The Basics of **BASIC**

BASIC *(Beginner's All-purpose Symbolic Instruction Code) is a computer programming language developed by John G. Kemeny and Thomas E. Kurtz (b. 1928) at Dartmouth College in the mid 1960s. One of the simplest high-level languages, with commands similar to English, it can be learned with relative ease even by schoolchildren and novice programmers. Since c. 1980, BASIC has been popular for use on personal computers.*[1]

For our exercises, we will be using the BASIC-256 version of BASIC. Locate and open the application *BASIC-256* and enter the following codes. Remember to press the green triangle button at the top of the page to run your program.

Exercise #1: This first code simply introduces the *print* command.

```
print “Greetings Earthling!”
```

Exercise #2: Variables

There are two main types of variables in BASIC; numeric and alphanumeric. Numbers are assigned to numeric variables. These variables can be as simple as a single letter or it can be more descriptive. Alphabetical or a combination of alphabetical, numbers, and special characters are assigned to *strings* – similar to the numeric variable but followed by a $ sign. Try the code below.

```
name$="Bruno"
age=6
print "The name of the bear in the music room is";name$
print "The number of months he has lived in the music room is";age
print
print "The name of the bear in the music room is ";name$
print "The number of months he has lived in the music room is ";age
print
print name$;" is the name of the bear living in the music room."
print "He has been in there for ";age;" months."
```

Note that we can combine the phrase inside the quotes with the variables in the print command by using a semicolon (;). Take a close look at the small differences between the three groups of statements and how spaces are used in the command lines. After you have entered and ran the program, try changing the variable names and/or the values assigned to the variables. Do not forget to make sure the variables are changed to the same variables set at the beginning of the program.

[1] https://www.britannica.com/technology/BASIC

Exercise #3: Input

The *input* command allows the user to enter information into the computer. Once again, note the type of variables being used – strings (name$) for alphanumeric and numeric variables (age) for numbers.

```
print "Entering data using the input command."
print
print "What is your name?"
input name$
print
print "How old are you?"
input age
print
print "Thank you ";name$;"."
print "You have entered your age as ";age;" years old."
```

Exercise #4: Input Phrases

The *input* command can be combined with a phrase. Study the syntax used in the code below and note the difference in the output compared to that of exercise #3. Instead of using a semicolon (;) to separate the phrase and the variable, a comma (,) is used in an input statement command.

```
print "Entering data using the input command."
print
input "What is your name? ",name$
input "What is your age? ",age
print
print "Thank you ";name$;"."
print "You have entered your age as ";age;" years old."
```

By using the input statement command, we can shorten the length of the code and still have the same end result as that of exercise #3. Why do you think it is important to find efficient methods when it comes to coding?

Exercise #5: Generating Random Numbers

Generating random numbers is a very common practice in computer programming. In this version of basic, we use the *rand* function to generate a random number. However, the number is always a number less than 1. In order to get a number greater than zero, we need to multiply the random number by 10. This will 'move' the decimal place over by one. Still, all those extra decimal places are 'messy'. We can get rid of them by using the *int* or integer function. Study the syntax of the **rand** and **int** functions carefully.

```
print "Introducing the random number generator."
print
number=rand
print number
print "This number is not very useful. Let's try something else."
print
print number*10
print "This is a bit better, but I still do not like all those numbers after the decimal point."
print
print int(number*10)
print "This one is much better for our purposes."
```

Exercise #6: Using the *if…then* commands

A big part of computer science is decision making. We will start by using a simple *if…then* condition. In other words, *if* a statement is true, *then* a certain action will be taken. The program below will have the computer ask a simple multiplication question and the user will enter a response.

```
a=7
b=8
product=a*b
print "What is ";a;" x ";b;" = ?"
input answer
if answer=product then print "Correct!"
```

Exercise #7: Generating a random number between 1 and 10

In exercise 5, we generated random numbers. This time, we will set the variable *number* to a random number between 1 and 10.

```
print "Generating a Random Number between 1 and 10"
print
number=int(rand*10)+1
print number
```

Exercise #8: Using the *if…then…else* commands

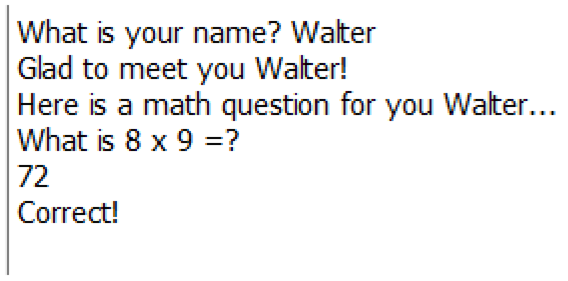
In exercise 6, we used the *if…then* statements to display a message when the answer to a question is correct. However, the answers will not always be correct. Run and study the following code.

```
a=7
b=8
product=a*b
print "What is ";a;" x ";b;" = ?"
input answer
if answer=product then print "Correct!" else print "Sorry, that is incorrect."
```

Exercise #9: Creating a Math Program

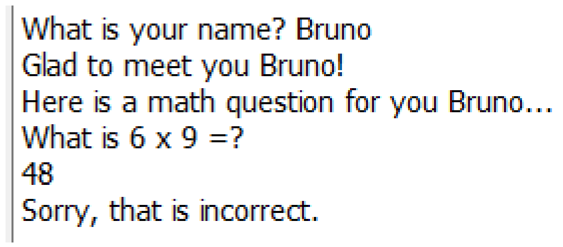
Now you have all the tools to create your first computer program! For this program, have the computer do the following tasks:

