T:LONG TO FIT ON ONE LINE
```



NOTICE THAT EVERY STATEMENT HAS A LINE NUMBER AND A **T:**, EVEN WHEN A SENTENCE IS DIVIDED!


Example 3 -

```
>90 T:**--**
>50 T:
```

T: STATEMENTS MAY BE USED FOR DRAWING DESIGNS.



T: FOLLOWED BY NOTHING CAUSES THE COMPUTER TO PRINT A BLANK LINE.



ANSWER STATEMENT A:

WHAT IT DOES

A: instructs the computer to:

- ① print a "?"
- ② wait for an answer
- ③ save the answer in the computer's memory


GENERAL FORM

[line number] [A:] [usually left blank]

Sample program:

```
>10 T:WHAT IS YOUR NAME
>20 A:
>30 T:OH, THAT'S A NICE NAME
>RUN
WHAT IS YOUR NAME
?
:
:
:
```


THE A: INSTRUCTS THE COMPUTER TO PRINT A ? AND WAIT FOR A RESPONSE.



THE PROGRAM WILL NOT CONTINUE UNTIL A RESPONSE IS TYPED-IN.

```
>10 T:WHAT IS YOUR NAME
>20 A:
>30 T:OH, THAT'S A NICE NAME
>RUN
WHAT IS YOUR NAME
?MIA
OH, THAT'S A NICE NAME
*READY
```

WHEN A RESPONSE IS TYPED-IN AFTER THE ?, THE PROGRAM CONTINUES.



★= 2-2

Enter PILOT. Use the same name you chose in exercise 2-1 .

A) 'DELeTe' the old program

Type:

```
>10 T:WHAT DO YOU THINK OF SCHOOL
>20 T:I'M GLAD TO HEAR THAT
>30 T:WHERE DO YOU GO TO SCHOOL
>40 T:DO YOU LIKE THAT SCHOOL
>RUN
```

What happened? The computer didn't wait for your responses!

What is missing?

Re-enter PILOT using the same name. Try inserting ANSWER **A:** statements in the program. RUN the program to see if it seems complete. When the program is correct, continue with (B).

B) Re-enter PILOT. Use the same program name. 'DELeTe' the old program.

Now, write a program of your own which will ask questions and wait for answers. Be creative!

Speaking of creativity, here's part of a program which Larry wrote:

```
>70 T:WHAT TIME IS IT WHEN AN
>80 T:ELEPHANT SITS ON YOUR FENCE
>90 A:
>100 T:THE ANSWER IS,
>110 T:TIME TO GET A NEW FENCE
```

REMEMBER, YOU DON'T
NEED TO PUT **?**'S
IN YOUR PROGRAM.

A: DOES IT FOR YOU.

Okay, it's your turn. Use your imagination.

When you're finished, ask a friend to try your program.



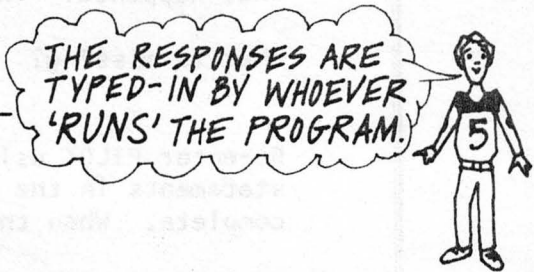
INPUT AND OUTPUT

Look at the LIS and RUN of an interactive program.

```
>LIS
10 T:WHAT'S A NICE PET TO HAVE
20 A:
30 T:WHY DO YOU THINK SO
40 A:
50 T:WHAT ANIMAL WOULD NOT BE A GOOD PET
60 A:

>RUN
WHAT'S A NICE PET TO HAVE
?MONKEYS
WHY DO YOU THINK SO
?THEY CLIMB TREES AND EAT BANANAS
WHAT ANIMAL WOULD NOT BE A GOOD PET
?A GIRAFFE

