

The background is a complex network of nodes and connections. Nodes are represented by circles of various sizes and colors, including blue, green, yellow, red, and purple. Some nodes have internal patterns like stripes or concentric circles. They are interconnected by a web of thin, light gray lines. The overall color palette is muted, with a light gray background and a subtle grid of small dots.

PYTHON 101

Plan

1. Datatypes (numerics, string, list, dict)
2. Loops , Conditions, Functions
3. Import/export data from files



Programming for Everybody (Getting Started with Python)

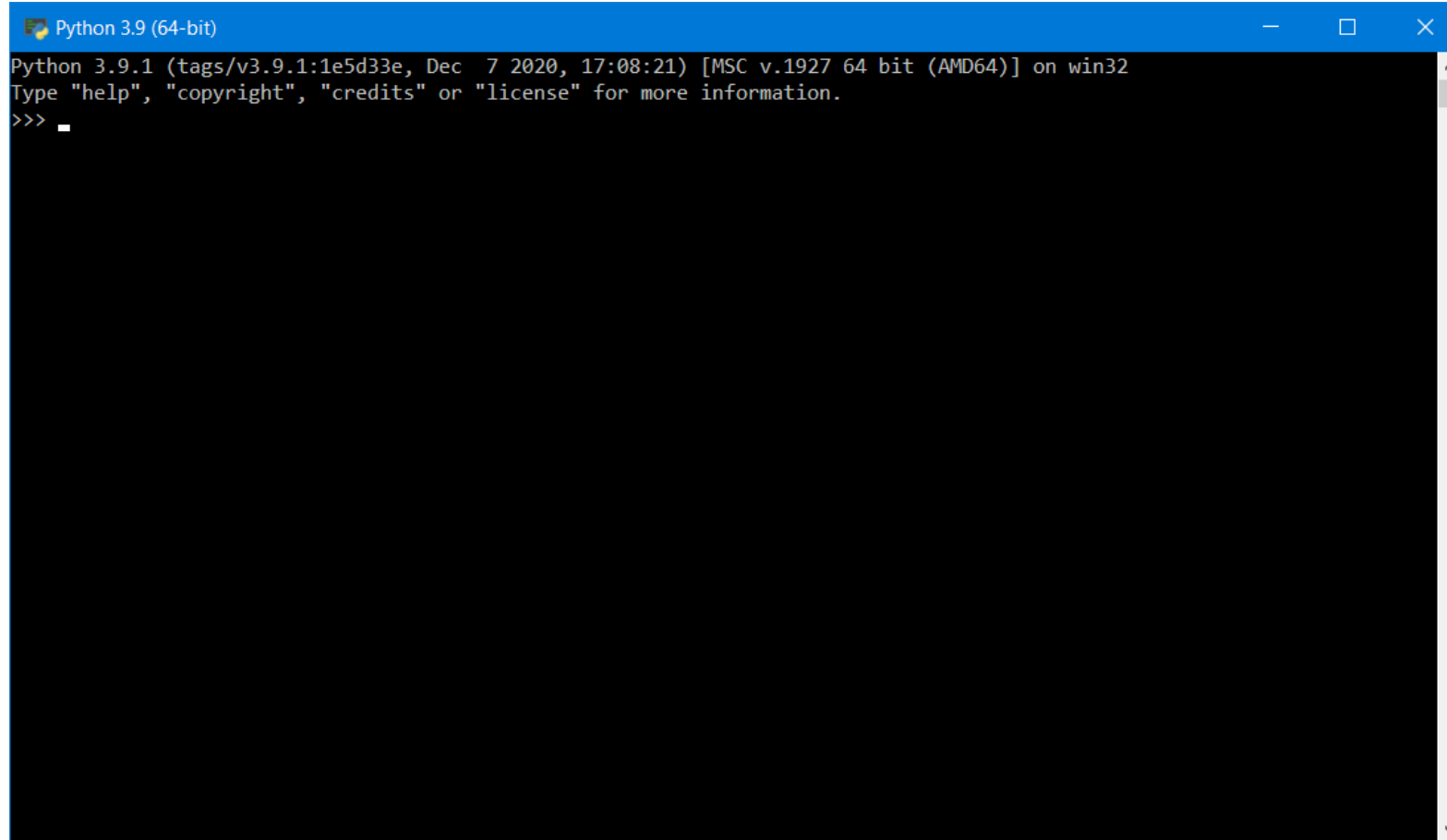
University of Michigan

Cours

★★★★☆ 4.8 (178 369) | 2,1 M étudiants

■ Mixed

First steps with the prompt



```
Python 3.9 (64-bit)  
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> _
```

1. Datatypes

- Integer
- Float
- String
- List
- Dict
- More advanced datatypes:
 - Array
 - Dataframe
 - ...

