

# **DATA 520**

## **Lecture 13**

### **Reading and Writing Files**

**Storing Data**

**Exams**

# Open Office (Apache) or Libre Office

## Application Monday

MS Office freeware substitute (word processor, spreadsheet, slide creator, drawing, database programs)

rtf file:

Word = 105 KB

Open Office: 6 KB

Innovations in Open Office that Microsoft has adopted:

- first introduced saving compressed files (now \*.docx)
- first had more informative questions at program close:

was: "Do you want to save changes?" (Yes, No)

now: "File has changed. What should I do with it?" (Save, Don't Save, Cancel)

# Editpad, Notepad++

## Application Monday

### Notepad substitutes

- read larger files
- search and replace more flexible
- can convert CR+LF <-> CR (Linux/Win/Mac)
- color-coded keywords for programming languages (Notepad++)



# Remember the lessons of the Chinese Zodiac!



**RABBIT**  
1939, 1951, 1963, 1975, 1987, 1999  
Luckiest of all signs, you are also talented and articulate. Affectionate, yet shy, you seek peace throughout your life. Marry a Sheep or Boar. Your opposite is the Cock.



**DRAGON**  
1940, 1952, 1964, 1976, 1988, 2000  
You are eccentric and your life complex. You have a very passionate nature and abundant health. Marry a Monkey or Rat late in life. Avoid the Dog.



**SNAKE**  
1941, 1953, 1965, 1977, 1989, 2001  
Wise and intense with a tendency towards physical beauty. Vain and high tempered. The Boar is your enemy. The Cock or Ox are your best signs.



**HORSE**  
1942, 1954, 1966, 1978, 1990, 2002  
Popular and attractive to the opposite sex. You are often ostentatious and impatient. You need people. Marry a Tiger or a Dog early, but never a Rat.



**SHEEP**  
1943, 1955, 1967, 1979, 1991, 2003  
Elegant and creative, you are timid and prefer anonymity. You are most compatible with Boars and Rabbits but never the Ox.



**MONKEY**  
1944, 1956, 1968, 1980, 1992, 2004  
You are very intelligent and are able to influence people. An enthusiastic achiever, you are easily discouraged and confused. Avoid Tigers. Seek a Dragon or a Rat.



**TIGER**  
1938, 1950, 1962, 1974, 1986, 1998  
Tiger people are aggressive, courageous, candid and sensitive. Look to the Horse and Dog for happiness. Beware of the Monkey.



**OX**  
1937, 1949, 1961, 1973, 1985, 1997  
Bright, patient and inspiring to others. You can be happy by yourself, yet make an outstanding parent. Marry a Snake or Cock. The Sheep will bring trouble.



**RAT**  
1936, 1948, 1960, 1972, 1984, 1996  
You are ambitious yet honest. Prone to spend freely. Seldom make lasting friendships. Most compatible with Dragons and Monkeys. Least compatible with Horses.



**COCK**  
1945, 1957, 1969, 1981, 1993, 2005  
A pioneer in spirit, you are devoted to work and quest after knowledge. You are selfish and eccentric. Rabbits are trouble. Snakes and Oxen are fine.



**DOG**  
1946, 1958, 1970, 1982, 1994, 2006  
Loyal and honest you work well with others. Generous yet stubborn and often selfish. Look to the Horse or Tiger. Watch out for Dragons.



**BOAR**  
1947, 1959, 1971, 1983, 1995, 2007  
Noble and chivalrous. Your friends will be lifelong, yet you are prone to marital strife. Avoid other Boars. Marry a Rabbit or a Sheep.

**CHINESE ZODIAC**

The Chinese Zodiac consists of a 12 year cycle, each year of which is named after a different animal that imparts distinct characteristics to its year. Many Chinese believe that the year of a person's birth is the primary factor in determining that person's personality traits, physical and mental attributes and degree of success and happiness throughout his lifetime. To learn about your Animal Sign, find the year of your birth among the 12 signs running around the border. If born before 1936, add 12 to the year you were born to find your year.



# Break down a problem into steps

1. Start with what we know and what we want
  - 12 year cycle with symbolic animals and personality traits
  - based on birth year
2. Remember your programming tools
  - list:** 12 animals and personality traits
  - input** prompt: year of birth (while checks for years < 1800, > 2016)
  - if** branching: choose list index based on modulo (remainder)
  - while** loop: accept another year of birth

