Chapter 5: Sequences Part 2: Lists and Encryption

Jan 30, 2020

Today’s Outline

- Review:
  - Strings
  - Lists
- Unicode
- Encryption

Sequences

- A string is a sequence of characters.

```
months = "JanFebMarAprMayJunJulAugSepOctNovDec"
```

- A list is a sequence of any type of object.

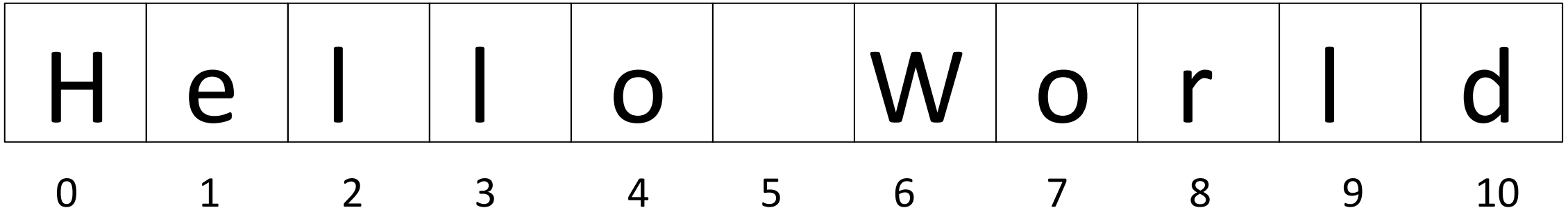
```
list_example = ["Jan", "Feb", 3, -5.6, "March"]
```

Strings vs. Lists

1) Lists can contain any type of objects. Strings are always sequences of characters.

2) Lists are mutable, but strings are not.

String Indexing



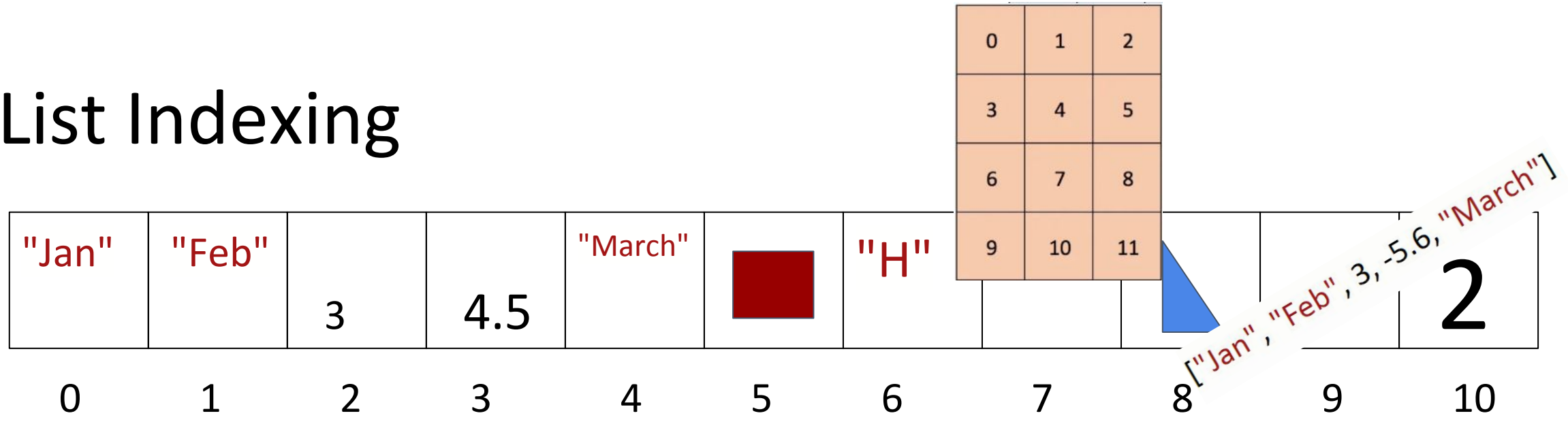
The characters in a string can be accessed individually.

Note:

The string indexing begins at 0.

The space counts as a character in the string.

List Indexing



The objects in a string can be accessed individually.