Python 3.9 (64-bit)

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

```
>>> 1+1
```

```
2
```

```
>>> a=1
```

```
>>> a+a
```

```
2
```

```
>>> a
```

```
1
```

```
>>> print(hello world)
```

```
File "<stdin>", line 1
```

```
    print(hello world)
```

```
        ^
```

```
SyntaxError: invalid syntax
```

```
>>> print("hello world")
```

```
hello world
```

```
>>> hello="world"
```

```
>>> hello+a
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: can only concatenate str (not "int") to str
```

```
>>> str(a)
```

```
'1'
```

```
>>> hello+str(a)
```

```
'world1'
```

```
>>> █
```

More on Strings

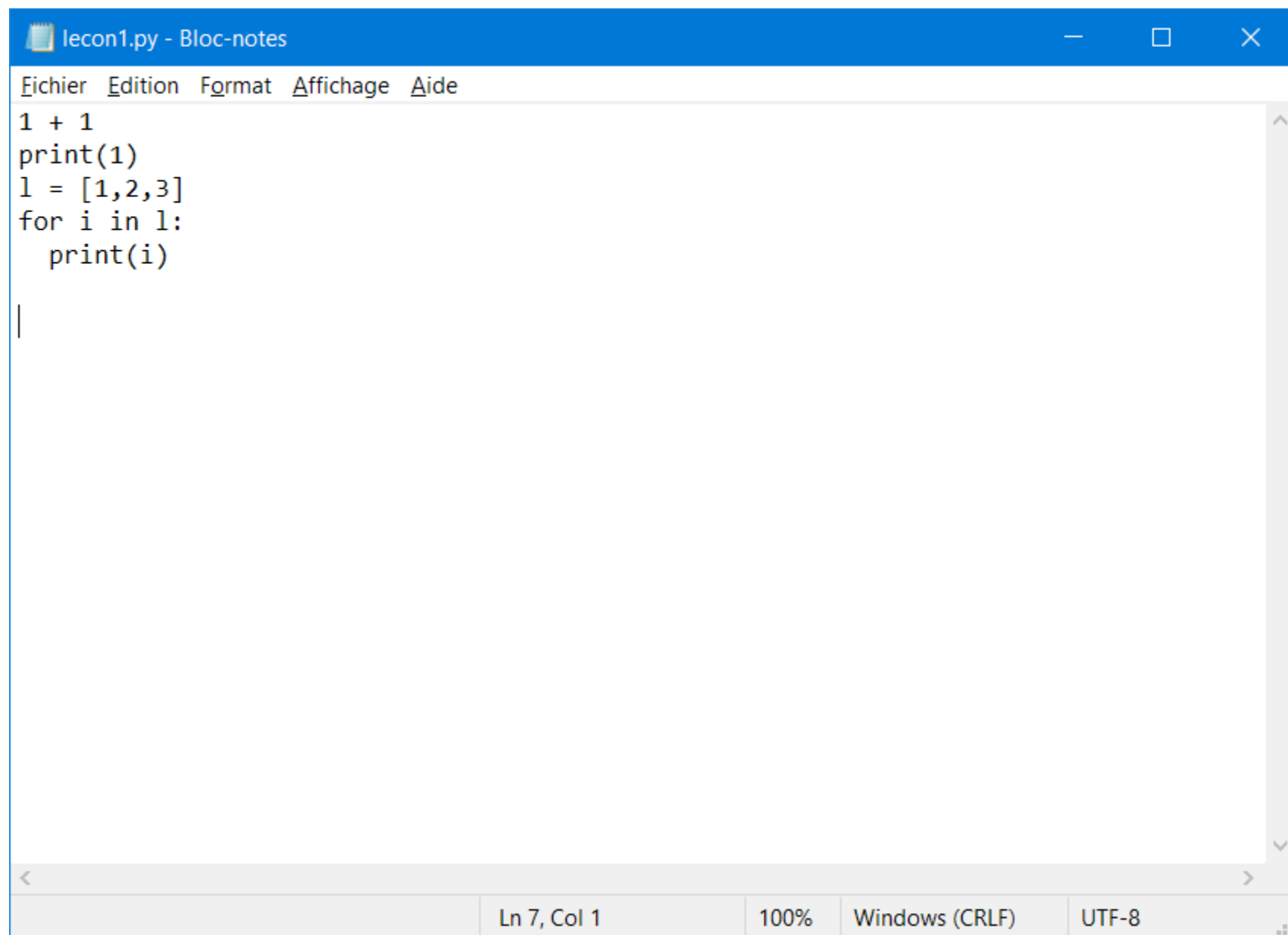
- [6.1: A string is a sequence](#)
- [6.2: Getting the length of a string using len](#)
- [6.3: Traversal through a string with a loop](#)
- [6.4: String Slices](#)
- [6.5: Strings are immutable](#)
- [6.6: Looping and Counting](#)
- [6.7: The in operator](#)
- [6.8: String Comparison](#)
- [6.9: String Methods](#)
- [6.E: Strings \(Exercises\)](#)
- [6.G: Strings \(Glossary\)](#)
- [6.10: Parsing strings](#)
- [6.11: Format operator](#)
- [6.12: Debugging](#)

