Intro to programming 6

Henri Vandendriessche
henri.vandendriessche@ens.fr

2022-10-25

Terminal cheat sheet reminder

- Bash commands to navigate directories
  - Print Working Directory. Print the path of the current directory

```
pwd
```

  - List all files of the current directory

```
ls folder
```

  - Moving into folder1 and subfolder2 at once.

```
cd folder1/subfolder2
```

  - Moving out of a directory

```
cd ..
```

  - Going back and forth in the directory tree

```
cd ../../folder1/subfolder1
```

  - Going back to the root directory

```
cd ~
```

- "**Tab**" to use the auto-completion
- **Ctrl + C** to stop a program execution
- Many more bash commands to use...

Previously on Intro to Programming (Python)

- Data types:
  - integer
  - float
  - string
  - boolean
- **If**, **For** and **While** loops:
  - syntax
  - indentation
- Data collections:
  - list
  - tuple
  - set
  - dictionary
- Python Standard library
  - Python modules
  - Python built-in functions
- Functions:
  - Parameters and arguments
  - Return values
  - Scope of variable
- Building abstraction with :
  - Recursive functions
  - High order functions

Today

Today

- Read and write files

Manipulate files : Open and read a file 1/3

- To do so you have one function available in the built-in functions of python
  - https://docs.python.org/3/library/functions.html
- The function **open()**
  - that works like that open(file, mode=‘r’, buffering=- 1, encoding=None, errors=None, newline=None, closefd=True, opener=None)
- see https://docs.python.org/3/library/functions.html#open
- Mode can be:
  - “r” - Read - Default value. Opens a file for reading, error if the file does not exist
  - “a” - Append - Opens a file for appending, creates the file if it does not exist
  - “w” - Write - Opens a file for writing, creates the file if it does not exist
  - “x” - Create - Creates the specified file, returns an error if the file exist

Manipulate files : Open and read a file 2/3

- to manipulate and for example print the text you need to read it using **read()**
- All available function are specified here:
  - https://docs.python.org/3/library/io.html

```
Myfile = open('Survival rules for programming.txt', 'r')
print(Myfile.read())
```

```
## Try by yourself before looking for solutions
##
## Internet is your best friend
##
## Read the manual
##
## There is always a manual
##
## Have you read the fucking manual?
##
## Not yet ? Then read it
##
## Always read the error message
```

```
print(Myfile.read(5))
```

```
Myfile.close()
```

Manipulate files : Open and read a file 3/3

- You can also use:
- readline() can be used to return one line
- readlines() can be used to return a list of lines
- NB 1: if the file is not close the next call of readline() or readlines() will take the subsequent lines of the file even though you specified the first index
- NB 2: As readlines() return a list you can use all the functions in the built in module **string** such as len(), joins(), split()...

```
Myfile = open('Survival rules for programming.txt', 'r')
print(Myfile.readline())
```

```
## Try by yourself before looking for solutions
```

```
print(Myfile.readlines(1))
```

```
## ['\n', 'Internet is your best friend\n']
```

```
print(Myfile.readlines()[1])
```

```
## Read the manual
```

Manipulate files : create a file

- Note that if you don't specify any path, it will be created in the current directory (ie, same directory as your script). It's called creating a file using a **relative path**

```
import os
path = os.getcwd()

print(os.listdir(path))
```

```
## ['6th class.Rmd', '6th-class.Rmd', '6th-class.pdf', 'Survival rules for programmi
```

```
MyTestFile = open('test.txt', 'x')

print(os.listdir(path))
```

```
## ['test.txt', '6th class.Rmd', '6th-class.Rmd', '6th-class.pdf', 'Survival rules f
```

- If you want to create in a precise directory you can specify it using an **absolute path**

```
MyTestFile = open('/home/henri/Desktop/test.txt', 'x')
```

Manipulate files : write a file

- We need an access mode 'w' if we want to create and write anything into a file
- Note that to be able to read a just created file you need to close it and open it again in read mode

```
MyTestFile = open('test2.txt', 'w')

MyTestFile.write("Once upon a time in a Cognitive Master")
```

```
## 38
```

```
MyTestFile.close()

MyTestFile = open('test2.txt', 'r')

print(MyTestFile.read())
```

```
## Once upon a time in a Cognitive Master
```

Manipulate files : append text to a file

- We need an access mode 'a'

```
MyTestFile = open('test2.txt', 'a')

MyTestFile.write("There was a module names Intro to programming")
```

```
## 45
```

```
MyTestFile.close()

MyTestFile = open('test2.txt', 'r')

print(MyTestFile.read())
```

```
## Once upon a time in a Cognitive MasterThere was a module names Intro to programmi
```