Note:

The indexing begins at 0.

String Slicing Month Example

months = "JanFebMarAprMayJunJulAugSepOctNovDec"

Jan: 1 Pos: 0

Feb: 2 Pos: 3

Mar: 3 Pos: 6

Month String Example

```
index = eval(input("What is the month number? "))
months = "JanFebMarAprMayJunJulAugSepOctNovDec"


start_pos = (index-1)*3
end_pos = start_pos+3
print(months[start_pos:end_pos])
```

Month Example: Redo with a list

Write a program that prints the abbreviation of a month given an INT from 1-12 that corresponds to the month.

```
months = ["Jan", "Feb", "Mar", "Apr", "May",
          "Jun", "Jul","Aug","Sep","Oct",
          "Nov","Dec"]
```

Month List Example

```
index = eval(input("What is the month number? "))
months = ["Jan", "February", "Mar", "Apr", "May",
          "Jun", "Jul","Aug","Sep","Oct",
          "Nov","Dec"]

print(months[index-1])
```

String Methods

function	meaning
`s.capitalize()`	Copy of `s` with only the first character capitalized.
`s.center(width)`	Copy of `s` centered 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` into a string, using `s` as separator.
`s.ljust(width)`	Like `center`, but `s` is left-justified.
`s.lower()`	Copy of `s` in all lowercase characters.
`s.lstrip()`	Copy of `s` with leading white space removed.
`s.replace(oldsub,newsub)`	Replace all occurrences of `oldsub` in `s` with `newsub`.
`s.rfind(sub)`	Like `find`, but returns the rightmost position.
`s.rjust(width)`	Like `center`, but `s` is right-justified.
`s.rstrip()`	Copy of `s` with trailing white space removed.
`s.split()`	Split `s` into a list of substrings (see text).
`s.title()`	Copy of `s` with first character of each word capitalized.
`s.upper()`	Copy of `s` with all characters converted to uppercase.

Table 5.2: Some string methods

Journal Writing

Write a program that converts a journal paper title into title capitalization.

ex. ***Outlier detection in high-density surface electromyographic signals***

should become:

Outlier Detection In High-density Surface Electromyographic Signals

The split() method

A powerful string method is the split() method. The split() method converts a string into a list. By default, the words are split wherever there is a space.

```
myString.split( )
```

```
wordList = "I am writing a list of words".split()
```

```
numberList = "32,24,25,57".split (",")
```

split () is useful for getting user information

Example: Obtain x and y coordinates from the user.

```
coords = input("Enter the coordinates in the format (x,y): ").split(",")
x_coord = float(coords[0])
y_coord = float(coords[1])
print(x_coord, y_coord)
```

Acronym

An acronym is a word formed by taking the first letters of the words in a phrase and making a word from them.

For example, RAM is an acronym for "random access memory."

Write a program that allows the user to type in a phrase and then outputs the acronym for that phrase.

Pig Latin

Pig Latin is a made-up language similar to English. To create a word in Pig Latin, take the first letter of a word, and place it at the end of the word with the suffix ‘ay’.

pig latin fun = igpay atinlay unfay

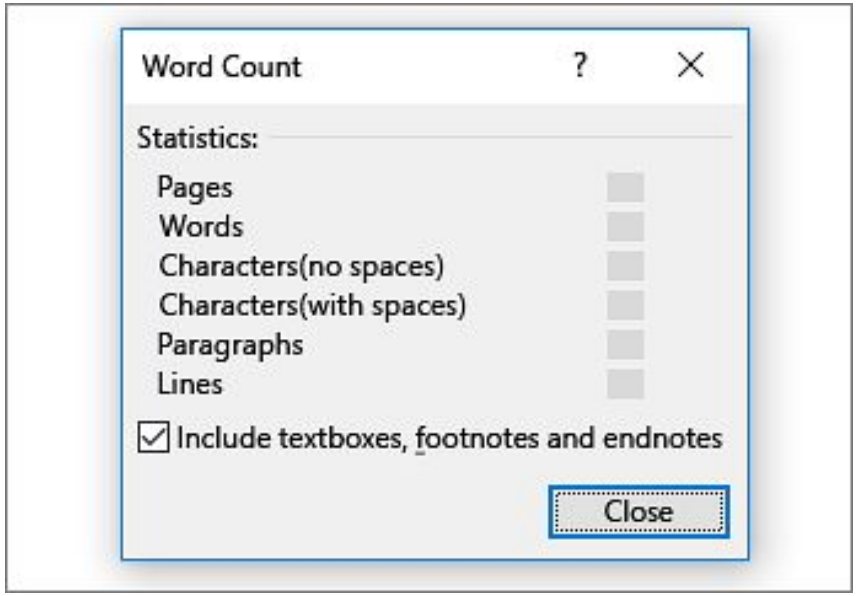
Write a program that converts a sentence into Pig Latin.

