Computers and Programs

Theresa Migler-VonDollen
CMPS 5P
The most common use of personal computers is word processing.

Text is represented in programs by the string data type.

A string is a sequence of characters enclosed within quotation marks or apostrophes.

Practice...

Getting a string as input.

Practice
The String Data Type

- We can access the individual characters in a string through indexing.
- The positions in a string are numbered from the left, starting with 0.
- The general form is `<string>(<expr>)`, where the value of `expr` determines which character is selected from the string.
- In a string of `n` characters, the last character is at position `n - 1` since we start counting with 0.
- We can index from the right side using negative indexes.
The String Data Type

- Indexing returns a string containing a single character from a larger string.
- We can also access a contiguous sequence of characters, called a *substring*, through a process called *slicing*.
- Slicing: `<string>[:<end>]`
- `start` and `end` should both be ints.
- The slice contains the substring beginning at position `start` and runs up to but doesn’t include the position `end`.
- If either expression is missing, then the start or the end of the string are used.
The String Data Type

- Can we put two strings together into a longer string?
- Concatenation “glues” two strings together +.
- Repetition builds up a string by multiple concatenations of a string with itself ⋆.
- The function len will return the length of a string.
Simple String Processing

- Usernames on a computer system
- Use the first initial and the first seven characters of the last name.
- Practice.
Another use - converting an int that stands for the month into the three letter abbreviation for that month.

Store all the names in one big string: “JanFebMarAprMayJunJulAugSepOctNovDec”.

Practice.

One weakness - this method only works where the potential outputs all have the same length.

How could you handle spelling out the months?
It turns out that strings are really a special kind of sequence, so these operations also apply to sequences.

Strings are always sequences of characters, but lists can be sequences of arbitrary values.

Lists can have numbers, strings, or both.

We can use the idea of a list to make our previous month program even simpler.

We change the lookup table for months to a list.

Practice.
Lists are *mutable*, meaning they can be changed.

Strings can not be changed.

Inside the computer, strings are represented as sequences of 1’s and 0’s, just like numbers.

A string is stored as a sequence of binary numbers, one number per character.

It doesn’t matter what value is assigned as long as it’s done consistently.
In the early days of computers, each manufacturer used their own encoding of numbers for characters.

ASCII system (American Standard Code for Information Interchange) uses 127 bit codes.

Python supports Unicode (100,000+ characters).

The ord function returns the numeric (ordinal) code of a single character.

The chr function converts a numeric code to the corresponding character.

Using ord and chr we can convert a string into and out of numeric form.
Write a “secret code” program.
For each character in a message print the corresponding number of the character.

A for loop iterates over a sequence of objects, so the for loop looks like: `for ch in <string>`. 

Practice.
Now write a decoder.

Outline:
get the sequence of numbers to decode
message = ""
for each number in the input:
    convert the number to the appropriate character
    add the character to the end of the message
print the message
The variable `message` is an accumulator variable, initially set to the empty string, the string with no characters (" ").

Each time through the loop, a number from the input is converted to the appropriate character and appended to the end of the accumulator.

How do we get the sequence of numbers to decode?

- Read the input as a single string, then split it apart into substrings, each of which represents one number.
Strings, Lists, and Sequences

- Strings are objects that have useful methods associated with them.
- One of these methods is `split`. This will split a string into substrings based on spaces.
- Example:
  ```python
  >>> "Hello string methods!".split()
  ['Hello', 'string', 'methods!']
  ```
  Split can be used on characters other than space, by supplying the character as a parameter.
How can we convert a string containing digits into a number?
Use `eval`.
The split function produces a sequence of strings. `numString` gets each successive substring.
Each time through the loop, the next substring is converted to the appropriate Unicode character and appended to the end of message.
Practice.
There are a number of other string methods:

- `s.capitalize()` - Copy of `s` with only the first character capitalized.
- `s.title()` - Copy of `s`; first character of each word capitalized.
- `s.center(width)` - Center `s` in a field of given width.
- `s.count(sub)` - Count the number of occurrences of `sub` in `s`.
- `s.find(sub)` - Find the first position where `sub` occurs in `s`.
- `s.join(list)` - Concatenate list of strings into one large string using `s` as separator.
- `s.ljust(width)` - Like center, but `s` is left-justified.
There are a number of other string methods:

- `s.lower()` - Copy of s in all lowercase letters
- `s.lstrip()` - Copy of s with leading whitespace removed
- `s.replace(oldsub, newsub)` - Replace occurrences of oldsub in s with newsub
- `s.rfind(sub)` - Like find, but returns the right-most position
- `s.rjust(width)` - Like center, but s is right-justified
- `s.rstrip()` - Copy of s with trailing whitespace removed
- `s.split()` - Split s into a list of substrings
- `s.upper()` - Copy of s; all characters converted to uppercase
The process of encoding information for the purpose of keeping it secret or transmitting it privately is called encryption.