Manipulate files : specific character and new lines

- Inside a string you can use the anti slash to insert special codes:
  - \n return to line
  - **** add a tab
  - **** return to line (same as \n in python)
  - " add a quotation mark inside a string delimited itself by “

```
MyTestFile = open('test2.txt', 'a')

MyTestFile.write("\nWith youngs and bright \tstudents \r!!!!!")
```

```
## 40
```

```
MyTestFile.close()

MyTestFile = open('test2.txt', 'r')

print(MyTestFile.read())
```

```
## Once upon a time in a Cognitive MasterThere was a module names Intro to programmi
## With youngs and bright   students
## !!!!!
```

```
MyTestFile.close()
```

Manipulate files : Automatic close of the file

- the **with open()** statement automatically close the file

```
lines = ['and one last line', '\n... and one last...']

with open("test2.txt", "a") as MyTestFile:
    for line in lines:
      MyTestFile.write(line)
```

```
## 17
## 20
```

```
MyTestFile = open("test2.txt", "r")
print(MyTestFile.read())
```

```
## Once upon a time in a Cognitive MasterThere was a module names Intro to programmi
## With youngs and bright   students
## !!!!!and one last line
## ... and one last...
```

Exercices

- 1 Write a script that prints the first 10 lines of a file
- 2 Write a script that prints the last 10 lines of a file (or the whole file is it is less than 10 lines long)
- 3 Write a script that opens and read a text file, and print all the lines that contain a given target word
- 4 compute the number of words (removing punctuation) in a text file (Hint: use split() and strip() functions)
- 5 compute the number of occurrences of each word in a text file
- 6 print a bar plot of the occurrences found in the previous exercices (using matplotlib)

Exercises 1

- 1 Write a script that prints the first 10 lines of a file

```
MyTestFile = open('Survival rules for programming.txt', 'r')
lines = MyTestFile.readlines()

for i in range(10):
  print(lines[i])
```

```
## Try by yourself before looking for solutions
##
##
##
## Internet is your best friend
##
##
##
## Read the manual
##
##
##
## There is always a manual
##
##
##
```

Exercises 1

- 1 Write a script that prints the first 10 lines of a file

```
MyTestFile = open('Survival rules for programming.txt', 'r')
lines = MyTestFile.readlines()
print(lines[0:9])
```

```
## ['Try by yourself before looking for solutions\n', '\n', 'Internet is your best f
```

Exercises 2

- 2 Write a script that prints the last 10 lines of a file (or the whole file is it is less than 10 lines long)

```
MyTestFile = open('Survival rules for programming.txt', 'r')
lines = MyTestFile.readlines()

for i in range(1,10):
  print(lines[-i])
```

```
## Always read the error message
##
##
##
## Not yet ? Then read it
##
##
##
## Have you read the fucking manual?
##
##
##
## There is always a manual
##
##
```

Exercises 3

- 3 Write a script that opens and read a text file, and print all the lines that contain a given target word

```
def print_line_with_specific_word(text,word):
  for l in range(1,10):
    if word in text[l]:
      print(text[l])
    else :
      print("FALSE")


MyTestFile = open('Survival rules for programming.txt', 'r')
text = MyTestFile.readlines()

print_line_with_specific_word(text,"manual")
```

```
## FALSE
## FALSE
## FALSE
## Read the manual
##
## FALSE
## There is always a manual
##
```

Exercises 4

- 4 compute the number of words (removing punctuation) in a text file (Hint: use split() and strip() functions)

```
import string

def count_words(text):
  c = 0
  for l in text: #
    #print(l.split())
    for w in l.strip().split():
      #print(list(w))
      for char in list(w):
        #print(c)
          if char in string.punctuation:
            w =w.replace(char,'') # You replace a punctuation by empty space
      if w != '': # check that you don't have an empty word
        c +=1
      print(w)

  return c

MyTestFile = open('Survival rules for programming.txt', 'r')
text = MyTestFile.readlines()

print(count_words(text))
```

Exercises 5

- 5 compute the number of occurrences of each word in a text file

```
def count_words(text):
  dict_words = {}
  for l in text:
    for w in l.strip().split():
      if w in dict_words:
        dict_words[w] +=1
      else:
        dict_words[w] = 1
  return dict_words


MyTestFile = open('Survival rules for programming.txt', 'r')
text = MyTestFile.readlines()

print(count_words(text))
```

```
## {'Try': 1, 'by': 1, 'yourself': 1, 'before': 1, 'looking': 1, 'for': 1, 'solution
```

Exercises 6

- 6 print a bar plot of the occurrences found in the previous exercises (using matplotlib)

```
import matplotlib.pyplot as plt

def count_words(text):
  dict_words = {}
  for l in text:
    for w in l.strip().split():
      if w in dict_words:
        dict_words[w] +=1
      else:
        dict_words[w] = 1
  return dict_words

def plot_frequency(dictionary):
  plt.bar(list(dictionary.keys()), dictionary.values(), color='g')
  plt.show()

MyTestFile = open('Survival rules for programming.txt', 'r')
text = MyTestFile.readlines()

plot_frequency(count_words(text))
```