Chapter 5: Sequences Part 3: Files

Feb 4, 2020

Today's Outline

- Review:
  - Encryption
  - Quiz 4 Review
- String Formatting
- Files
- Announcements

Strings and Lists

```
#pig latin
english = input("Type a sentence ").split()
for word in english:
  print(word[1:len(word)]+word[0]+'ay',end=" ")
```

Unicode Characters in Python

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

```
ord ("A")
```

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

```
chr(65)
```

Substitution Cipher

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

Ex: Atbash Cipher

ABCDEFGHIJKLMNOPQRSTUVWXYZ

ZYXWVUTSRQPONMLKJIHGFEDCBA

Atbash Cipher

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

```
message = input("Type a message: ").upper().replace(" ","")
for ch in message:
  number = 91-(ord(ch)-64)
  print(chr(number),end="")
```

Quiz 4 Review

pollev.com/itec5920w

Quiz 4 will be due this Sunday, Feb 9

How do you access the letter e in the string s = "the"?

a) s["e"]

b) s[3]

c) s[2]

d) s[e]

How do you determine the number of characters in the string s?

a) s.len()

b) len(s)

c) s.length()

d) length(s)

What is the result of the Python expression "3" + "4"?

a) "7"

b) 7

c) "34"

d) 34

What does the Python expression 3 * "A" evaluate to?

a) It produces an error

b) "3A3A3A"

c) "3A"

d) "AAA"

What would the following Python program output?

```
for ch in "the quick fox":
    print(ch)
```

a) "the quick fox" on one line

b) output each of the characters (excluding spaces) in the string with one character per line

c) output "the" on the first line "quick" on the second line and "fox" on the third line

d) none of the above

What data type does the string's split() method return?

a) int

b) list

c) string

d) none of the above

What would the following Python program output?

```
text = "the quick fox".split()
for ch in text:
    print(ch)
```

a) "the quick fox" on one line

b) output "the" on the first line "quick" on the second line and "fox" on the third line

c) output each character in the string, excluding the spaces, with one character per line

d) none of the above

What function gives the Unicode value of a character?

a) ord()

b) uni()

c) chr()

d) ascii()

What can't be contained in a list?

a) arrays

b) lists

c) graphics objects

d) none of the above

How do I change the entry in the list myList to the correct spelling?

myList = ["blue", "yellowww", "orange"]

a) myList[2] = "yellow"

b) myList[1] = "yellow"

c) mylist("yellow",2)

d) none of the above

format() method

Another powerful string method is the format method(). It allows parameters to be inserted into slots specified by curly brackets { }

```
"String {0} template {1} ".format(value1, value2)
```

```
greeting = "Hello {0}, nice to {1} you!".format("Bob", "meet")
coins = "You have {0} dimes and {1} quarters".format(5,10)
```

format() with decimal places

The number of decimal places to include for a number can be indicated within the curly bracket.

```
import math
print("The value of pi is {0:0.2}".format(math.pi))
print("The value of pi is {0:0.5}".format(math.pi))
```

format() with decimal places: fixed point

Using "f" indicates that the value will be a fixed point number. The specified number of decimal places will always be shown, even if they are 0.

```
print("You have ${0:0.2}".format(5.2))
print("You have ${0:0.2f}".format(5.2))
```

format() and spacing

The width indicates how many spaces a value will take up.

```
print("The value of pi is {0:10.5} and the value of e is
{1:10.5}".format(math.pi, math.e))
```

This feature is useful for making tables.

Format justification

The following symbols are used to indicate the justification:

left: <

right : >

centre: ^

```
print("left justification:{0:<10}".format("Hi!"))
print("right justification:{0:>10}".format("Hi!"))
print("centre justification:{0:^10}".format("Hi!"))
```

