# #

text

Examples:

`# This is a comment`

Hash symbol: A comment.

Comments are ignored by the computer and discarded when a program is *translated* into *machine code*. They are used by programmers to explain the purpose of sections of code. This is helpful when you return to a program after a period of time, or when you work in teams.

Comments are typically used:

- At the beginning of a program to explain its purpose.
- Before each subprogram: def
- Before each selection statement: if, else, match, case.
- Before each iteration: for, while.
- To explain difficult to comprehend code.
- To remind the author why unusual approaches have been taken.

Associated keywords: `def`

DEF

def *identifier(parameters)*:

Examples:

```
def square(x):

    x = x * x

    return x
```

Command: Defines a new subprogram. Subprograms are also called *subroutines*.

Code must be indented inside the subprogram. You cannot use spaces in the identifier name of the subprogram. It is common to use underscores to separate words in the name of the subprogram instead.

Don't forget the colon at the end of this command.

Subprograms can be *procedures* that do not return a value or *functions* that do return a value. Procedures are used to structure a program into smaller more manageable parts. This is known as *problem decomposition*. Functions are used to create reusable program components. Subprograms avoid unnecessary code duplication and make the code easier to read which also makes finding errors in code, called *debugging* easier.

Associated keywords: `return`

FLOAT

variable = float(*parameter*)

Examples:

```
x = float("5")
```

```
x = float(6)
```

```
x = float(y)
```

Function: Casts a parameter to a floating point number.

Different types of data are stored in different ways by the computer. Inputs taken from the keyboard are always sequences of alphanumeric characters called *strings*. These sequences of characters are stored as numbers by the computer using a standard such as *ASCII* or *Unicode*. The input string "5.6" (strings are *qualified* by quotes) is stored as the numbers 53, 46, 54. To do calculations on the input, it first needs to be converted from a string to a number, or from 53, 46, 54 to 5.6. This is known as *casting*.

Whole numbers with no fractional part are known as *integers*. Numbers with a fractional part (decimal places) are known as *floating points*, *floats*, *real numbers* or *reals*.

A *run-time error* will occur if the parameter cannot be cast to a float.

Associated keywords: `int`

INPUT

variable = input(*parameter*)

Examples:

```
surname = input("Enter your surname: ")
```

```
prompt = "Enter your forename: "
forename = input(prompt)
```

Function: returns an input from the keyboard.

Inputs from the keyboard allow user interaction with your program. Inputs are always sequences of alphanumeric characters called *strings* terminated when the user presses the enter key.

The parameter prompt to the user is optional but informs the user what data they are expected to enter.

It is common to add an extra space at the end of the prompt to separate the prompt and the input.

Remember to do calculations on the input, the input data will need to be *cast* to an *integer* or *float*.

INT

variable = int(*parameter*)

Examples:

```
x = int("5")
```

```
x = int(6.5)
```

```
x = int(y)
```

Function: Casts a parameter to an integer.

Different types of data are stored in different ways by the computer. Inputs taken from the keyboard are always sequences of alphanumeric characters called *strings*. These sequences of characters are stored as numbers by the computer using a standard such as *ASCII* or *Unicode*. The input string "5" (strings are *qualified* by quotes) is stored as the number 53. To do calculations on the input, it first needs to be converted from a string to a number, or from 53 to 5. This is known as *casting*.

Whole numbers with no fractional part are known as *integers*. Numbers with a fractional part (decimal places) are known as *floating points*, *floats*, *real numbers* or *reals*. Using int to cast a float to an integer also removes the fractional component.

A *run-time error* will occur if the parameter cannot be cast to an integer.

Associated keywords: `float`

PRINT

print(*parameters*)

Examples:

```
print("Hello World")
```

```
print(x)
```

Command: Outputs a value to the screen. The value can be any data type: Boolean, integer, list, float, string.

Don't forget that what you want to output must be enclosed in brackets. Strings will need to be qualified by quotes.

Each print statement puts the output on a new line. You can prevent a new line by concatenating end = "" to the output. E.g.

```
print("Hello", end = "")
```

```
print("World")
```

A single blank line, or the end of a line can be output with:

```
print()
```

Multiple parameters can be output on one line. Each parameter is separated with a comma. The output will include an automatic space between each parameter:

```
print("You are", age, "years old")
```

RETURN

return *expression*

Examples:

```
def square(x):
    return x * x


y = square(5)
```

Command: Returns a value from a subprogram.

Subprograms that return values are called *functions*.

In the example above, the number 5 is passed into the function called square and assigned to the parameter x. The variable is then multiplied by itself and returned as the output from the function square into the variable y which assigns it the value 25.

The return expression can be a Boolean, e.g. return True, a variable, e.g. return total, a list or the result of a calculation.

It is possible to return more than one value in Python, each separated with a comma, but many languages do not support this so it is generally avoided in favour of returning a list.

Subprograms that do not return a value are called *procedures*.

Associated keywords: `def`

Appendix 1

Concatenation

```
x = "Hello" + " " + "World"
```

To concatenate means to join together. A comma can be used to concatenate strings inside a print statement. A plus symbol needs to be used outside of a print statement.

Numbers should be cast to strings before they are concatenated.

Mathematical operators

+	x = 6 + 5	Addition
-	x = 6 - 5	Subtraction
*	x = 6 * 5	Multiplication
/	x = 6 / 5	Division

Command Index