2. Loops , Conditions, Functions

- Loops
 - Conditions
 - Functions
-
- Write your first program (in Notepad)

Python 3.9 (64-bit)

```
>>> l=[1,2,3]
>>> for i in l:
...     print(i)
...
1
2
3
>>> _
```

The image shows a screenshot of a Python IDE window titled "lecon1.py - Bloc-notes". The window has a blue title bar with standard minimize, maximize, and close buttons. Below the title bar is a menu bar with the following options: Fichier, Edition, Format, Affichage, and Aide. The main text area contains the following Python code:

```
1 + 1
print(1)
l = [1,2,3]
for i in l:
    print(i)
```

The cursor is positioned at the start of a new line (line 7) below the code. The bottom status bar displays the following information: "Ln 7, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

Python 3.9 (64-bit)

```
>>> if a > 1:  
...     print('hello')  
...  
>>>
```

More on Lists

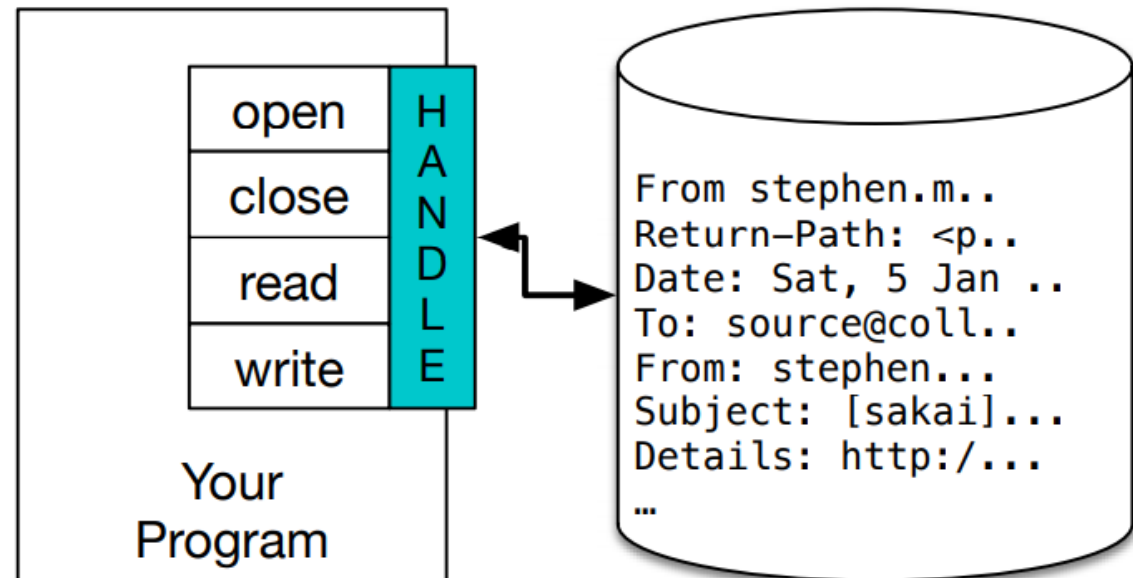
- [8.1: A list is a sequence](#)
- [8.2: Lists are mutable](#)
- [8.3: Traversing a List](#)
- [8.4: List operations](#)
- [8.5: List Slices](#)
- [8.6: List Methods](#)
- [8.7: Deleting Elements](#)
- [8.8: Lists and Functions](#)
- [8.9: Lists and Strings](#)
- [8.E: Lists \(Exercises\)](#)
- [8.G: Lists \(Glossary\)](#)
- [8.10: Parsing lines](#)
- [8.11: Objects and Values](#)
- [8.12: Aliasing](#)
- [8.13: List arguments](#)
- [8.14: Debugging](#)