*READY
```



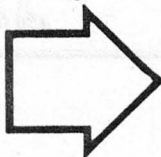
Compare the LIS and RUN. How are they the same? How are they different?

Name all the differences you can see:

NO A in run
NO T
NO H'S

When an interactive program is RUN there are:

- 1 Messages the computer types
- 2 Messages typed-in by the person who 'RUNs' the program



OUTPUT - messages which the computer types on the terminal

INPUT - responses typed-in by the person 'RUNning' a program

This exercise does not need to be done at a terminal.

A) Look at the RUN of the program on the last page.

There were 3 lines of **OUTPUT** produced during the RUN:

- ① WHAT'S A NICE PET TO HAVE
- ② WHY DO YOU THINK SO
- ③ What Animal Would Not Be A Good Pet
(What goes here?)

There were 3 lines of **INPUT**, too. They were:

- ① MONKEYS
- ② THEY CLIMB TREES AND EAT BANANAS
- ③ Giraffe
(What goes here?)

Which PILOT instruction causes OUTPUT? **L15**

Which PILOT instruction allows for INPUT? **A5** RUN

Think about this:

If the same program were RUN again, would the **INPUT** be the same? **Yes**
What about the **OUTPUT**? **NO**

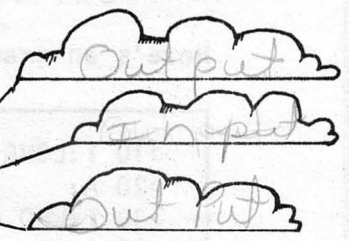
B) Fill in the empty "clouds" below with the word **INPUT** or **OUTPUT**, whichever is correct.

```

XEQ-$PILOT
PILOT PROGRAM NAME? LIKE
>10 T:TELL ME SOMETHING YOU LIKE
>20 A:
>30 T:THAT'S INTERESTING!
>RUN
TELL ME SOMETHING YOU LIKE ←
?BUTTERFLIES AND BIRDS ←
THAT'S INTERESTING! ←

*READY

```

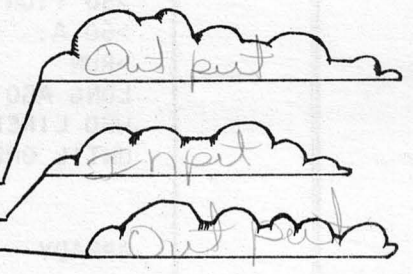


```

XEQ-$PILOT
PILOT PROGRAM NAME? LIKE (OLD PROGRAM)
>RUN
TELL ME SOMETHING YOU LIKE ←
?PLAYING BASKETBALL ←
THAT'S INTERESTING! ←

*READY

```



Which changed each time the program was RUN, the **OUTPUT** or **INPUT**?

A SPECIAL CHARACTER "+"

Look at this interactive program:

```
>10 T:HOW BIG IS A COMPUTER
>20 A:
>RUN
HOW BIG IS A COMPUTER
?VERY BIG

*READY
```

NOTICE THAT THE COMPUTER TYPES A QUESTION ON ONE LINE BUT TYPES A ? ON A NEW LINE.



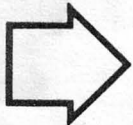
Now, here's another way to write this program:

```
>10 T:HOW BIG IS A COMPUTER+
>20 A:
>RUN
HOW BIG IS A COMPUTER?VERY BIG

*READY
```

SEE THE "+" AT THE END OF LINE 10.

BECAUSE OF THE "+" THE COMPUTER STAYS ON THE SAME LINE.



A "+" at the end of a **T:** statement causes the computer to wait for input on the same line with the output.

Notice that the "+" is not typed on the terminal during the RUN of the program.

What do you think would happen if "+" was in the middle of a **T:** statement?

2-4

Sometimes it's fun to write a program that tells a story, but with parts missing! When someone 'RUNS' the program, they fill in the missing parts; so the story changes each time it's RUN.

Here's an example of a "fill-in" story program.

```
>10 T:LONG AGO THERE LIVED A+
>20 A:
>30 T:WHO LIKED TO+
>40 A:
>50 T:UNTIL ONE DAY+
>60 A:
>RUN
LONG AGO THERE LIVED A ?WICKED WITCH
WHO LIKED TO ?CAST SPELLS
UNTIL ONE DAY ?SOMEONE CAST A SPELL ON HER.