- generate and assign two random numbers to two variables
- ask the user for his/her name
- display the question and have the user answer the question
- display an appropriate message for the user’s response to the question
- your program *OUTPUT* should look similar to the following *output*



```
What is your name? Walter
Glad to meet you Walter!
Here is a math question for you Walter...
What is 8 x 9 =?
72
Correct!
```

OR



```
What is your name? Bruno
Glad to meet you Bruno!
Here is a math question for you Bruno...
What is 6 x 9 =?
48
Sorry, that is incorrect.
```

Exercise #10: A Simple Counting Program Using a Loop

Loops and counters are very common in computer programs. Here is an example of a simple counting program using a loop structure. We use the *while* and *end while* structure. Note that the statements between the *while* and *end while* commands are indented. By indenting the statements, we can clearly see the commands being done during the loop. This example also introduces the *rem* or *remark* statement. The computer ignores *rem* statements. They are for the programmers – think of them as reminders or 'sticky notes' that help programmers.

```
rem A Simple Counting Program Using a Loop
counter=0 rem initialize the counter
input "How high shall I count? ",max
while counter<max
        counter=counter+1 rem increases the value of the counter
        print counter
end while
```

Exercise #11a: A Simple Guessing Game

Enter and study the following code for a simple guessing game.

```
rem A Simple Guessing Game
num=int(rand*20)+1 rem set the random number between 1 and 20
print "A Simple Guessing Game"
print
input "Please enter your name: ",name$
print "Thank you ";name$;"."
print
input "I have a number between 1 and 20. Can you guess my number?",guess
if guess=num then print "You got it!" else print "I am sorry, that is not my number."
```

Exercise #11b: A Simple Guessing Game using structured *if…then* statements

Enter and study the following code for a simple guessing game. Note the difference between this one and Exercise 11a.

```
rem A Simple Guessing Game
num=int(rand*20)+1 rem set the random number between 1 and 20
print "A Simple Guessing Game"
print
input "Please enter your name: ",name$
print "Thank you ";name$;"."
print
input "I have a number between 1 and 20. Can you guess my number?",guess
if guess=num then
    print "You got it!"
else
    print "I am sorry, that is not my number."
    print "My number was ";num;"."
end if
```

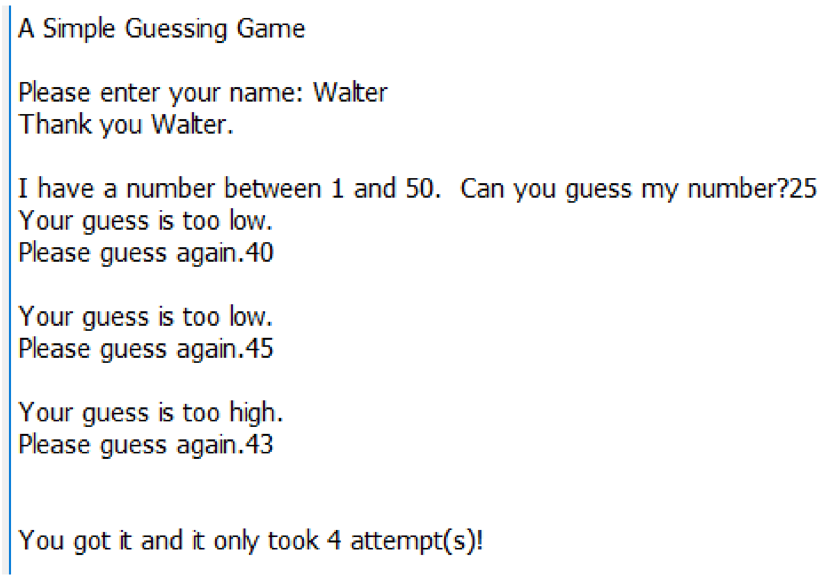
Exercise #11c: A Simple Guessing Game using nested *if…then* statements

Okay, now we get a little more complicated. We will add another level of decision making based on a prior decision. Some programmers call decisions within decisions as *nested if…then* statements. Enter and run the code below – note the indentations.

```
rem A Simple Guessing Game
num=int(rand*20)+1 rem set the random number between 1 and 20
print "A Simple Guessing Game"
print
input "Please enter your name: ",name$
print "Thank you ";name$;"."
print
input "I have a number between 1 and 20. Can you guess my number?",guess
if guess=num then
    print "You got it!"
else
    print "I am sorry, that is not my number."
    if guess<num then
        print "Your guess is too low."
    else
        print "Your guess is too high."
    end if
end if
```

Exercise #12: An Interactive Guessing Game

Time for you to put everything together again! This time, create a guessing game similar to exercises 11a, 11b, and 11c. However, the user will enter his/her name and will keep guessing until he/she guesses the random number. Each time an incorrect guess is entered, the computer will state if the guess is too low or too high. When the user finally guesses the correct number, the computer will display the number of attempts needed to guess the number. *Hint:* You will need to use the *while* and *end while* structure for this exercise. Good luck! Here is a sample output.



```
A Simple Guessing Game

Please enter your name: Walter
Thank you Walter.

I have a number between 1 and 50.  Can you guess my number?25
Your guess is too low.
Please guess again.40

Your guess is too low.
Please guess again.45

Your guess is too high.
Please guess again.43


You got it and it only took 4 attempt(s)!
```