Format() specifier summary

{index:[justification]width.#decimal places}

{0:<10.5}

{3:^0.2f}

Simple Examples

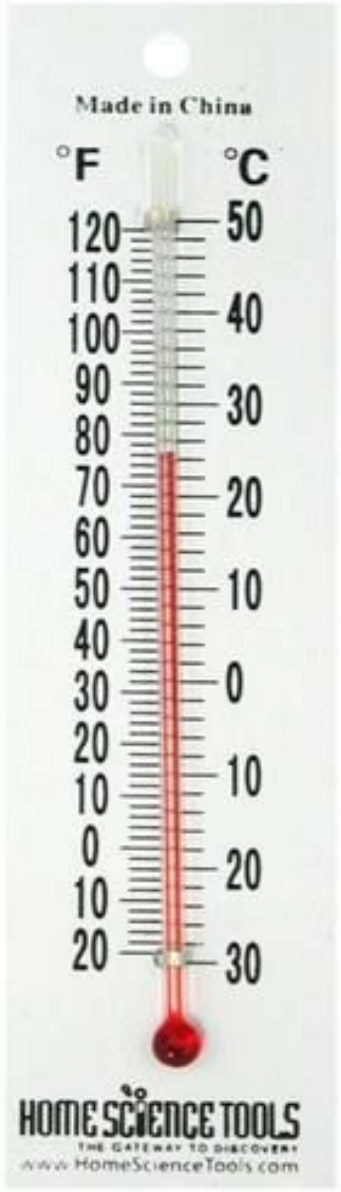
```
"I will eat {1} and {0} for breakfast.".format ("eggs", "toast")

"Hello {1}".format ("World","Universe","Galaxy")

"{0:0.2f} {1:0.2f}".format(2.3, 2.3468)
```

```
def tempConvert (celsius):
  fahrenheit = celsius*9/5 + 32
  return fahrenheit


def main():
  print('Celsius','Fahrenheit')
  for temp_c in range(0,110,10):
    temp_f = tempConvert(temp_c)
    print(temp_c, temp_f)
main()
```



Problem: Use the .format() function to make the temperature table more beautiful

```
Celsius Fahrenheit
0 32.0
10 50.0
20 68.0
30 86.0
40 104.0
50 122.0
60 140.0
70 158.0
80 176.0
90 194.0
100 212.0
```

Celsius	Fahrenheit
0	32.0
10	50.0
20	68.0
30	86.0
40	104.0
50	122.0
60	140.0
70	158.0
80	176.0
90	194.0
100	212.0

Files in Python

A file is a sequence of memory that is stored in secondary memory (ex. hard drive).

Conceptually a **text file** is a very long string that is stored in secondary memory.

Files in Python

Conceptually a **text file** is a very long string.

For example, the text:

Hello World

It is February 4 today.

would be stored in a file as:

Hello World\nIt is February 4 today.

File Processing

To use a file, we need to associate the file with an **object** in our program. This is known as “opening” the file in our program.

In Python, the open () function is used to create file objects.

Opening reading files in Python

```
infileName = "fileName"

#open the file for reading

infile = open (infileName, "r" )
```

Opening writing files in Python

```
outfileName = "fileName"

#open the file for writing

outfile = open (outfileName, "w" )
```

Reading lines from a text file

1: infile.read() - reads the entire file into a string

2: infile.readline() - reads the next line of a file

3: infile.readlines() - reads all of the remaining lines of a file into a list

File reading: infile.read() example

```
fileName = "poem.txt"
infile = open(fileName,'r')
poem = infile.read()
print(poem)
```

File reading: infile.readline() example

```
fileName = "poem.txt"
infile = open(fileName,'r')
for i in range(5):
    line = infile.readline()
    print(line)
```

File reading: infile.readlines() example

```
fileName = "poem.txt"
infile = open(fileName,'r')
lineList = infile.readlines()
print(lineList)
```

*** What is the problem with this?

** Recommended way to read lines

```
for line in infile:
    [operations to perform on each line]


fileName = "poem.txt"
infile = open(fileName,'r')
for line in infile:
    print(line)
infile.close()
```

Closing files

Once we have finished with the file, we can close it using the close() method.

Closing the file ensures that all of the changes made to the file object are saved in the file on the disk/hard drive.

