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

- Read and write files

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **open()**.

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **open()**.
    - that works like that open(file, mode=‘r’, buffering=- 1, encoding=None, errors=None, newline=None, closefd=True, opener=None)

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **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

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **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:

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **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

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **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

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **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

Manipulate files : Open and read a file 1/3

- To interact with files in Python, you have a built-in function available: **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 2/4

- To manipulate and, for example, print the text from a file, you can use **read()**.

Manipulate files : Open and read a file 2/4

- To manipulate and, for example, print the text from a file, you can use **read()**.
- All available file-related functions are specified in the IO module: https://docs.python.org/3/library/io.html

Manipulate files : Open and read a file 2/4

- To manipulate and, for example, print the text from a file, you can use **read()**.
- All available file-related functions are specified in the IO module: https://docs.python.org/3/library/io.html
- Example 1:

```
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
```

```
Myfile.close()
```

Manipulate files : Open and read a file 2/4

- Example 2:

```
Myfile = open('Survival rules for programming.txt', 'r')
print(Myfile.read(5))
```

```
## Try b
```

```
Myfile.close()
```

Manipulate files : Open and read a file 3/4

- You can also use:

```
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 : Open and read a file 3/4

- You can also use:
  - readline() can be used to return one line

```
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 : Open and read a file 3/4

- You can also use:
  - readline() can be used to return one line
  - readlines() can be used to return a list of lines

```
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 : Open and read a file 3/4

- You can also use:
  - readline() can be used to return one line
  - readlines() can be used to return a list of lines

- Note 1: If a file is not closed, the next call of **readline()** or **readlines()** will continue from where it left off, even if you specify a line index.

```
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 : Open and read a file 3/4

- You can also use:
  - readline() can be used to return one line
  - readlines() can be used to return a list of lines

- Note 1: If a file is not closed, the next call of **readline()** or **readlines()** will continue from where it left off, even if you specify a line index.

- Note 2: Since **readlines()** returns a list, you can use all the functions from the built-in module string, such as **len()**, **join()**, and **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

- If you don't specify a path, the file will be created in the current directory (i.e., the same directory as your script). This is called creating a file using a **relative path**.

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

print(os.listdir(path))

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

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

Manipulate files : create a file

- If you don't specify a path, the file will be created in the current directory (i.e., the same directory as your script). This is called creating a file using a **relative path**.

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

print(os.listdir(path))

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

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

- 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

- To create and write to a file, you need to use the access mode 'w'.
- Note that to read a file just created, you need to close it and open it again in read mode. >
  > python >   MyTestFile = open('test2.txt', 'w') >   MyTestFile.write("Once upon a time in a Cognitive Master") > >
  > >   ## 38 > >
  > python >   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

- To append text to a file, use 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

- Within a string, you can use the escape character (anti-slash) to insert special codes:

```
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 progr
## With youngs and bright      students
## !!!!!
```

```
MyTestFile.close()
```

Manipulate files : specific character and new lines

- Within a string, you can use the escape character (anti-slash) to insert special codes:
    - **n** newline

```
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 progr
## With youngs and bright      students
## !!!!!
```

```
MyTestFile.close()
```

Manipulate files : specific character and new lines

- Within a string, you can use the escape character (anti-slash) to insert special codes:
  - **n** newline
  - **t** tab

```
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 progr
## With youngs and bright     students
## !!!!!
```

```
MyTestFile.close()
```

Manipulate files : specific character and new lines

- Within a string, you can use the escape character (anti-slash) to insert special codes:
  - **n** newline
  - **t** tab
  - **r** carriage return (same as \n in python)

```
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 progr
## With youngs and bright 	students
## !!!!!
```

```
MyTestFile.close()
```

Manipulate files : specific character and new lines

- Within a string, you can use the escape character (anti-slash) to insert special codes:
  - **n** newline
  - **t** tab
  - **r** carriage return (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 progr
## With youngs and bright     students
## !!!!!
```

```
MyTestFile.close()
```

Manipulate files : Automatic close of the file

- The with **open()** statement automatically closes 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...
```

Exercises

- 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 if it is less than 10 lines long).
- 3 Write a script that opens and reads a text file, printing all lines containing a given target word.
- 4 Calculate the number of words (removing punctuation) in a text file. Hint: use the **split()** and **strip()** functions.
- 5 Calculate the number of occurrences of each word in a text file.
- 6 Print a bar plot of the word occurrences found in the previous exercises using matplotlib.