28/02/2022

Python pour l'ingénieur

Compléments

Module de formation

© Copyright 2022 Cyril Keime & Arnaud Bêche

FICHIERS CSV

© Copyright 2022 Cyril Keime & Arnaud Bêche

Généralités

- Un fichier CSV est très pratique et supporte la gestion de configuration avec git.

- Il mérite de ce fait un peu d'attention ! Utiliser un fichier CSV requiert de réfléchir aux étapes suivantes :

LIRE	read
NETTOYER	clean
NORMALISER	normalize
FILTRER	filter
TRANSFORMER	transform
ENREGISTRER	write

format texte ouvert représentant des données tabulaires sous forme de valeurs séparées par un délimiteur (point-virgule par exemple)

© Copyright 2022 Cyril Keime & Arnaud Bêche

Liste de tuples

- La chaîne de traitement de l'information peut être implémentée avec une suite de fonctions élémentaires :

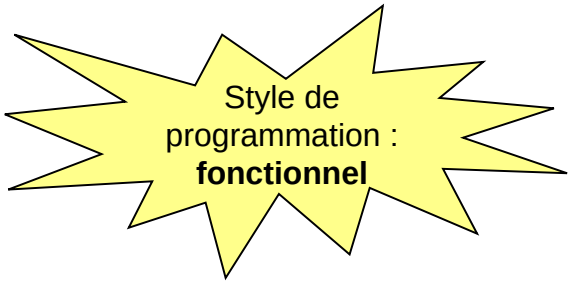
  ENTREE liste de tuples (+ paramètres)

  SORTIE liste de tuples

- Autrement dit, une ligne du fichier CSV devient un tuple, et l'ensemble du fichier une liste de tuples.

- Intérêt du tuple : immutable. Le seul moyen de le modifier est d'en créer un nouveau.

- Chaque fonction élémentaire crée et renvoie une nouvelle liste de tuples.



1) Lire la documentation des fonctions suivantes :

```
map()
filter()
```

2) Ne pas utiliser d'état global (stateless)

© Copyright 2022 Cyril Keime & Arnaud Bêche

Package csv

- Lire un fichier CSV requiert l'utilisation du package csv.
  - Note : privilégier le package csv plutôt que de réinventer une fonction de lecture…

```
def read_data(path):
    """Return a list of tuples (CSV rows)
       (without header row)
    """
    data = []
    with open(path, newline='') as f:
        myreader = csv.reader(f, delimiter=';')
        next(myreader)
        for row in myreader:
            data.append(tuple(row))
    return data
```

- En style fonctionnel, créer des fonctions lambda pour accéder aux colonnes.
  - Note : ne pas utiliser les indices de colonnes dans les fonctions élémentaires ! (En tout cas, limiter au strict nécessaire, si possible dans une seule fonction.)

```
MODE       = lambda x:x[0]
EMETTEUR   = lambda x:x[1]
PLATEFORME = lambda x:x[2]
ID_UNIQUE  = lambda x:x[3]
```

EXEMPLE

```
liste_plateforme = [PLATEFORME(t) for t in data]
```


© Copyright 2022 Cyril Keime & Arnaud Bêche

FICHIERS XML

© Copyright 2022 Cyril Keime & Arnaud Bêche

Généralités

- Package utilisé : lxml
- «The lxml XML toolkit is a Pythonic binding for the C libraries libxml2 and libxslt.»
- «lxml is the most feature-rich and easy-to-use library for processing XML and HTML in the Python language.»
- Licences :
  - lxml : BSD
  - libxml2 : MIT
  - libxslt : MIT

https://lxml.de/

```
sudo pip install lxml
```


© Copyright 2022 Cyril Keime & Arnaud Bêche

Ecrire un fichier XML (1/2)