3. Import/export data from files

```
>>> fhand = open('mbox.txt')
```

```
>>> print(fhand)
```

```
<_io.TextIOWrapper name='mbox.txt' mode='r'  
encoding='cp1252'>
```



Reading files

```
fhand = open('mbox-short.txt')  
count = 0  
for line in fhand:  
    count = count + 1  
print('Line Count:', count)
```

Writing files

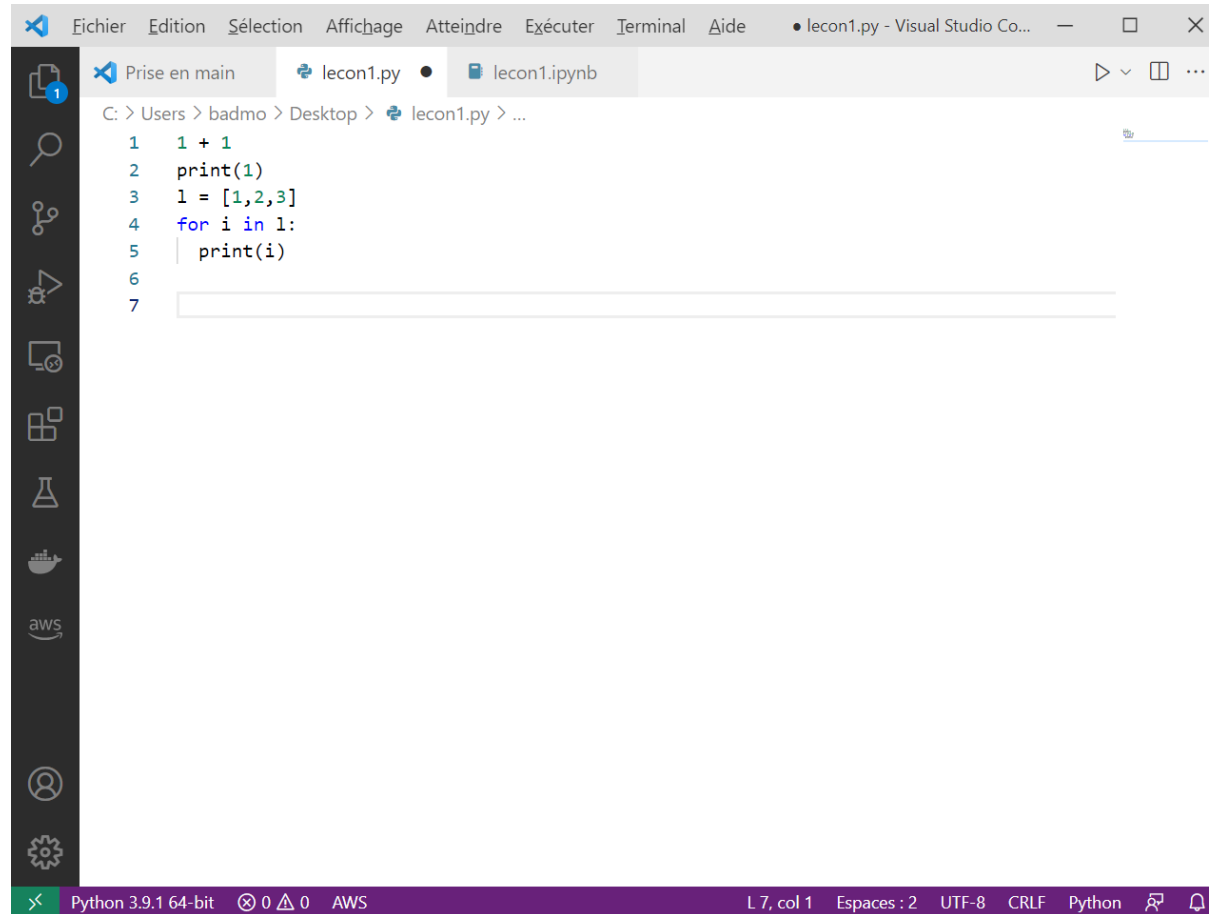
```
>>> fout = open('output.txt', 'w')
>>> line1 = "This here's the wattle,\n"
>>> fout.write(line1)
>>> line2 = 'the emblem of our land.\n'
>>> fout.write(line2)
>>> fout.close()
```