# Break down a problem into steps

Build a doghouse





# Break down a problem into steps

**Build a doghouse like these**

- materials
- tools



# doctest

## Using doctest (built-in)

```
# getmedian.py
```

```
def median(pool):
```

```
    '''Statistical median to demonstrate doctest.
```

```
>>> median([2, 9, 9, 7, 9, 2, 4, 5, 8]) # sorted 2,2,4,5,7,8,9,9,9
```

```
7
```

```
>>> median([2, 9, 9, 9, 2, 4, 5, 8])      # sorted 2,2,4,5,8,9,9,9
```

```
6.5
```

```
...
```

```
copy = sorted(pool)
```

```
nvals = len(copy)
```

```
if nvals % 2 == 1:
```

```
    return copy[(nvals - 1) // 2]
```

```
else:
```

```
    return (copy[nvals//2 + 1] + copy[nvals//2]) / 2
```

```
if __name__ == '__main__': # test function only when called alone
```

```
    import doctest
```

```
    doctest.testmod()
```

```
File "C:/Users/sousley/AppData/Local/Programs/Python/Python35-32/median.py"
```

```
Failed example:
```

```
    median([2, 9, 9, 9, 2, 4, 5, 8])
```

```
Expected:
```

```
    6.5
```

```
Got:
```

```
    8.5
```

```
*****
```

```
1 items had failures:
```

```
    1 of    2 in __main__.median
```

```
***Test Failed*** 1 failures.
```



# doctest

## Using doctest (correct)

```
# getmedian.py

def median(pool):
    '''Statistical median to demonstrate doctest.
    >>> median([2, 9, 9, 7, 9, 2, 4, 5, 8])
    7
    >>> median([2, 9, 9, 9, 2, 4, 5, 8])
    6.5
    ...

    copy = sorted(pool)
    nvals = len(copy)
    if nvals % 2 == 1:
        return copy[(nvals - 1) // 2]
    else:
        return (copy[nvals//2 - 1] + copy[nvals//2]) / 2  ## changed first + to - (minus)

if __name__ == '__main__':
    import doctest
    doctest.testmod(verbose=True)
```

# Opening and reading a file

Files are read from beginning to end, sequentially

- you can open a file in **read** (default), **write**, or **append** mode

```
file = open('file_example.txt', 'r') # open in read mode
contents = file.read() # assign all text to string (with \n)
print(contents)
file.close() # ALWAYS close open files!
```

^^^^ save as file\_reader.py

**create a text file `file_example.txt` in the same directory with contents:**

First line of text

Second line of text

Third line of text

# Opening and reading a file

Files are read from the beginning - where it is being read is called the cursor (placeholder)

- you can open with read (default), write, or append access

Another way: using **with**

```
with open('file_example.txt', 'r') as rfile: # open with read access
    contents = rfile.read()                #    ^^ variable name
```

**with** open(*«filename»*, *«mode»*) **as** *«variable»*:

*«block»*

\*\*\* This way automatically closes the file after code block is executed \*\*\*



# Opening and reading a file

Files are located in directories or paths or folders

- needed when not in same folder as program

## Linux

/usr/local/python

## Mac

/Users/pgries/Desktop/file\_example.txt

## Windows

C:\Users\Steve9\AppData\Local\Programs\Python\Python36-32\file\_example.txt

- but "\" is the escape character with special meaning, so use either:

C:\\Users\\Steve9\\AppData\\Local\\Programs\\Python\\Python36-32\\file\_example.txt

C:/Users/Steve9/AppData/Local/Programs/Python/Python36-32/file\_example.txt

# Opening and reading a file

Current working directory: default directory/path when looking for files  
to find default current working directory:

```
import os  
os.getcwd()
```

```
'C:\\Users\\Steve9\\AppData\\Local\\Programs\\Python\\Python36-32'
```

you can change current working directory:

```
os.chdir('D:\\Python_Experiments\\programs') # absolute path to program directory
```

```
open('data1.txt') # relative path (cwd) = D:\\Python_Experiments\\programs
```

```
open('data\\data1.txt') # relative path = D:\\Python_Experiments\\programs\\data
```

```
open('..\\data1.txt') # relative path = D:\\Python_Experiments\\
```

```
open('..\\data\\data1.txt') # relative path = D:\\Python_Experiments\\data
```

```
open('..\\..\\data1.txt') # relative path = D:\\
```

# Opening and reading a file

Files are read from the beginning

- where Python is reading at any one time is called the cursor (placeholder)

```
# read entire file into string
with open('file_example.txt', 'r') as rfile:
    contents = rfile.read()
print(contents)

# read first ten characters
with open('file_example.txt', 'r') as example_file:
    first_ten_chars = example_file.read(10)
    the_rest = example_file.read() # cursor moved to 11th character

print("The first 10 characters:", first_ten_chars)
print("The rest of the file:", the_rest)
```



# Opening and reading a file

## Reading lines

```
with open('file_example.txt', 'r') as example_file:
```

```
    lines = example_file.readlines()
```

```
print(lines)
```

```
['First line of text.\n', 'Second line of text.\n', 'Third line of text.\n']
```

- notice that it is a list of strings

## Save these lines to planets.txt

**Mercury**

**Venus**

**Earth**

**Mars**

# Opening and reading a file

## Reading lines and printing

```
>>> with open('planets.txt', 'r') as planets_file:
    planets = planets_file.readlines()

>>> planets

['Mercury\n', 'Venus\n', 'Earth\n', 'Mars\n']

# print each item in list, one line at a time, removing newlines (\n)

>>> for planet in planets:
    print(planet.strip())

Mercury

Venus

Earth

Mars

# then, one line at a time in reverse order

>>> for planet in reversed(planets): # reverse order of list
    print(planet.strip())

Mars

Earth

Venus

Mercury
```