1

```
# Importer builder et etree
import lxml.builder
import lxml.etree
```

2

```
# Créer un ElementMaker
E = lxml.builder.ElementMaker()
```

3

```
# Créer un élément terminal.
my_elem = E.my_elem(
    'Ceci est un exemple.',
    answer='42',
    unit='no unit'
)
```

```
    # Créer un élément en spécifiant un élément de niveau inférieur.
    root_elem = E.root_elem(
        my_elem,
        attr=''
    )
```

```
# Ajouter un élément.
my_elem_bis = E.my_elem_bis('')
root_elem.append(my_elem_bis)
```

```
    # Modifier le contenu.
    my_elem_bis.text = 'Ceci est un autre exemple.'
```


© Copyright 2022 Cyril Keime & Arnaud Bêche

Ecrire un fichier XML (2/2)

4

```
def write_xml(xml):
    """Crée un fichier XML."""
    txt = lxml.etree.tostring(xml, encoding='unicode', pretty_print=True)
    txt = '<?xml version="1.0" encoding="UTF-8" ?>\n' + txt
    with open('fichier_xml_creer.xml', 'w', encoding='utf-8') as f:
        f.write(txt)
```


© Copyright 2022 Cyril Keime & Arnaud Bêche

Lire un fichier XML

1

```
# Importer etree
import lxml.etree
```

2

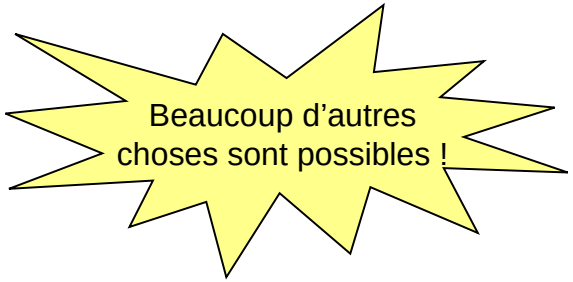
```
# Lecture d'un fichier XML.
xml = lxml.etree.parse('fichier_xml_creer.xml')
```

3

```
# Trouver un élément.
print(xml.find('my_elem_bis').text)

# Lister les sous-éléments.
# (et afficher le tag et les attributs)
for elem in xml.getroot():
    print(f'{elem.tag} --> {elem.attrib}')

# Parcourir tous les éléments.
# (et afficher le chemin)
for elem in xml.iter():
    print(f'{elem.tag} --> {xml.getpath(elem)}')
```



© Copyright 2022 Cyril Keime & Arnaud Bêche

Pour aller plus loin

- The lxml.etree Tutorial
  https://lxml.de/tutorial.html

- Introduction to the Python lxml Library
  https://stackabuse.com/introduction-to-the-python-lxml-library/

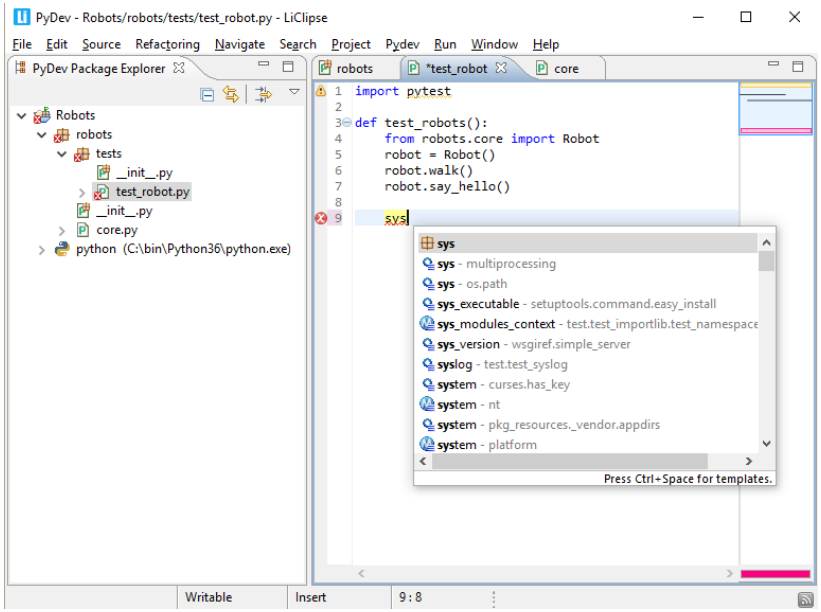
© Copyright 2022 Cyril Keime & Arnaud Bêche