More on Files

- [7.1: Persistence](#)
- [7.2: Opening Files](#)
- [7.3: Text files and Lines](#)
- [7.4: Reading Files](#)
- [7.5: Searching through a File](#)
- [7.6: Letting the user choose the file name](#)
- [7.7: Using try, except, and open](#)
- [7.8: Writing Files](#)
- [7.9: Debugging](#)
- [7.E: Files \(Exercises\)](#)
- [7.G: Files \(Glossary\)](#)

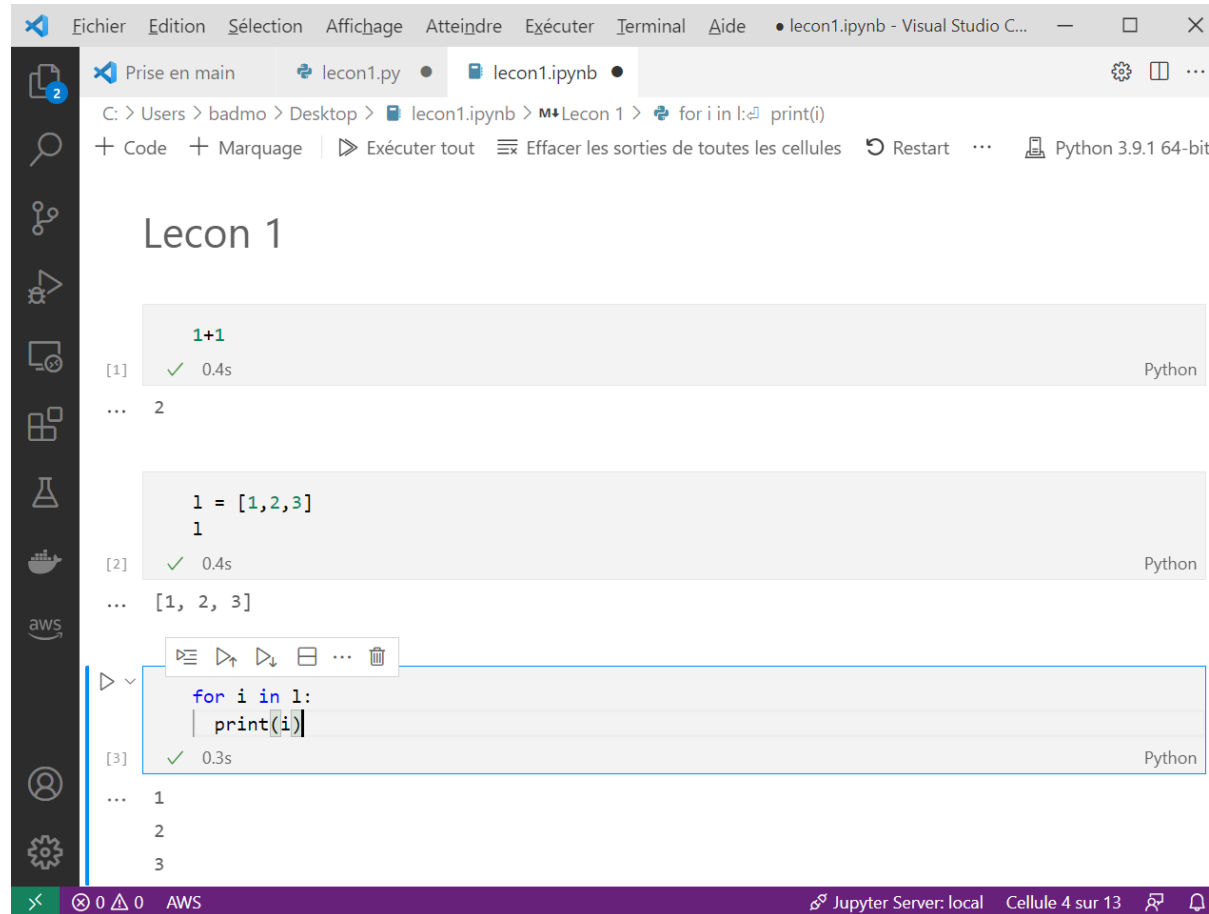
Visual Studio Code

Recommended editor for Python



<https://code.visualstudio.com/>

Notebook format for interactivity



Next...

