Chapter 1

**Q. What does int(g) mean?**

A:It tells Python to interpret the user’s input as a number rather than a letter.

Within programming languages, the number 5 is different than the letter ‘5’.

**Q: So what if we’d left it out?**

A: The computer would have treated the input entered by the user of the program as a letter. If you ask the computer if a letter is equal to a number, it gets confused and tells you it isn’t.

**Q:Why’s that?**

A:Because if the computer thinks that two pieces of information are of different

“types,” it assumes that there’s no way they can be equal.

**→ Loops let you run the same piece of code over and over again.**

**Programming TOOLS**

\* programs are created from code statements:

commands do things.

Branches decide things.

Loops repeat things.

\* Conditionals help you decide if

somethings is True or False.

\* Assignment sets a name to a value.

\* A named value is stored in a “variable”.

**Python TOOLS**

\*if/else branches

\* while loops

\* = assignment operator

\* == equality operator

\* != inequality operator

\* > greater than operator

\* print() displays a message on the screen

\* input() gets and return user input

\* int() converts characters to numbers

\* randint() produces a random number

**CHAPTER 2**

A string is a series of characters.

The first character in a string has an offset of 0, because it is zero

characters from the start. The second character has an offset of 1, and so on:

**But how do you get at more than one character?**

**S[14] →** If you provide a single index after the variable name, you get a single character.

**S[138:147] →** This will read the smaller substring from the entire string contained within “s”.

→ If you provide two index values , you extract a group of characters from the first index up to(But not including) the second index.

**Q: What does the import line of code do?**

A:It gives the program the ability to talk to the Internet. The

urllib.request code comes as standard with Python 3.

→ You can download the HTML of a web page as a textual **string.**

→ A string is a sequence of characters.

→ You can access individual characters in a string using an **offset**.

S[0],s[1],s[3]...

→ The offset is known as the **index value** of the character (or just **index** for short).

→ Strings within strings are called **substrings.**

→ Substrings are specified using two index values−for example:

text[10:20].

→ The first index value is the location of the first character of the substring.

→ The second index value is the location **after** the last character of the substring (up to, but not including).

→ Subtract the second index from the first to work out how long the substring should be.

**Programming TOOLS**

→ String are sequences of individual characters.

→ Individual string characters are referenced by index.

→ Index values are offsets that start from zero.

→ Methods provide variables with built-in functionality.

→ Programming libraries provide a collection of related pre-built code and functions.

→ As well as having a value, data in variables also have a “data type”

→ Number is a data type.

→ String is a data type .

**Python TOOLS**

→ S[4]- access the 5th character of the variable “s”, which is a string

→ s[b:12]- access a sub-string within the string “s” (up to, but not including)

→ s.find() method for searching strings .

→ s.upper() method for converting strings to UPPERCASE

→ float() converts strings to decimal point numbers known as “floats”

→ + addition operator

→ > greater than operator

→ urllib.request library for talking to the web

→ time library for working with dates/time

**CHAPTER 3**

**Functions**

**Reuse code with functions**

Most programming languages let you create reusable, shareable code with

functions. A function is a chunk of code that you separate out from the

rest of your program, give a name, and then call from your code.

Different languages have different ways of creating functions. In Python,

use the def keyword to define a new function.

In Python, it’s important that you define the function before you use it, so

make sure the code that calls (or uses) the function comes after the definition

of the function:

**Q:The return() command is just like print(), except nothing**

**appears on screen, right?**

**A:**Well... sort of. The print() command is designed

to display (or output) a message, typically on screen. The

return() command is designed to allow you to arrange for a

function you write to provide a value to your program. Recall the

use of randint() in Chapter 1: a random number between

two values was returned to your code. So, obviously, when

providing your code with a random number, the randint()

function uses return() and not print() . In fact, if

randint() used print() instead of return() , it

would be pretty useless as a reusable function.

**Q:So, it’s a case of return() letting a function give you**

**something back?**

**A:**Yes, that’s it, exactly.

**Q:So, using a function lets you share the repeated code in**

**a controlled way?**

**A:** Yes, it does. There’s also a guiding principle among

prgrammers known as **DRY**: **Don’t Repeat Yourself**. Using functions

lets you keep your code DRY.

**Q. What happens if the function omits the return()**

**command? Does each function have to have one?**

**A:** No, the use of return() is not required. In fact, the

current version of your get\_name() function doesn’t use

return() at all. But, your function feels like it gives you

something because it prints the current price on screen. When the

return() command is omitted, a function returns a special no

value. In Python, this value is called None .

**Q. So just to be clear, using return() is optional?**

**A:** Yes, it is.

**Q. Does return() always come at the end of the function?**

**A:** Usually, but this is not a requirement, either. The return()

can appear anywhere within a function and, when it

is executed, control returns to the calling code from that point in the

function. It is perfectly reasonable, for instance, to have multiple

uses of return() within a function, perhaps embedded

with if statements which then provide a way to control which

return() is invoked when.

**Q. Can return() send more than one result back to the**

**caller?**

**A:** Yes, it can. return() can provide a list of results to the

calling code

**Programming TOOLS**

→ Avoid code duplication with functions.

→ Parameters are variables that you can pass to functions.

→ Functions can return values.

→ Computers use stack frames to record and track variables.

→ When you call a function, a new stack frame is created for the function to use.

→ stack frames (and local variables) are thrown away when you exit a function.

→ A variable is said to be “in scope” whenever it’s value can be seen by some code.

**Python TOOLS**

→ Use “def” to create functions.

→ Use return() to send a value back to the code that called the function.

→ Pass parameters to functions by placing them between parameters .