Chapter six Strings.

**6.1 A string is a sequence**

A string is a *sequence* of characters. You can access the characters one at a time

with the bracket operator:

>>> fruit = 'banana'

>>> letter = fruit[1]

The second statement extracts the character at index position 1 from the fruit

variable and assigns it to the letter variable.

The expression in brackets is called an *index*. The index indicates which character

in the sequence you want (hence the name).

But you might not get what you expect:

>>> print(letter)

a

For most people, the first letter of “banana” is “b”, not “a”. But in Python, the

index is an offset from the beginning of the string, and the offset of the first letter

is zero.

>>> letter = fruit[0]

>>> print(letter)

b

So “b” is the 0th letter (“zero-th”) of “banana”, “a” is the 1th letter (“one-th”),

and “n” is the 2th (“two-th”) letter.

You can use any expression, including variables and operators, as an index, but

the value of the index has to be an integer. Otherwise you get:

>>> letter = fruit[1.5]

TypeError: string indices must be integers

LEN

len is a built-in function that returns the number of characters in a string.

Alternatively, you can use negative indices, which count backward from the end of

the string. The expression fruit[-1] yields the last letter, fruit[-2] yields the

second to last, and so on.

**6.3 Traversal through a string with a loop**

A lot of computations involve processing a string one character at a time. Often

they start at the beginning, select each character in turn, do something to it, and

continue until the end. This pattern of processing is called a *traversal*.

This loop traverses the string and displays each letter on a line by itself. The

loop condition is index < len(fruit), so when index is equal to the length of

the string, the condition is false, and the body of the loop is not executed. The

last character accessed is the one with the index len(fruit)-1, which is the last

character in the string.

Slice

segment of a string is called a *slice*.

A segment of a string is called a *slice*. Selecting a slice is similar to selecting a

character:

>>> s = 'Monty Python'

>>> print(s[0:5])

Monty

>>> print(s[6:12])

Python

The operator returns the part of the string from the “n-th” character to the “m-th”

character, including the first but excluding the last.

If you omit the first index (before the colon), the slice starts at the beginning of

the string. If you omit the second index, the slice goes to the end of the string:

>>> fruit = 'banana'

>>> fruit[:3]

*'ban'*

>>> fruit[3:]

*'ana'*

If the first index is greater than or equal to the second the result is an *empty string*,

represented by two quotation marks:

>>> fruit = 'banana'

>>> fruit[3:3]

*''*

An empty string contains no characters and has length 0, but other than that, it

is the same as any other string.

It is tempting to use the operator on the left side of an assignment, with the

intention of changing a character in a string. For example:

>>> greeting = 'Hello, world!'

>>> greeting[0] = 'J'

TypeError: 'str' object does not support item assignment

The “object” in this case is the string and the “item” is the character you tried

to assign. For now, an *object* is the same thing as a value, but we will refine that

definition later. An *item* is one of the values in a sequence.

The reason for the error is that strings are *immutable*, which means you can’t

change an existing string. The best you can do is create a new string that is a

variation on the original:

>>> greeting = 'Hello, world!'

>>> new\_greeting = 'J' + greeting[1:]

>>> print(new\_greeting)

Jello, world!

This example concatenates a new first letter onto a slice of greeting. It has no

effect on the original string.

**6.6 Looping and counting**

The following program counts the number of times the letter “a” appears in a

string:

word = 'banana'

count = 0

**for** letter in word:

**if** letter == 'a':

count = count + 1

print(count)

This program demonstrates another pattern of computation called a *counter*. The

variable count is initialized to 0 and then incremented each time an “a” is found.

When the loop exits, count contains the result: the total number of a’s.

**counter** A variable used to count something, usually initialized to zero and then

incremented.

**empty string** A string with no characters and length 0, represented by two quotation

marks.

**format operator** An operator, %, that takes a format string and a tuple and generates

a string that includes the elements of the tuple formatted as specified

by the format string.

**format sequence** A sequence of characters in a format string, like %d, that specifies

how a value should be formatted.

**format string** A string, used with the format operator, that contains format

sequences.

**flag** A boolean variable used to indicate whether a condition is true or false.

**invocation** A statement that calls a method.

**immutable** The property of a sequence whose items cannot be assigned.

**index** An integer value used to select an item in a sequence, such as a character

in a string.

**item** One of the values in a sequence.

**method** A function that is associated with an object and called using dot notation.

**object** Something a variable can refer to. For now, you can use “object” and

“value” interchangeably.

**search** A pattern of traversal that stops when it finds what it is looking for.

**sequence** An ordered set; that is, a set of values where each value is identified by

an integer index.

**slice** A part of a string specified by a range of indices.

**traverse** To iterate through the items in a sequence, performing a similar operation

on each.