*READY
```

THE STORY IS DIFFERENT EACH TIME IT'S RUN.

THE INPUT DEPENDS ON WHO 'RUNS' THE PROGRAM.



2-4

Write a PILOT "fill-in" story program of your own. You might want to develop the story on paper first, and then type-it-in at the terminal.

Be inventive!

When it's finished
let someone else RUN it, too.

VARIABLES

Have you ever RUN a program which asked you your name and then remembered it? Like this:

```
>RUN  
HI, WHAT'S YOUR NAME  
?JOHN  
GLAD TO MEET YOU JOHN
```

THE INPUT "JOHN" IS USED
LATER IN THE PROGRAM.



You can write this type of program in PILOT! But you will need to use something called a variable.

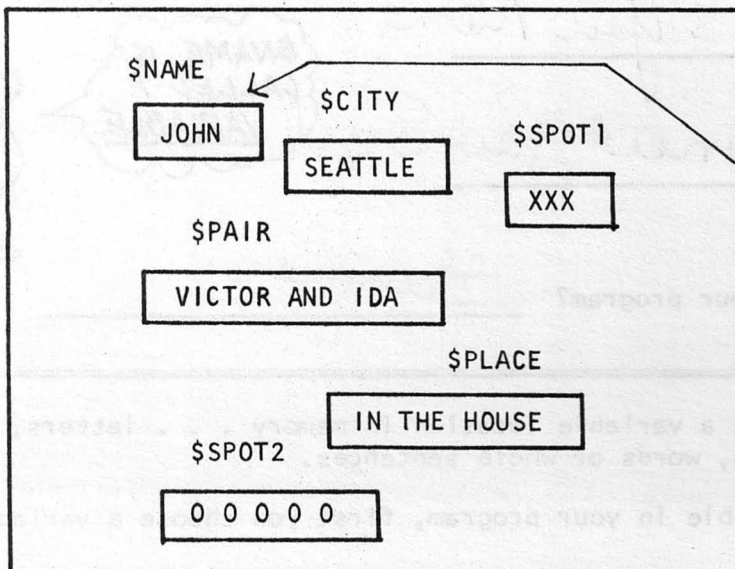
*

*

*

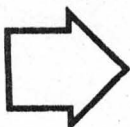
Variables allow you to store messages in the computer's memory.

COMPUTER MEMORY



MESSAGES ARE STORED
IN VARIABLE SPACES IN
THE COMPUTER'S MEMORY.

"JOHN" IS STORED INSIDE
A SPACE CALLED \$NAME.



A variable reserves a special place in the computer's memory. When the computer wants to "remember" a message, it puts the message inside a variable location in memory. Then the computer recalls the message from this space, when needed.

A) Type in a program which, when RUN, would produce this output:

```
>RUN
HI, WHAT'S YOUR NAME
?
GLAD TO MEET YOU
*READY
```

YOUR PROGRAM SHOULD WAIT FOR A RESPONSE



The program should look something like this:

```
>10 T:HI, WHAT'S YOUR NAME
>20 A:
>30 T:GLAD TO MEET YOU
```

RUN the program before going on.

B) Re-enter PILOT using the same program name you used in (A). Change the program to look like this:

