ABCC- Any Body Can Code

Table Of Contents

[Introduction and Objective 2](#_Toc59365469)

[Features of ABCC 2](#_Toc59365470)

[‘Print’ statement 2](#_Toc59365471)

[Purpose 2](#_Toc59365472)

[Usage 2](#_Toc59365473)

[‘Variable’ statement 2](#_Toc59365474)

[Purpose 2](#_Toc59365475)

[Usage 2](#_Toc59365476)

[‘For’ statement 3](#_Toc59365477)

[Purpose 3](#_Toc59365478)

[Usage: 3](#_Toc59365479)

[‘If else’ 4](#_Toc59365480)

[Purpose 4](#_Toc59365481)

[Usage 4](#_Toc59365482)

[‘define method’ 4](#_Toc59365483)

[Purpose 4](#_Toc59365484)

[Usage 4](#_Toc59365485)

[Support statements 5](#_Toc59365486)

# Introduction and Objective

**ABCC** is a Voice-Based Coding platform. **ABCC** is made for anyone who needs a break from typing. **ABCC** was specifically made for the visually impaired who aren’t able to experience the joy of coding. This doc gives details about the features of **ABCC**. It is also a handy guide detailing the voice commands that are currently supported in **ABCC**.

# Features of ABCC

ABCC is made to be as user-friendly as possible. ABCC is a master at user-friendliness.

* ABCC types code for you.
* ABCC uses your mic to get input.
* ABCC needs absolutely no configuration. Install it and code right away
* ABCC has easy yet powerful commands/keywords

# ‘Print’ statement

## Purpose

In ABCC, the print statement is as simple as saying “print hello world”. ABCC’s intelligence automatically detects the context and the statement if used correctly.

## Usage

|  |
| --- |
| User says: “print hello world”  Result: “print(“hello world”)” //To print a –String-  User says: “print placeholder x”  Result: “print(x)” //To print a single –Variable-  User says: “print my name is format name and age is format age”  Result: “print(“My name is {} and age is {}”.format(name, age))” //To print a multiple –Variables- in a –String- |

# ‘Variable’ statement

## Purpose

A variable is a container that stores a value. In ABCC initializing a variable needs no expertise, just say “variable x is equal to ten” and ABCC will automatically decide the type and initialize it for you

## Usage

|  |
| --- |
| User says: “variable x is equal to 10”  Result: “x = 10” //To initialize a variable of type –Int-  User says: “variable y is equal to hello world”  Result: “y = “hello world”” //To initialize a variable of type –String-  User says: “variable a is equal to input”  Result: “a = input()” //To initialize a variable that takes user input  User says: “variable a is equal to input enter a number”  Result: “a = input(“enter a number”)” //To initialize a variable that takes user input by printing a hint on the console  User says: “variable a is equal to variable x”  Result: “a = x” //To initialize a variable that holds the value of another variable  User says: “variable a is equal to variable x plus b”  Result: “a = x” //To initialize a variable that holds the value of another variable added with another variable  Note. Plus will be converted to +, minus will be converted to -, for the multiplication say Unicode star as no language understands x as multiply hence it has a \* as a multiplication operator, divide will be converted to / since the computer understands ‘/’ as divide sign. |

# 

# ‘For’ statement

## Purpose

A “for” loop is a loop that repeats a specific number of times.

To use for loops in ABCC say “for i in range 10”. The for can also be run a specific number of times that is stored in a variable.

## Usage:

|  |
| --- |
| User says: “for i in range 10”  Result: “for i in range(10):” //To initialize a for loop to repeat 10 times  User says: “for i in range 10”  Result: “for i in range(x)”” //To initialize a for loop to repeat x times. If x is 10 then the for will repeat 10 times |

# ‘If else’

## Purpose

A If else statement is generally referred to a decision statement. In ABCC generating a if statement is as simple as making a decision. Note. Greater than, lesser than, greater than or equal to, lesser than or equal to operators are also supported

## Usage

|  |
| --- |
| User says: “if x is equal to 10”  Result: “if x == 10:” //To tell the computer if x is 10 then do something  User says: “else if x is equal to 20”  Result: “elif x == 20:” // else if is written after an if to specify if the above condition isn’t true then check if x is 20 then do something  User says: “else”  Result: “else:” // else is written after a conditon to specify if none of the above condition is true then do this |

# ‘define method’

## Purpose

A define method statement is to define a method. A method contains code that can be called over and over again, without needing to write it again. Defining methods is the key to a structured code and easily changeable code.

## Usage

|  |
| --- |
| User says: “define method x”  Result: “def x():” //defines a method with name x  User says: “add parameters a and b to method x”  Result: “def x( a , b ):” //adds parameters to an already existing function  User says: “return variable x”  Result: “return x” // returns a variable from a function  User says: “return hello world”  Result: “return “hello world”” // returns a value from a method. Int, String also supported  User says: “call method x”  Result: “x()” //calls an existing method x. This executes all the code written in the method x  User says: “call method x with parameters a and b”  Result: “x()” //calls an existing method x with inputs as variable a and variable b. |

# 

# Support statements

|  |  |
| --- | --- |
| Indent | Adds indentation to the line. |
| go to line <line Number> | Goes to the line specified.  e.g Go to line 20  Take the cursor to line 20 |
| Open terminal | Opens the terminal |
| Triangle | Presses “enter” resulting in a new line |
| Read logs | Opens up the logs generated by ABCC consisting of User input, methods called and errors |
| Write | Writes a text without taking into account whether or not statement  e.g- write print hello world  result- writes print hello world |
| Back | Presses backspace resulting in a character being deleted or going back an indent |