DEBUG

© Copyright 2022 Cyril Keime & Arnaud Bêche

PyDev (1/2)

- Exemple : PyDev
- PyDev is a Python IDE for Eclipse.

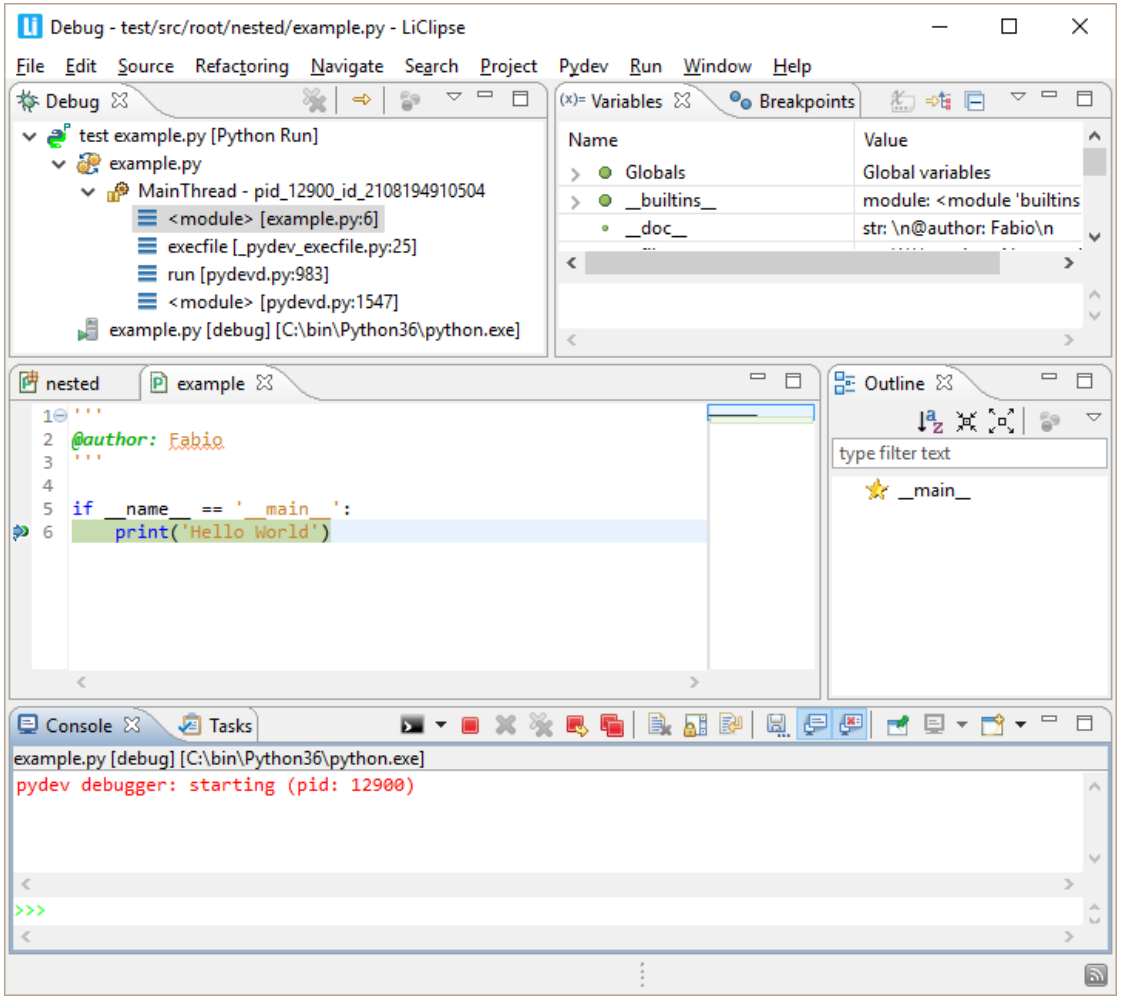


https://www.pydev.org/

© Copyright 2022 Cyril Keime & Arnaud Bêche

PyDev (2/2)

Now, let's go on to starting a debug session. First, you'll need to add a breakpoint in the print('Hello World') line. To do so, go to that line, use Ctrl+F10 and select 'Add breakpoint' (or double-click the