```
>10 T:HI, WHAT'S YOUR NAME
>20 A:$NAME
>30 T:GLAD TO MEET YOU $NAME
```

HINT: REPLACE 2 LINES.



What output was produced?

Hi What's your Name
Paul
Glad to meet you Paul

\$NAME IS CALLED A VARIABLE.



What did you name your program?

James

Any message can be stored in a variable location in memory . . . letters, numbers and other characters, words or whole sentences.

When you want to use a variable in your program, first you choose a variable name.

VARIABLE NAMES

GENERAL
FORM

[\$] [any name or word, no longer than 8 characters,
beginning with a letter]

Examples-

\$NAME \$HOB0 \$BONG1 \$CRANBO \$NAME3

WHAT
IT
DOES

Each variable name refers to a specific location in the computer's memory:

\$NAME \$BONG1 \$NAME3

ANSWER STATEMENT A: WITH VARIABLES

HOW
IT
WORKS

A: followed by a variable name reserves a location in memory,
waits for an input and stores the input in the variable location.

GENERAL
FORM

[line number] [A:] [variable name]

Example 1-

>100 A:\$HAPPY

This statement causes the computer to:

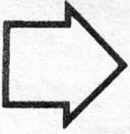
- ① reserve a space in memory named \$HAPPY
- ② print a "?" and wait for input
- ③ store the input in the space named \$HAPPY

Example 2-

>35 A:\$BLEAP 1

This statement causes the computer to reserve a space named \$BLEAP 1, print a "?" and wait for input, store the input in the space named \$BLEAP 1.

THE VALUE OF A VARIABLE



The value of a variable is the message stored inside it.

Sample program-

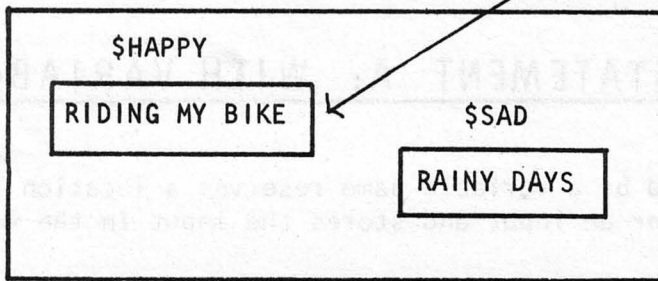
```
>10 T:WHAT MAKES YOU HAPPY
>20 A:$HAPPY
>30 T:WHAT MAKES YOU SAD
>40 A:$SAD
>RUN
WHAT MAKES YOU HAPPY
?RIDING MY BIKE
WHAT MAKES YOU SAD
?RAINY DAYS

*READY
```

WHEN THIS PROGRAM IS RUN \$HAPPY AND \$SAD ARE GIVEN VALUES.



Computer memory after the program is RUN:



"RIDING MY BIKE" IS THE VALUE OF \$HAPPY.



WHAT IS THE VALUE OF \$SAD?

RAINY DAYS

2-6

- A) Enter PILOT, using the same name you used in Exercise 2-5 .
LIS the program.