Writing to a text file

Writing to a text file is performed using the print() function.

print ([item to print] , file=outputfile)

Poem Example

Print the first 5 lines of the poem into a new text file named **poem5.txt**

Appending to a file

To append to a file you can use: 'a'

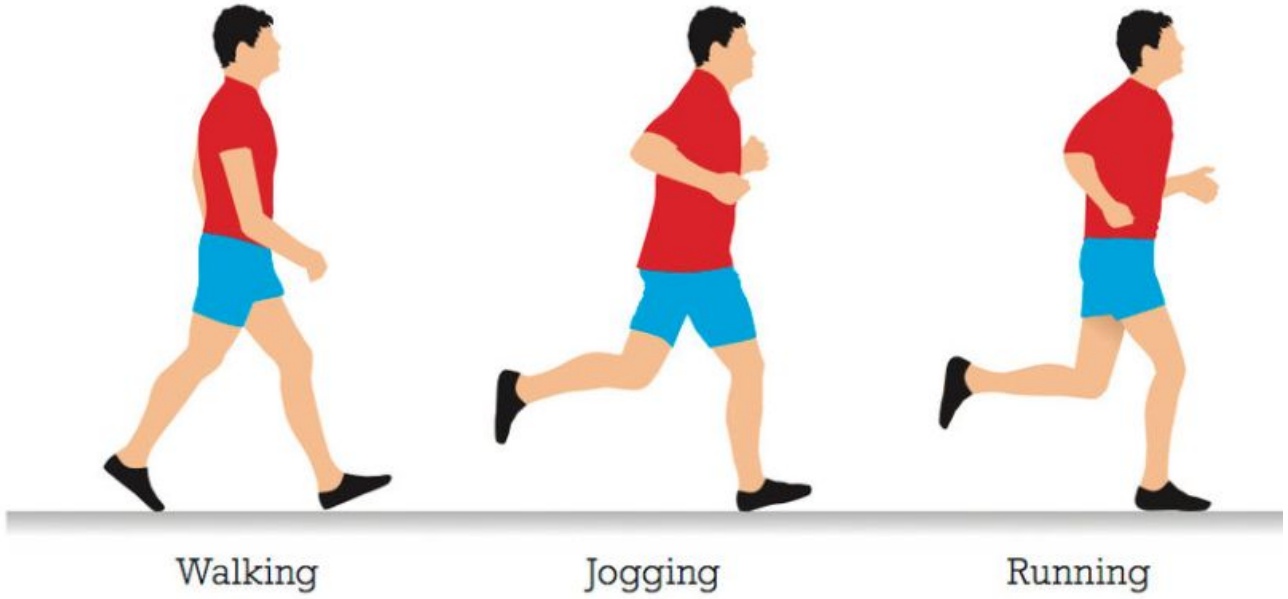
```
appendfile = open(fileName,'a')
```

Poem Example Part 2

Append the last 5 lines of the poem to **poem5.txt**

File example

I collected heart data from 20 subjects. They performed three trials each of of walking, jogging, and running.



I have 20x3x3 = 180 files to process and I don’t want to have to individually upload them into Python.

File Names

The files are saved with this format:

Subject{subject_number}_{activity}_Trial{trial_number}

ex. Subject1_Walk_Trial1
Subject1_Walk_Trial2
Subject1_Walk_Trial3
Subject1_Run_Trial1

Process many files

```
for subject in range (1,11):
    for trial in range (1,4):
        for action in ["walk", "run", "climb"]:
            fileName = "Subject{0}_{1}_Trial{2}.txt".format(subject, action, trial)
            infile = open(fileName,'r')

            [code to process each file]

            infile.close()
```

Directory Problems

Python always looks in the current directory (directory the program is saved in) to find the files.

To use a file from a different directory, you will have to indicate the entire file name:

```
fileName = "C:\Users\emmafarago\Documents\Programming Examples\poem.txt"
```

Tkinter Library: File Dialogues

```
from tkinter.filedialog import askopenfilename
infileName = askopenfilename ( )

outfileName = asksavefilename ( )
```

Other types of Files

When reading and writing to other types of files, you may need to import other libraries

Excel: Pandas library

https://nbviewer.jupyter.org/urls/bitbucket.org/hrojas/learn-pandas/raw/master/lessons/01%20-%20Lesson.ipynb

File Quality

(A)

(B)

(C)