Word Counts

Write a program that calculates the number of words in a sentence entered by a user.

Write a program that determines the average word length in a sentence entered by the user.

** Hint: the len() function can be used to find the length of a list or of a string.



List Method: append()

```
#adding new items to a list
squares = []
for x in range (1,101):
  squares.append (x*x)
```

ASCII

ASCII (American Standard Code for Information Interchange). ASCII uses the numbers 0 through 127 to represent the characters typically found on an (American) computer keyboard.

Capital letters A-Z are represented by the values 65-90

Lowercase letters a-z have values 97- 122.

Unicode

Unicode uses the same codes as ASCII for the 127 characters originally defined, but aims to support characters for all of the written languages.

Unicode contains a repertoire of over 137,000 characters covering 150 modern and historic scripts, and emoji.

ASCII vs. Unicode

ASCII is a 7 bit code that can represent 2^7 = 128 characters

Unicode has about 1,112,000 valid code points. How many bits do we need to encode it?

Unicode

The most common encoding system for unicode is a variable length encoding system called UTF-8. 95% of webpages use UTF-8.

- ASCII characters are stored using one byte of data (UTF-8 has backwards compatibility with ASCII)
- More rare characters are stored using up to 4 bytes of data

Unicode Characters in Python

The ord() function is used to find the unicode value of a character.

```
ord ("a")
ord("9")
ord("!")
ord("π")
```

Unicode Characters in Python

The chr() function is used to find the character for a unicode value.

```
chr(90)
chr(50)
chr(128525)
print(chr(30005),chr(33041), sep = "")
```

Encoding

Let’s write a program that converts a sentence into Unicode.

Example:

Strings are fun!

will output:

83 116 114 105 110 103 115 32 97 114 101 32 102 117 110 33

Decoding

Write a program that converts Unicode into text.

Encryption

Encryption is the process of encoding information to keep it secret.

The encoding/decoding examples that we did with unicode are essentially **substitution ciphers**. In a substitution cipher, each character of the original message is replaced with a different symbol.

i.e. a = 97, b = 98, … z = 122

Alphabet Problem

Write a program that prompts the user for a letter, and then returns the position of that letter in the alphabet.

Ex: A = 1

B = 2

Z = 26

Hint: ord ("A") = 65

Substitution Cipher

In a simple substitution cipher, each letter is substituted for a different letter.

Ex: Atbash Cipher

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ

ZYXWVUTSRQPONMLKJIHGFEDCBA
```

ABCDEFGHIJKLMNOPQRSTUVWXYZ

ZYXWVUTSRQPONMLKJIHGFEDCBA

What does this message say?

ZGGZXPZGWZDM

Atbash Cipher

Design a program that can encrypt a message into Atbash Cipher.

Atbash Cipher

Design a program that can encrypt a message into Atbash Cipher.

Hint1 : What should the lookup table be?

Hint 2: What is the unicode value of A?

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ

ZYXWVUTSRQPONMLKJIHGFEDCBA
```

Atbash Cipher

Design a program that can encrypt a message into Atbash Cipher.

Design a program that can decrypt the message.

Lab 3 Announcement

The graphics.py module probably won't work using Jupyter notebooks or Google Colab.

I recommend coming to lab to use Anaconda Spyder for this lab. I will try to make the assignment shorter than the previous ones so that you can finish it in the lab. If you need additional time, please let me know.