- Plotting
- Manage packages, Notebooks & Environments



EXERCISES

Immo.py

- Call a function `price`, that will return the price of real estate in Paris as a dictionary
- Get a list of location (lat,lon) and price
- Select only the estate under 500k€
- Write a file containing this data
- (Bonus: plot the data on a map)

Immo.py

```
def dvf(code_commune):  
    # demande de valeur financiere  
    # dvf(75114)  
    import urllib.request  
    import json  
  
    try:  
        url = "https://26yrburnr0.execute-api.eu-west-3.amazonaws.com/dev/dvf?code_commune=".  
        response = urllib.request.urlopen(url)  
        html = response.read()  
        json_data = json.loads(html)  
  
    except urllib.error.URLError:  
        # if the API doesnt work, read the file  
        print('Problem with the API')  
  
    return json_data
```

Python

<https://gist.github.com/slevin48/05c0d4f348f0f10870a0fa721cfcb1b1>

Immo.py

```
d=dvf(75114)
```

Python

```
d.keys()
```

Python

```
v=d['valeur_fonciere']  
v
```

Python

```
{'9969': 95000.0,  
 '9970': 545000.0,  
 '9971': 220000.0,  
 '9972': 132500.0,
```

Immo.py

```
for i in v:
    # print(v[i])
    if v[i]>500000:
        print("indice : " + i + " - valeur : " + str(v[i]))
```

Python

indice : 9970 - valeur : 545000.0

indice : 9973 - valeur : 590000.0

```
f = open("mes_prix.txt", "w")
for i in v:
    # print(v[i])
    if v[i]>500000:
        f.write("indice : " + i + " - valeur : " + str(v[i]))
        f.write("\n")
f.close()
```

Python

Music.py

- Parse Streaming History
 - Create a dictionary
 - id
 - artist name
 - track name
- Music Taste Analysis
 - Get music features
 - Plot features
- Get recommendation



Music for loop

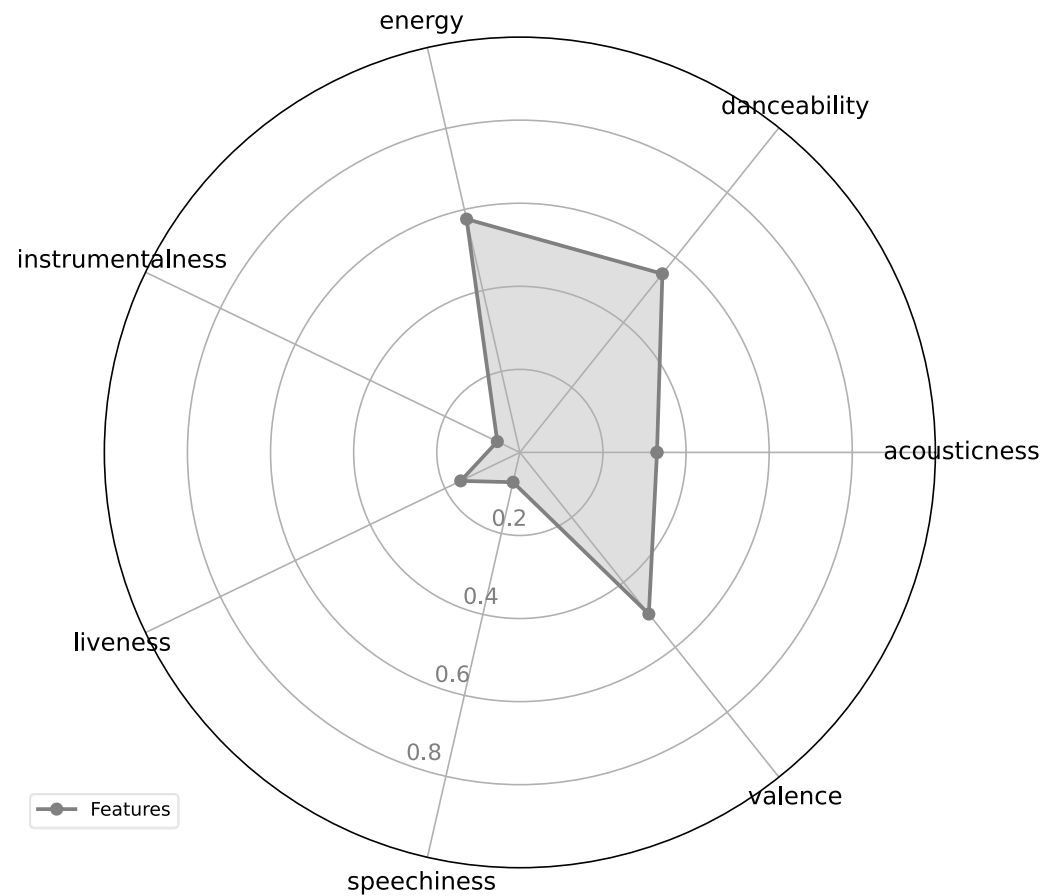
```
with open("saved_tracks_20210306.json", "r") as f:  
    results = json.load(f)
```

```
tracks = []  
for idx, item in enumerate(results['items']):  
    track = item['track']  
    tracks.append([idx, track['artists'][0]['name'],  
                  track['name']])
```