left ruler), then use the F11 to re-run the last launch. Doing so, will trigger you to go to the debug perspective.

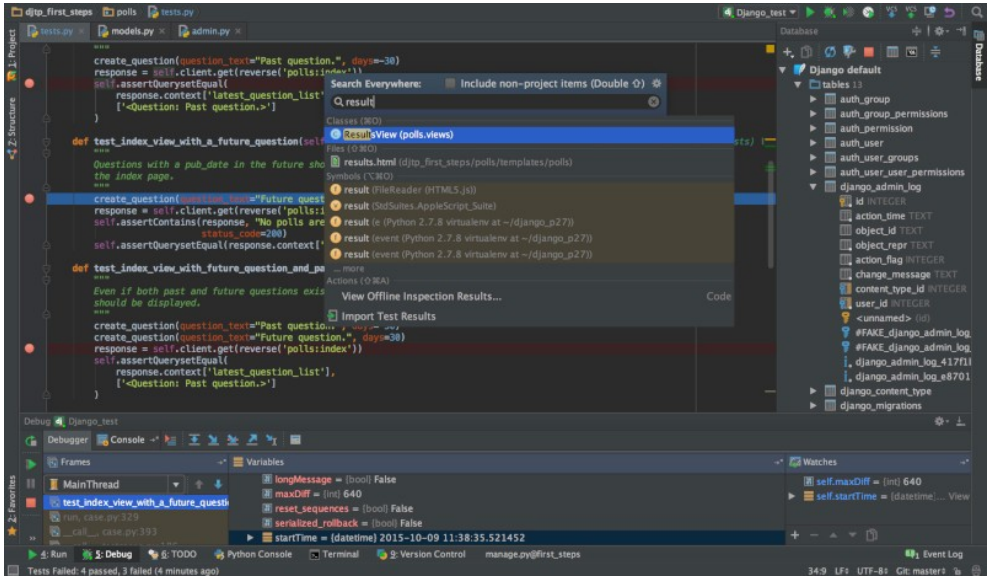


https://www.pydev.org/manual_101_run.html

© Copyright 2022 Cyril Keime & Arnaud Bêche

PyCharm (1/2)

- Exemple : PyCharm
- «The Python IDE for Professional Developers»
- PyCharm Community Edition



https://www.jetbrains.com/pycharm/

© Copyright 2022 Cyril Keime & Arnaud Bêche

PyCharm (2/2)