```
>LIS
10 T:HI, WHAT'S YOUR NAME
20 A:$NAME
30 T:GLAD TO MEET YOU $NAME
>
```

EDIT YOUR PROGRAM IF IT DOES NOT LOOK LIKE THIS.

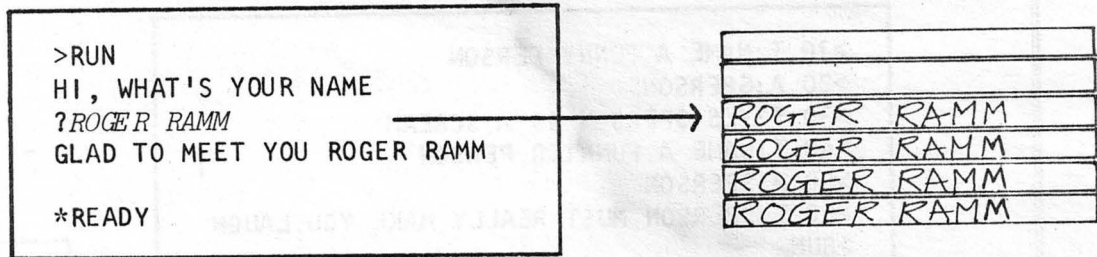


2-6

Let's RUN the program several times, and watch what happens to the variable location called \$NAME, in the computer's memory.

USE "ROGER RAMM" as input.

\$NAME



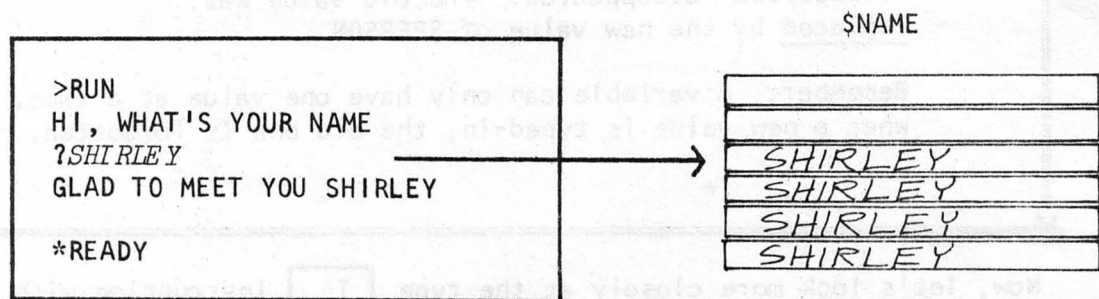
At the beginning of the program there is nothing inside of \$NAME.

When line 20 is RUN, "ROGER RAMM" is placed inside of \$NAME. "ROGER RAMM" becomes the value of \$NAME.

When line 30 is RUN, the value of \$NAME is typed on the terminal.

B) Re-enter PILOT, using the same program name.

RUN the program again, using "SHIRLEY" as input.



This time, when line 20 is RUN, "SHIRLEY" becomes the value of \$NAME.

C) RUN the program once more. This time you make up a name to use as input.

What value does \$NAME have now?

\$NAME

Remember: Each time the program is RUN a different input can be typed-in and the value of \$NAME will change. For that reason we call spaces like \$NAME variables, because their values can vary (change).

2-6

- D) Type this program and RUN it. Use "TWEEDLEDEE" for the first input and "ELMER FUDD" for the second input. Watch what happens to the value of \$PERSON.

```
>10 T:NAME A FUNNY PERSON
>20 A:$PERSON
>30 T:YES $PERSON IS A SCREAM
>40 T:NAME A FUNNIER PERSON
>50 A:$PERSON
>60 T:$PERSON MUST REALLY MAKE YOU LAUGH
>RUN
NAME A FUNNY PERSON
?TWEEDLEDEE
YES TWEEDLEDEE IS A SCREAM
NAME A FUNNIER PERSON
?ELMER FUDD
ELMER FUDD MUST REALLY MAKE YOU LAUGH

*READY
```

\$PERSON

| |
|------------|
| |
| |
| TWEEDLEDEE |
| TWEEDLEDEE |
| TWEEDLEDEE |
| ELMER FUDD |
| ELMER FUDD |
| ELMER FUDD |

Look what happened to \$PERSON!
First, "TWEEDLEDEE" was the value of \$PERSON.
But, as soon as "ELMER FUDD" was typed-in,
"TWEEDLEDEE" disappeared. The old value was
replaced by the new value of \$PERSON.

Remember: A variable can only have one value at a time.
When a new value is typed-in, the old one is forgotten.

*

*

*

Now, let's look more closely at the type **T:** Instruction with variables.

TYPE STATEMENT **T:** WITH VARIABLES

WHAT
IT
DOES

T: followed by a message with variables causes the computer to type a message with the value of each variable appearing in place of the variable name.

GENERAL
FORM

[line number] **T:** [a message with variables]

Examples -

```
>230 T:WHAT'S HAPPENING $NAME!  
>20 T:I LOVE TO EAT $FOOD  
>55 T:$ANSWER  
>10 T:$IDEA IS VERY GOOD $NAME
```

Sample Program -

```
>10 T:WHAT'S YOUR NAME+  
>20 A:$NAME  
>30 T:WHAT'S YOUR FAVORITE THING TO EAT+  
>40 A:$FOOD  
>50 T:WHAT'S HAPPENING $NAME!  
>60 T:I LOVE TO EAT $FOOD, $NAME.  
>RUN  
WHAT'S YOUR NAME?WENDY  
WHAT'S YOUR FAVORITE THING TO EAT?CHEESE  
WHAT'S HAPPENING WENDY!  
I LOVE TO EAT CHEESE, WENDY.  
  
