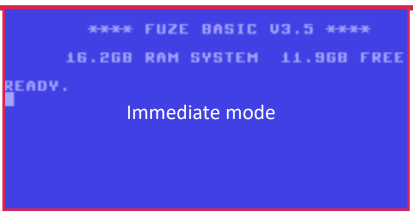
FUZE BASIC

Worksheet: 6a

"More variables"



```
**** FUZE BASIC V3.5 ****
16.2GB RAM SYSTEM  11.9GB FREE
READY.
```

Immediate mode

Computers love numbers. In fact without numbers a computer could not perform even the most lowly of tasks. Even now, the computer you are sitting in front of is throwing around millions, if not billions of numbers every single second, that is unless it is switched off of course!

The computer stores numbers in its memory. Even a simple computer these days has billions of memory locations, so how are we supposed to remember which number we left in which memory location?!

It's a lot easier than you might expect, and in fact, it's something we already do in day-to-day life all the time. We use names to refer to often used numbers; a '**century**' = 100 years, a '**Kilogram**' = 1,000 grams, a '**mile**' = 1,609.34 metres, '**the speed of light**' or '**c**' = 299,792,458 metres per second.

We don't need to write 1,609.34 every time we want to refer to a mile, we just use the word instead.

```
A = 5
B = 5
C = A + B
Print C
```

In **immediate mode**, type in the commands displayed to the left. After each command press the **ENTER** key.

The letters '**A**', '**B**' and '**C**' are called **Variables**.

Try working with basic sums like this but include other arithmetical operators like * (multiply), / (divide) and - (minus).

```
A = A + 1
B = A * A
D = 10 / 2 + A
C = D - B
```

```
A = ( A + 1 ) * B
C = ( 100 * B ) / ( 2 - A)
D = ( 10 / 2 ) + A
C = D - B
```

All these are acceptable. Notice the use of brackets to force sums to happen in the order we need to process the calculations. An open bracket must always have a corresponding close bracket and a closing bracket only closes the last open bracket - there must always be an equal number of brackets.

```
CLS
fuzeguess = RND (10) + 1
INPUT "Hello, enter a number between 1 and 10? ", yourguess
WAIT (1)
CLS
PRINT "Your guess was ";  yourguess
PRINT "My guess was ";  fuzeguess
IF yourguess = fuzeguess  THEN PRINT "Wow, you got it
right!"
If yourguess <> fuzeguess  Then Print "Wrong!"
End
```

Press **F2** to open the editor and enter the program on the left.

RUN with F3

CLS clears the screen.

The computer picks a random number out of 10. As computers start counting from 0, this gives us numbers 0 to 9, so we add 1 to make sure we end up with a number between 1 and 10.

We have introduced a new function;

IF THEN

IF yourguess = fuzeguess THEN
& **IF yourguess <> fuzeguess THEN**

[=] checks if they are **EQUAL TO**

[<>] Checks if they are **NOT EQUAL TO**

ADVANCED CHALLENGE:

+ - / * are generally called operators.

= < > <> <= >= are called comparison operators.

[=] is **EQUAL** [<>] is **NOT EQUAL** [<] is **LESS THAN** [>] is **MORE THAN** [<=] **is LESS THAN OR EQUAL** [>=] is **MORE THAN OR EQUAL**

Knowing this, just before the **END** command, add;

If yourguess < fuzeguess THEN PRINT "You were too low!"

Could you check for "Too High" or "Too Low" as well?

FUZE BASIC

Worksheet: 6b

"More variables"

```
turns = 10
fuzeguess=RND (100)+1
CLS
PRINT "I am thinking of a number between 1 and 100"
WAIT (2)
LOOP
INK = White
PRINTAT (1, 3); "You have "; turns; " turns left"
INPUT "What is your guess ? ", playerguess
WAIT (1)
INK=orange
IF playerguess < fuzeguess THEN PRINT "Too low"
IF playerguess > fuzeguess THEN PRINT "Too high"
WAIT (1)
turns = turns - 1
IF turns < 1 THEN
PRINT "Bad luck! GAME OVER"
END
ENDIF
CLS
REPEAT UNTIL playerguess = fuzeguess
PRINT "WELL DONE"
END
```

Once again, return to the editor with **F2** if you're in immediate mode and then delete any code that is there. The quick way to do this is to click the **New** icon.

Enter the program as displayed on the left. This is quite a long program so you should take care to make sure you get it exact. Otherwise you'll be **debugging**… say what now?

*'**Debugging**' is the name given to working out what is wrong with a program. Bugs can be simple spelling errors, wrong commands and just about anything else.*

*The term originates, according to popular belief, from the mid forties when real bugs and insects would get caught up in computing circuits causing shorts and glitches. To '**Debug**' therefore was to literally, remove bugs from the machine!*

When you're ready **RUN the program with F3**

Play the game!

Let's go through the code to see what is going on.

First we initialise two variables, **turns = 10** and **fuzeguess = RND (100)+1**.

turns is the number of turns we allow the player to guess the number, **fuzeguess** picks a random number between **0** and **99** and then adds **1** so it becomes between **1** and **100**.

The program displays some text and then you're prompted to **INPUT** a number which is stored in the variable **playerguess**.

The main **LOOP** is started.

We compare **playerguess** with **fuzeguess** to see if it is **LESS THAN** [**<**] or **MORE THAN** [**>**] and display a message to match.

Next we reduce the number of turns by 1 with **turns = turns - 1** and then check to see if the player has run out of turns using **IF turns is LESS THAN** [**<**] **1 THEN PRINT …** and end the game.

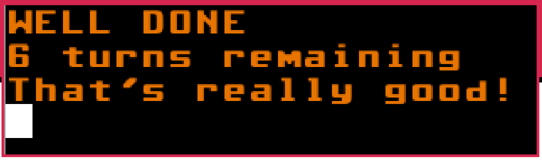
The line **REPEAT UNTIL playerguess = fuzeguess** is the most important one in the entire program. It will jump back to the start of the LOOP at the top unless **playerguess is EQUAL to** [**=**] **fuzeguess** in which case it finishes the program with **PRINT "WELL DONE"**

ADVANCED CHALLENGE:
Make the game harder or easier by changing the value of turns.

Can you add a section before the final END statement that displays how many turns were remaining.

Next add a message depending on how many turns are left so; IF turns > 4 THEN PRINT "That's really good!"

Now do the same but with a negative message if turns is LESS THAN OR EQUAL to 2. Check the first challenge overleaf for a clue.



```
WELL DONE
6 turns remaining
That's really good!
```