https://www.jetbrains.com/help/pycharm/part-1-debugging-python-code.html

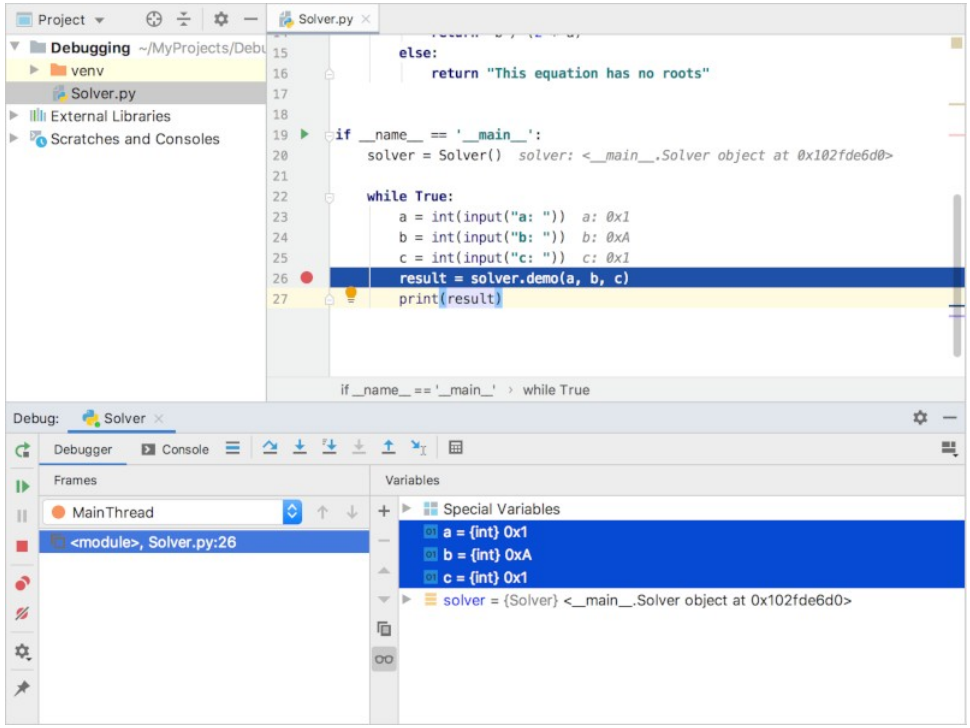
This brief tutorial is over - congrats! Let's repeat what you've learnt from it:

You've refreshed your knowledge of the breakpoints and learnt how to place them.

You've learnt how to begin the debugger session, and how to show the Python prompt in the debugger console.

You've refreshed your knowledge about the inline debugging.

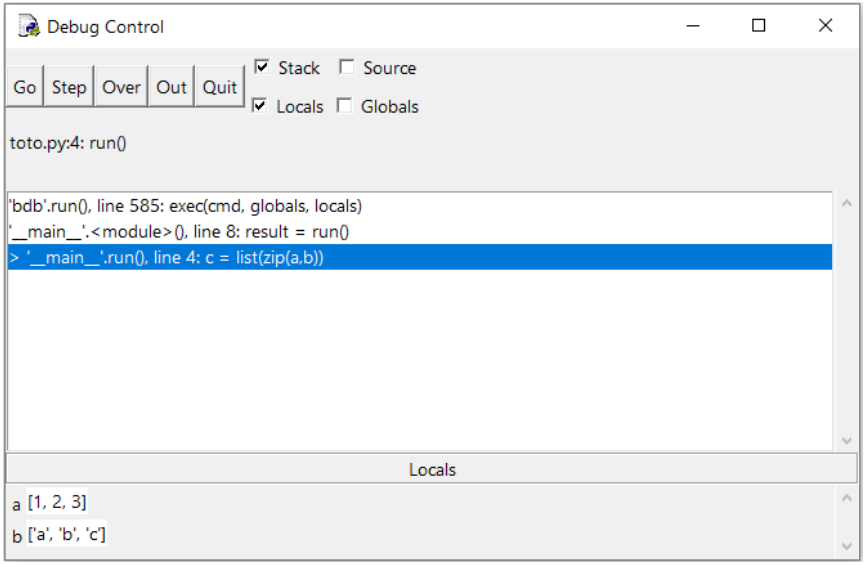
You've tried hands on stepping, watches and evaluating expressions.



© Copyright 2022 Cyril Keime & Arnaud Bêche

IDLE

- Dans le code source, clic droit pour activer ou désactiver un point d'arrêt.
- Dans la fenêtre principale (Shell), activer le debugger : Debug / Debugger
- Dans la fenêtre du code source, lancer le programme (F5)



Go : to make the program run at normal speed until a breakpoint is encountered.