*READY
```

Notice that "WENDY" become the value of \$NAME and "CHEESE" becomes the value of \$FOOD. The values are output when the program is RUN.

Type in this 'mystery story' program, and RUN it to see how it works.

```

>10 T:NAME A DISASTER
>20 A:$DISASTER
>30 T:WHAT IS YOUR NAME
>40 A:$NAME
>50 T:A PERSON YOU KNOW
>60 A:$PERSON
>70 T:WHAT IS YOUR FRIEND'S NAME
>80 A:$FRIEND
>90 T:
>100 T:ONE FINE DAY $NAME TURNED ON THE TELEVISION
>110 T:ONLY TO FIND A NEIGHBOR, $PERSON ON A
>120 T:TELEVISION COMMERCIAL. $NAME CALLED $FRIEND IN
>130 T:TO WATCH IT TOGETHER. THEY SAW THE WHOLE THING AND
>140 T:DECIDED IT WAS A HORRIBLE $DISASTER
>RUN

```

*READY

\$DISASTER

\$NAME

\$PERSON

\$FRIEND

FILL-IN THE VALUES OF THE VARIABLES AS THEY ARE INPUT.



WHAT HAPPENED?
FILL-IN THE 'RUN' OF THE PROGRAM.



MYSTERY PROGRAMS WITH VARIABLES

These are the steps to follow to create a mystery program:

- 1 Write a story, any old story. For example:

Once upon a time a green prince flew to earth from Mars. The prince met a nice princess and lived happily ever after.

- 2 Choose a few words in the story which will be allowed to change. In your PILOT program, you will replace these words with variables. Think of a category for each variable. For example:

Once upon a time a green prince
(colors)


flew to earth from Mars. The
(ways to travel)

prince met a nice princess and
(word to describe someone | something a person could be)

lived happily ever after.

THE WORDS CHOSEN TO CHANGE ARE UNDERLINED.

A CATEGORY WHICH INCLUDES THE WORD IS BELOW EACH.



The categories will help you think up questions for your story program.

- 3 Choose PILOT variable names to go with each of the words you have underlined. The variable names should help you remember your categories. For example:

| | | |
|----------|----|------------------|
| \$COLOR | -- | colors |
| \$MOVED | -- | ways to travel |
| \$NICE | -- | describes people |
| \$PERSON | -- | type of person |

- 4 Think up questions for each of your variables. For example:

NAME A COLOR
(\$COLOR)

THINK OF A WORD THAT DESCRIBES A PERSON
(\$NICE)

WHAT COULD YOU GROW UP TO BE?
(\$PERSON)

NAME A WAY THAT YOU MIGHT TRAVEL
(\$MOVED)

- 5 Program the computer to ask these questions. For example:

```
>10 T:NAME A COLOR
>20 A:$COLOR
```

DO THIS FOR EACH QUESTION.



- 6 Program the computer to tell your story. Be sure to use the variable names in the story. Remember line numbers and `T:`.

```
>100 T:ONCE UPON A TIME A $COLOR PRINCE
>110 T:$MOVED TO EARTH FROM MARS
>120 T:THE PRINCE MET A $NICE $PERSON
>130 T:AND LIVED HAPPILY EVER AFTER
```

THE STORY PART
OF THE PROGRAM
MIGHT LOOK
LIKE THIS.



- 7 This is an example of what your finished program will look like:

```
>LIS
10 T:NAME A COLOR
20 A:$COLOR
30 T:THINK OF A WORD THAT DESCRIBES A PERSON
40 A:$NICE
50 T:WHAT COULD YOU GROW UP TO BE
60 A:$PERSON
70 T:NAME A WAY THAT YOU MIGHT TRAVEL
80 A:$MOVED
90 T:ONCE UPON A TIME A $COLOR PRINCE
100 T:$MOVED TO EARTH FROM MARS
110 T:THE PRINCE MET A $NICE $PERSON
120 T:AND LIVED HAPPILY EVER AFTER
>END

*READY
```