Music Dict

```
trackDict = {"id":[], "artist":[], "name":[]}  
for idx, item in enumerate(results['items']):  
    track = item['track']  
    trackDict["id"].append(idx)  
    trackDict["artist"].append(track['artists'][0]['  
name'])  
    trackDict["name"].append(track['name'])
```

Music taste analysis



Music taste analysis

```
import spotifyAPI
from secret import clientId, clientSecret
token = spotifyAPI.get_token(clientId, clientSecret)
lucy_id = spotifyAPI.get_track_id2('Lucy in the Sky'
, token, artist = 'The Beatles')

url = "https://open.spotify.com/track/"+lucy_id
import webbrowser
webbrowser.open(url)
```

```
import pandas as pd
```

```
lucy_features = spotifyAPI.get_features(lucy_id, token)
```

```
df = pd.DataFrame(lucy_features, index=[0])
```

```
df_features = df.loc[:, ['acousticness', 'danceability',  
                          'energy', 'instrumentalness', 'liveness', 'speechiness',  
                          'valence']]
```

```
spotifyAPI.feature_plot(df_features)
```

Music recommendation

```
json_response = spotifyAPI.get_track_reco(lucy_id,to
ken)
uris =[]
for i in json_response['tracks']:
    uris.append(i)
    print(f"\n{i['name']}\n by {i['artists']
[0]['name']}")
```

Sources

 <p>PYTHON DATA STRUCTURES</p>  <p>Python Data Structures University of Michigan</p> <p><u>Data Structures</u></p> <p>1 COURS</p>	 <p>PYTHON WEB DATA</p>  <p>Using Python to Access Web Data University of Michigan</p> <p><u>Using Python to Access Web Data</u></p> <p>1 COURS</p>	 <p>PYTHON DATABASES</p>  <p>Using Databases with Python University of Michigan</p> <p><u>Using Databases with Python</u></p> <p>1 COURS</p>	 <p>PYTHON CAPSTONE</p>  <p>Capstone: Retrieving, Processing, and Visualizing Data with Python University of Michigan</p> <p><u>Capstone: Retrieving, Processing, and Visualizing Data with Python</u></p> <p>1 COURS</p>
---	--	--	---

[PY4E - Python for Everybody](#)

[Python for Everybody \(dr-chuck.com\)](#)

Go further

[Crash Course on Python | Google](#)



Using Python to Interact with the Operating System

Google

[Using Python to Interact with the Operating System](#)
1 COURSE



Google IT Automation with Python

Google

[Google IT Automation with Python](#)
6 COURSE



Configuration Management and the Cloud

Google

[Configuration Management and the Cloud](#)
1 COURSE



Troubleshooting and Debugging Techniques

Google

[Troubleshooting and Debugging Techniques](#)
1 COURSE

[Introduction to Git and GitHub | Google](#)

[Automating Real-World Tasks with Python | Google](#)

Data Science



Python for Data Science and AI

IBM

Cours

★★★★☆ 4.6 (19 833) | 260 000 étudiants


Beginner



Data Analysis with Python

IBM

1 COURS



Data Visualization with Python

IBM

1 COURS



Machine Learning with Python

IBM

1 COURS



Statistics for Data Science with Python

IBM

1 COURS

Uninstall Python