Cryptography is the study of encryption methods.

Encryption is used when transmitting credit card and other personal information to a web site.

Strings are represented as a sort of encoding problem, where each character in the string is represented as a number that’s stored in the computer.

The code that is the mapping between character and number is an industry standard, so it’s not “secret”.
The encoding/decoding programs we wrote use a substitution cipher, where each character of the original message, known as the plaintext, is replaced by a corresponding symbol in the cipher alphabet.

The resulting code is known as the ciphertext.

This type of code is relatively easy to break.

Each letter is always encoded with the same symbol, so using statistical analysis on the frequency of the letters and trial and error, the original message can be determined.
Modern encryption converts messages into numbers.
Sophisticated mathematical formulas convert these numbers into new numbers - usually this transformation consists of combining the message with another value called the “key”
To decrypt the message, the receiving end needs an appropriate key so the encoding can be reversed.
In a private key system the same key is used for encrypting and decrypting messages. Everyone you know would need a copy of this key to communicate with you, but it needs to be kept a secret.

In public key encryption, there are separate keys for encrypting and decrypting the message.

In public key systems, the encryption key is made publicly available, while the decryption key is kept private.

Anyone with the public key can send a message, but only the person who holds the private key (decryption key) can decrypt it.
Often we will need to do some string operations to prepare our string data for output.

Let’s say we want to enter a date in the format “05/24/2003” and output “May 24, 2003”.

How could we do that?

- Input the date in mm/dd/yyyy format (dateStr).
- Split dateStr into month, day, and year strings.
- Convert the month string into a month number.
- Use the month number to lookup the month name.
- Create a new date string in the form “Month Day, Year”.
- Output the new date string.

Practice.
Sometimes we want to convert a number into a string.

We can use the `str` function.

If value is a string, we can concatenate a period onto the end of it.

If value is an int, what happens?
String Formatting

- String formatting is an easy way to get beautiful output.
- Practice with a change counter...

  Change Counter

  Please enter the count of each coin type.
  Quarters: 6
  Dimes: 0
  Nickels: 0
  Pennies: 0

  The total value of your change is 1.5

- Wouldn’t we like it to say $1.50?
We can format our output by modifying the print statement as follows:

```python
print("The total value of your change is \${0:.2f}".format(total))
```

Now we get something like: The total value of your change is $1.50.

```python
<template-string>.format(<values>)
```

The {} within the template-string mark “slots” into which the values are inserted.

Each slot has a description that includes a format specifier telling Python how the value for the slot should appear.
The formatting specifier has the form: `<width>.<precision><type>`.  

- `f` means “fixed point” number. 
- `<width>` tells us how many spaces to use to display the value. 0 means to use as much space as necessary. 
- `<precision>` is the number of decimal places.
String Formatting

- >>> “Hello {0} {1}, you may have won $2”
  .format(“Mr.”, “Smith”, 10000)
  ‘Hello Mr. Smith, you may have won $10000’

- >>> ‘This int, {0:5}, was placed in a field of width 5’.format(7)
  ‘This int,    7, was placed in a field of width 5’
A file is a sequence of data that is stored in secondary memory.

Files can contain any data type, but the easiest to work with are text.

A file usually contains more than one line of text.

Python uses the standard newline character (\n) to mark line breaks.
Hello
World

Goodbye

Stored in a file as:
Hello\nWorld\n\n\n\nGoodbye\n
The process of opening a file involves associating a file on disk with an object in memory.

We can manipulate the file by manipulating this object.

- Read from the file
- Write to the file

When done with the file, it needs to be closed. Closing the file causes any outstanding operations and other bookkeeping for the file to be completed.

In some cases, not properly closing a file could result in data loss.
Files Processing

- **Reading a file into a word processor:**
  - File opened.
  - Contents read into RAM
  - File closed
  - Changes to the file are made to the copy stored in memory, not on the disk.

- **Saving a word processing file:**
  - The original file on the disk is reopened in a mode that will allow writing (this actually erases the old contents)
  - File writing operations copy the version of the document in memory to the disk
  - The file is closed
Working with text files in Python:

- Associate a disk file with a file object using the open function
  \[ \text{filevar} = \text{open}(\text{name}, \text{mode}) \]
- Name is a string with the actual file name on the disk. The mode is either ‘r’ or ‘w’ depending on whether we are reading or writing the file.
- \( \text{Infile} = \text{open}("\text{numbers.dat"}, \ "r") \)
Files Methods

- `<file>.read()` - returns the entire remaining contents of the file as a single (possibly large, multi-line) string.
- `<file>.readline()` - returns the next line of the file. This is all text up to and including the next newline character.
- `<file>.readlines()` - returns a list of the remaining lines in the file. Each list item is a single line including the newline characters.
- Practice