# Opening and reading a file

## Reading lines and printing them sorted alphabetically

```
# in shell:

>>> planets

['Mercury\n', 'Venus\n', 'Earth\n', 'Mars\n']

# then, one line at a time sorted alphabetically

>>> for planet in sorted(planets): # sorted list
    print(planet.strip())

Earth

Mars

Mercury

Venus


*** original list has not changed ***

>>> planets

['Mercury\n', 'Venus\n', 'Earth\n', 'Mars\n']
```



# Opening and reading a file

**For Line in File:** reads every line in a file

Lines include the newline escape code, affect length

Lines: manageable chunks of text, stripping `\n` easiest by line

```
with open('planets.txt', 'r') as data_file:                                ['Mercury\n', 'Venus\n', 'Earth\n', 'Mars\n']
    for line in data_file: # it KNOWS what you want
        print(len(line))

8
6
6
5
```

The last is 'Mars', so `\n` is counted as a character

```
with open('planets.txt', 'r') as data_file:
    for line in data_file:
        print(len(line.strip()))
```

# Opening and reading a file

**Readline: reads one line in a file**

Many data are processed line by line

**Save as hopedale.txt**

Coloured fox fur production, HOPEDALE, Labrador, 1834-1842

# Source: C. Elton (1942) "Voles, Mice and Lemmings", Oxford Univ. Press

# Table 17, p.265--266

22

29

2

16

12

35

8

83

166

# Opening and reading a file

## Readline: reads one line in a file

```
#hopedale_sum.py

with open('hopedale.txt', 'r') as hopedale_file:

    # Read the description line. (first line) - do nothing with it
    hopedale_file.readline()

    # Keep reading comment lines until we read the first piece of data.
    data = hopedale_file.readline().strip()
    while data.startswith('#'):
        data = hopedale_file.readline().strip()

    # Now we have the first piece of data. Accumulate the total number of pelts.
    total_pelts = int(data)

    # Read the rest of the data.
    for data in hopedale_file:
        total_pelts = total_pelts + int(data.strip())

print("Total number of pelts:", total_pelts)
```

# Reading a web page

## We can read data from web pages

```
# load module
import urllib.request
url = 'http://robjhyndman.com/tsdldata/ecology1/hopedale.dat'
with urllib.request.urlopen(url) as webpage:
    for line in webpage:
        line = line.strip()
        line = line.decode('utf-8')
        print(line)
```



# Reading a web page

We can read data from web pages

- what if we don't decode them?

```
# load module
import urllib.request
url = 'http://robjhyndman.com/tsdldata/ecology1/hopedale.dat'
with urllib.request.urlopen(url) as webpage:
    for line in webpage:
        line = line.strip()
        # line = line.decode('utf-8')
        print(line)
```

# Writing to a file

## Create and write to a file

```
# create file for writing
```

```
with open('topics.txt', 'w') as output_file:  
    output_file.write('Computer Science')
```

(On office PC: wrote to: C:\Users\sousley\AppData\Local\Programs\Python\Python36-32)

- take a look, then modify code

```
# create file for writing? warn against overwriting?
```

```
with open('topics.txt', 'w') as output_file:  
    output_file.write('Computer Science 222')
```

## We overwrote the file

- we can also modify to get feedback:

```
def writef():  
    file = open('topics.txt', 'w')  
    return(file.write('Computer Science 111')) # will return the number of characters written  
    file.close()
```

# Appending to a file

## Append to a file

### - it must already exist

# add to a file

```
>>> with open('topics.txt', 'a') as output_file:  
    output_file.write('Software Engineering')
```

# best practice: end a line with a newline

```
>>> with open('topics.txt', 'w') as output_file:  
    output_file.write('Computer Science' + '\n')
```

```
>>> with open('topics.txt', 'a') as output_file:  
    output_file.write('Software Engineering' + '\n')
```

# Reading one file, appending to another file

(will not run)

```
# totalfilenumbers.py
```

```
def sum_number_pairs(input_file, output_filename):
```

```
    """ (file open for reading, str) -> NoneType
```

```
    Read the data from input_file, which contains two floats per line
```

```
    separated by a space. Open file named output_file and, for each line in
```

```
    input_file, write a line to the output file that contains the two floats
```

```
    from the corresponding line of input_file plus a space and the sum of the
```

```
    two floats.
```

```
    """
```

```
with open(output_filename, 'w') as output_file:
```

```
    for number_pair in input_file:
```

```
        number_pair = number_pair.strip()
```

```
        operands = number_pair.split()
```

```
        #return(str(operands))
```

```
        total = float(operands[0]) + float(operands[1])
```

```
        new_line = '{0} {1}'.format(number_pair, total)
```

```
        output_file.write(new_line)
```

```
# number_pairs.txt
```

```
1.3 3.4
```

```
2 4.2
```

```
-1 1
```



# Homework 10

Due before class Wednesday

Make it run!

`totalfilenumbers.py`