Step : to step through your code, one line at a time.

Over : to step over a function call.

Out : to step out of a function.

Quit : to stop the execution of the entire program.

© Copyright 2022 Cyril Keime & Arnaud Bêche

CHAMPS DE BITS

© Copyright 2022 Cyril Keime & Arnaud Bêche

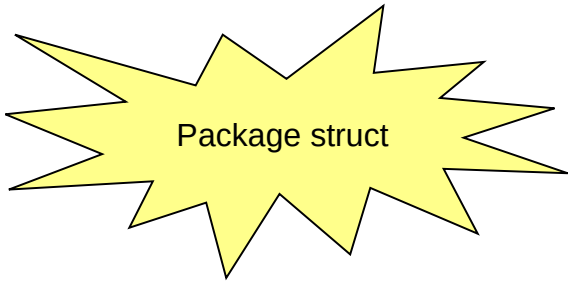
Généralités

- The core built-in types for manipulating binary data are bytes and bytearray. They are supported by memoryview which uses the buffer protocol to access the memory of other binary objects without needing to make a copy.
- The array module supports efficient storage of basic data types like 32-bit integers and IEEE754 double-precision floating values.
- Bytes objects are immutable sequences of single bytes.
- bytearray objects are a mutable counterpart to bytes objects.
- memoryview objects allow Python code to access the internal data of an object that supports the buffer protocol without copying. Built-in objects that support the buffer protocol include bytes and bytearray.
- struct — Interpret bytes as packed binary data

https://realpython.com/python-bitwise-operators/

© Copyright 2022 Cyril Keime & Arnaud Bêche

Exemples




```
s = 'hhl'  # short short long
print(f'{struct.calcsize(s)} bytes')
b = struct.pack('hhl', 1, 2, 3)
print(b)
v = struct.unpack('hhl', b'\x03\x00\x02\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00')
print(v)
```

```
def get_bit(value, bit_index):
    return value & (1 << bit_index)

def get_normalized_bit(value, bit_index):
    return (value >> bit_index) & 1

def set_bit(value, bit_index):
    return value | (1 << bit_index)

def clear_bit(value, bit_index):
    return value & ~(1 << bit_index)

def toggle_bit(value, bit_index):
    return value ^ (1 << bit_index)
```

1) Lire la documentation de la fonction suivante :

`bin()`

2) Ecriture d'un nombre binaire : (exemple)

`0b10100000`

3) Ne pas oublier qu'un int Python **n'est pas** un int C

© Copyright 2022 Cyril Keime & Arnaud Bêche

PYDOC

© Copyright 2022 Cyril Keime & Arnaud Bêche

Généralités

- à partir du shell Linux :

```
$ python3 -m pydoc -w fichier_xml_creer
$ firefox fichier_xml_creer.html
```

- à partir du shell Python :

```
>>> import fichier_xml_creer
>>> help(fichier_xml_creer)
```


© Copyright 2022 Cyril Keime & Arnaud Bêche

Exemple

fichier_xml_creer

index
cours_python/src/Complements/fichier_xml_creer.py

```
Créer un fichier XML

package lxml
```

Modules

lxml

Functions

run()
Démonstration de l'utilisation de lxml.

show_xml(xml)
Crée et affiche une chaîne de caractères au format XML.

write_xml(xml)
Crée un fichier XML.