(D)

			pre-treatment						Post treatment				
Weeks	Treatment	Individual	Bold1	Bold2	Bold3	Bold4	Bold5	Bold6	Bold7	Bold8	Bold9	Bold10
4	Mixed	M_4_10	297.91	356.56	600	160.66	600	160.66	600	119.83	153.53	241.34
4	Mixed	M_4_11	600	600	600	77.56	313.32	477.56	313.32	384.26	339.12	357.55
4	Mixed	M_4_12	600	600	600	335.09	212.22	335.09	412.22	414.03	415.46	399.41
4	Mixed	M_4_13	103.5	442.25	241.34	339.63	119.72	139.63	119.72	210.73	113.47	111.31
4	Mixed	M_4_14	600	600	600	178.41	68.57	463.77	441.23	420.89	369.81	402.63
4	Mixed	M_4_15	561.71	600	600	257.62	124.72	257.62	124.72	600	600	600
4	Mixed	M_4_16	600	281.46	600	562.01	183.25	562.01	183.25	600	84.53	600
4	Mixed	M_4_17	600	600	259.94	600	452.7	492.23	452.7	499.45	594.72	458.71
4	Mixed	M_4_18	600	170.13	58.75	435.60	28.22	35.21	28.22	83.75	600	12.21
4	Mixed	M_4_19	600	90.56	43.48	600	276.63	58.75	233.51	358.66	502.37	63.77
4	Mixed	M_4_20	600	277.53	63.77	600	165.58	600	165.58	46.35	334.00	600
4	Mixed	M_4_21	600	600	107.37	349.00	327.22	349.00	327.22	600	408.28	288.32
4	Mixed	M_4_22	264.59	600	583.78	600	209.16	482.35	409.16	484.53	495.62	317.35
4	Mixed	M_4_23	600	600	475.82	600	419.38	600	600	600	600	600
4	Mixed	M_4_24	35.72	137.59	100.26	113.47	312.17	13.54	19.5	53.97	66.21	58.75
4	Mixed	M_4_25	403.03	234.66	554.88	31.72	76.66	31.72	76.66	42.13	39.53	54.23
4	Mixed	M_4_26	83.00	170.56	600	518.31	600	518.31	600	305.85	498.31	554.88
4	Mixed	M_4_27	600	600	305.85	153.53	358.66	453.53	358.66	452.65	358.94	359.94
4	Mixed	M_4_28	600	600	600	91.91	590.41	591.91	590.41	500.21	454.23	443.37
4	Mixed	M_4_29	35.06	114.27	404.35	600	600	600	155.42	119.25	600	598.62
4	Mixed	M_4_30	252.47	600	600	262.83	194.62	262.83	194.62	24.56	530.35	171.75
4	Mixed	M_4_31	600	600	143.37	404.35	233.00	404.35	233.00	61.2	252.72	114.28
4	Mixed	M_4_32	110.53	600	600	600	600	495.56	577.45	585.56	497.56	583.78
4	Mixed	M_4_33	581.03	600	600	600	600	600	600	600	600	600
4	Mixed	M_4_34	23.06	600	183.28	552.25	181.06	552.25	181.06	114.04	188.09	214.94
4	Mixed	M_4_35	600	337.74	171.75	184.65	452.09	184.65	452.09	107.28	518.31	600
4	Mixed	M_4_36	600	600	598.62	237.40	91.79	237.40	91.79	600	383.22	199.38
4	Mixed	M_4_37	600	449.44	600	285.47	262.85	285.47	262.85	600	600	143.37
4	Mixed	M_4_38	251.06	600	600	600	198.57	512.23	398.57	477.00	482.09	482.34
4	Mixed	M_4_39	42.21	600	96.00	600	191.54	600	232.1	600	600	600
4	Mixed	M_4_40	471.34	600	146.53	201.83	260.09	201.83	260.09	471.75	72.19	100.26
4	Mixed	M_4_41	600	403.25	317.35	298.59	191.54	298.59	191.54	600	344.09	600
4	Mixed	M_4_42	600	84.53	600	600	348.88	600	348.88	600	441.94	182.09
4	Mixed	M_4_43	600	600	600	94.72	570.93	594.72	570.93	499.87	525.48	541.02

https://laskowskilab.faculty.ucdavis.edu/2020/01/29/retractions/

Project

The link to the project proposal submission is now active.

Think about:

- What do I want to learn about?
- What project would be useful for my CV/portfolio?

Labs

I will post my solutions to the labs.

Challenge: Try to only use the concepts that we learned in class to solve the problems.

Pig Latin Bonus

Napoleon the pig is very clever and wants to read War and Peace.

Make a program that translates War and Peace into Pig Latin and saves it into a .txt file so that he can read the book.