

Python

Text



Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See http://software-carpentry.org/license.html for more information.





American English in the 1960s:

Python Text



American English in the 1960s:

26 characters × {upper, lower}



American English in the 1960s:

26 characters × {upper, lower}

+ 10 digits



American English in the 1960s:

26 characters × {upper, lower}

- + 10 digits
- + punctuation



American English in the 1960s:

26 characters × {upper, lower}

- + 10 digits
- + punctuation
- + special characters for controlling teletypes (new line, carriage return, form feed, bell, ...)



American English in the 1960s:

26 characters × {upper, lower}

- + 10 digits
- + punctuation
- + special characters for controlling teletypes (new line, carriage return, form feed, bell, ...)
- = 7 bits per character (ASCII standard)





1. Fixed-width records



1. Fixed-width records

A crash reduces your expensive computer to a simple stone.



1. Fixed-width records

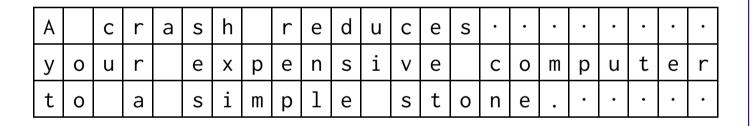
A crash reduces your expensive computer to a simple stone.

| Α | | С | r | а | S | h | | r | е | d | u | С | е | S | • | • | • | • | • | • | • | • |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| У | 0 | a | ٢ | | е | X | р | e | n | S | i | > | е | | C | 0 | m | р | u | t | е | r |
| t | 0 | | а | | S | i | m | р | 1 | е | | S | t | 0 | n | е | • | • | • | • | • | • |



1. Fixed-width records

A crash reduces your expensive computer to a simple stone.



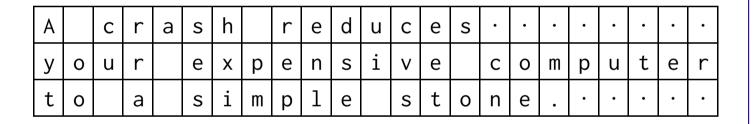
Easy to get to line N

Python Text



1. Fixed-width records

A crash reduces your expensive computer to a simple stone.

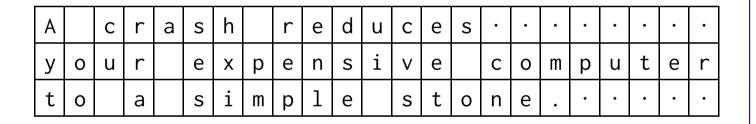


Easy to get to line N
But may waste space



1. Fixed-width records

A crash reduces your expensive computer to a simple stone.



Easy to get to line N

But may waste space

What if lines are longer than the record length?



- 1. Fixed-width records
- 2. Stream with embedded end-of-line markers



- 1. Fixed-width records
- 2. Stream with embedded end-of-line markers

A crash reduces your expensive computer to a simple stone.

| Α | С | r | а | S | h | | r | е | d | u | С | е | S | У | 0 | u | r | | е | X | р | е | n | S | i | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| е | С | 0 | m | р | u | t | е | r | | t | 0 | | а | S | i | m | р | 1 | е | | S | t | 0 | n | е | |



- 1. Fixed-width records
- 2. Stream with embedded end-of-line markers

A crash reduces your expensive computer to a simple stone.

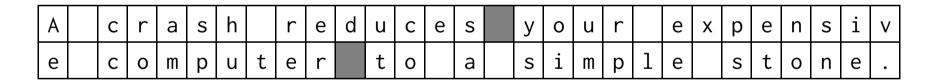


More flexible



- 1. Fixed-width records
- 2. Stream with embedded end-of-line markers

A crash reduces your expensive computer to a simple stone.



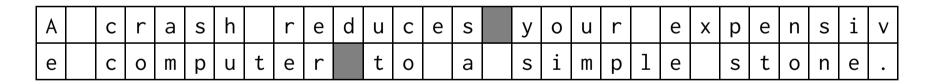
More flexible

Wastes less space



- 1. Fixed-width records
- 2. Stream with embedded end-of-line markers

A crash reduces your expensive computer to a simple stone.



More flexible

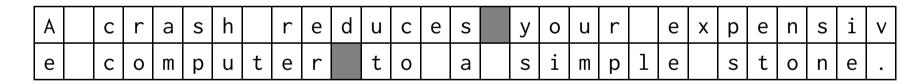
Skipping ahead is harder

Wastes less space



- 1. Fixed-width records
- 2. Stream with embedded end-of-line markers

A crash reduces your expensive computer to a simple stone.



More flexible

Skipping ahead is harder

Wastes less space

What to use for end of line?