Data

E = <lxml.builder.ElementMaker object>

© Copyright 2022 Cyril Keime & Arnaud Bêche

MARKDOWN

© Copyright 2022 Cyril Keime & Arnaud Bêche

Généralités

```python
import markdown


def md2raw(md, is_file=True):
    """Convert md to html (Markdown)"""
    if is_file:
        with open(md) as f:
            md = f.read()
    return markdown.markdown(
        md,
        output_format='html5',
        extensions=['markdown.extensions.tables'])
```

```
sudo pip install markdown
```


© Copyright 2022 Cyril Keime & Arnaud Bêche

Exemple

```
 md = """
# Littérature
**« Longtemps, je me suis couché de bonne heure »** est l'_incipit_
de Du côté de chez Swann, premier tome du roman À la recherche du
temps perdu de l'écrivain français Marcel Proust. Il s'agit là de
l'une des phrases les plus célèbres de la littérature française.

Pour certains, elle résume à elle seule toute la *Recherche*.
"""
```

Littérature

« Longtemps, je me suis couché de bonne heure » est l'*incipit* de Du côté de chez Swann, premier tome du roman À la recherche du temps perdu de l'écrivain français Marcel Proust. Il s'agit là de l'une des phrases les plus célèbres de la littérature française.

Pour certains, elle résume à elle seule toute la *Recherche*.

© Copyright 2022 Cyril Keime & Arnaud Bêche

MATPLOTLIB

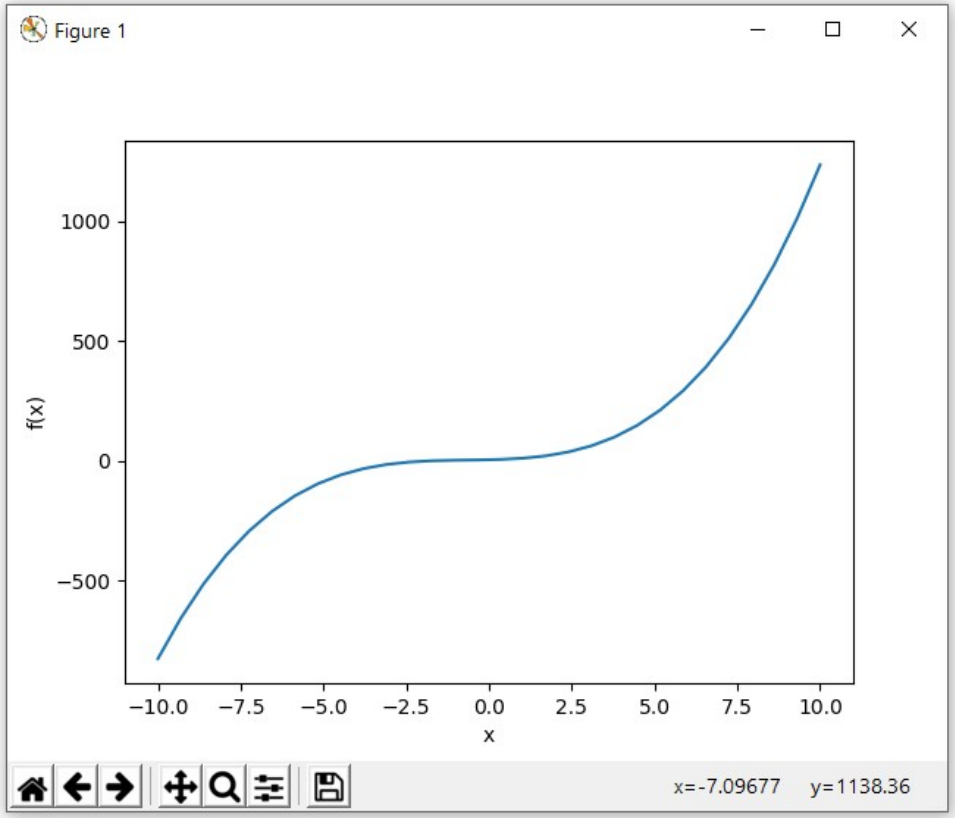
© Copyright 2022 Cyril Keime & Arnaud Bêche