Windows: carriage return + newline ('\r\n')

Python Text



Windows: carriage return + newline ('\r\n')

Oh dear...



Windows: carriage return + newline ('\r\n')

Oh dear...

Python converts '\r\n' to '\n' and back on Windows



Windows: carriage return + newline ('\r\n')

Oh dear...

Python converts '\r\n' to '\n' and back on Windows

To prevent this (e.g., when reading image files)

open the file in *binary mode*



Unix: newline ('\n') Windows: carriage return + newline ('\r\n') Oh dear... Python converts '\r\n' to '\n' and back on Windows To prevent this (e.g., when reading image files) open the file in *binary mode* reader = open('mydata.dat', 'rb')





How to represent ĕ, β, Я, ...?



How to represent ĕ, β, Я, ...?

7 bits = 0...127

Python Text



How to represent ĕ, β, Я, ...?

7 bits = 0...127

8 bits (a byte) = 0...255



How to represent ĕ, β, Я, ...?

7 bits = 0...127

8 bits (a byte) = 0...255

Different companies/countries defined different

meanings for 128...255



How to represent ĕ, β, Я, ...?

7 bits = 0...127

8 bits (a byte) = 0...255

Different companies/countries defined different

meanings for 128...255

Did not play nicely together



How to represent ĕ, β, Я, ...?

7 bits = 0...127

8 bits (a byte) = 0...255

Different companies/countries defined different

meanings for 128...255

Did not play nicely together

And East Asian "characters" won't fit in 8 bits



1990s: Unicode standard



1990s: Unicode standard

Defines mapping from characters to integers



Defines mapping from characters to integers

Does *not* specify how to store those integers



Defines mapping from characters to integers

Does *not* specify how to store those integers

32 bits per character will do it...



Defines mapping from characters to integers

Does *not* specify how to store those integers

32 bits per character will do it...

...but wastes a lot of space in common cases



Defines mapping from characters to integers

Does *not* specify how to store those integers

32 bits per character will do it...

...but wastes a lot of space in common cases

Use in memory (for speed)

Python Text



Defines mapping from characters to integers

Does *not* specify how to store those integers

32 bits per character will do it...

...but wastes a lot of space in common cases

Use in memory (for speed)

Use something else on disk and over the wire

Python Text





First 128 characters (old ASCII) stored in 1 byte each



First 128 characters (old ASCII) stored in 1 byte each Next 1920 stored in 2 bytes, etc.



First 128 characters (old ASCII) stored in 1 byte each Next 1920 stored in 2 bytes, etc.

0xxxxxxx 7 bits



First 128 characters (old ASCII) stored in 1 byte each Next 1920 stored in 2 bytes, etc.

| | 0xxxxxxx | 7 bits |
|----------|----------|---------|
| 110yyyyy | 10xxxxxx | 11 bits |



First 128 characters (old ASCII) stored in 1 byte each Next 1920 stored in 2 bytes, etc.

| | | 0xxxxxxx | 7 bits |
|----------|----------|----------|---------|
| | 110ууууу | 10xxxxxx | 11 bits |
| 1110zzzz | 10уууууу | 10xxxxxx | 16 bits |



First 128 characters (old ASCII) stored in 1 byte each Next 1920 stored in 2 bytes, etc.

| | | | 0xxxxxxx | 7 bits |
|----------|----------|----------|----------|---------|
| | | 110ууууу | 10xxxxxx | 11 bits |
| | 1110zzzz | 10уууууу | 10xxxxxx | 16 bits |
| 11110www | 10zzzzzz | 10уууууу | 10xxxxxx | 21 bits |



First 128 characters (old ASCII) stored in 1 byte each Next 1920 stored in 2 bytes, etc.

| | | | 0xxxxxxx | 7 bits |
|----------|----------|----------|----------|---------|
| | | 110ууууу | 10xxxxxx | 11 bits |
| | 1110zzzz | 10уууууу | 10xxxxxx | 16 bits |
| 11110www | 10zzzzzz | 10уууууу | 10xxxxxx | 21 bits |

The good news is, you don't need to know





Python 2.* provides two kinds of string Classic: one byte per character



Classic: one byte per character

Unicode: "big enough" per character



Classic: one byte per character

Unicode: "big enough" per character

Write u'the string' for Unicode



Classic: one byte per character

Unicode: "big enough" per character

Write u'the string' for Unicode

Must specify *encoding* when converting from

Unicode to bytes



Classic: one byte per character

Unicode: "big enough" per character

Write u'the string' for Unicode

Must specify *encoding* when converting from

Unicode to bytes

Use UTF-8



created by

Greg Wilson

October 2010



Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.