Exemple

```
import numpy as np
import matplotlib.pyplot as plt

func = np.poly1d(np.array([1, 2, 3, 4]).astype(float))
x = np.linspace(-10, 10, 30)
y = func(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.show()
```



© Copyright 2022 Cyril Keime & Arnaud Bêche

PANDAS

© Copyright 2022 Cyril Keime & Arnaud Bêche

Exemple

```
import pandas as pd

data = pd.read_csv('fichier_csv.csv', sep=';')
print(data)

print(data.query('Id_unique_emetteur > 1105'))
```

```
      Id_unique_mode  Id_unique_emetteur  ...  Id_localisation_RefTact Comment
0                  1                  10  ...                        0  # Mode
1                  2                  11  ...                        0  # Mode
2                  3                  12  ...                        0  # Mode
3                  4                  13  ...                        0  # Mode
4                  5                  14  ...                        0  # Mode
...              ...                 ...  ...                      ...     ...
1095            1096                1105  ...                        0  # Mode
1096            1097                1106  ...                        0  # Mode
1097            1098                1107  ...                        0  # Mode
1098            1099                1108  ...                        0  # Mode
1099            1100                1109  ...                        0  # Mode

[1100 rows x 9 columns]
      Id_unique_mode  Id_unique_emetteur  ...  Id_localisation_RefTact Comment
1096            1097                1106  ...                        0  # Mode
1097            1098                1107  ...                        0  # Mode
1098            1099                1108  ...                        0  # Mode
1099            1100                1109  ...                        0  # Mode

[4 rows x 9 columns]
```


© Copyright 2022 Cyril Keime & Arnaud Bêche

PETITE IHM

© Copyright 2022 Cyril Keime & Arnaud Bêche

Exemple

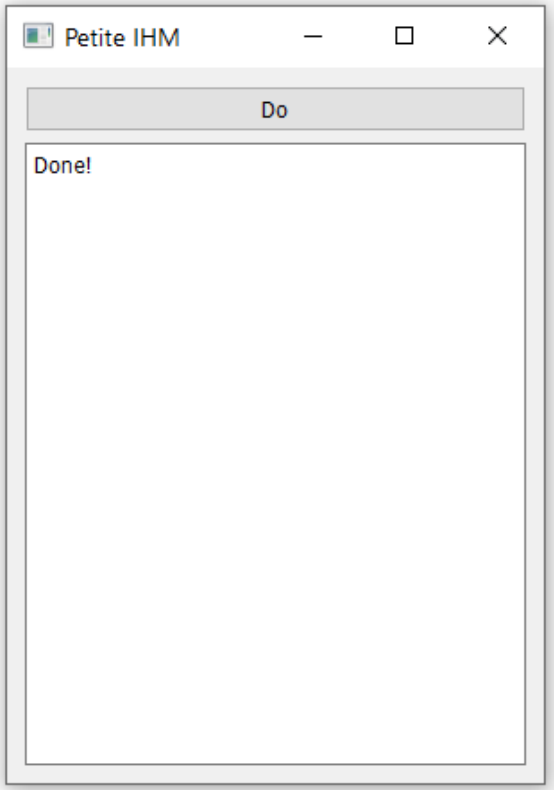
```
def run():
    """Application."""
    app = QApplication(sys.argv)
    mainwindow = TodoWindow()
    mainwindow.setMinimumSize(300, 700)
    mainwindow.show()
    app.exec()
```

```
class TodoWindow(QMainWindow):
    """Main window."""
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Petite IHM')
        self.mainwidget = TodoWidget(self)
        self.setCentralWidget(self.mainwidget)
```

```
class TodoWidget(QWidget):
    """Main widget."""
    def __init__(self, parent):
        super().__init__(parent)
        self.build_interface()

    def build_interface(self):
        # Widgets
        self.buDo = QPushButton('Do', self)
        self.teTsk = QTextEdit(self)
        self.teTsk.setReadOnly(True)
        self.teTsk.setLineWrapMode(QTextEdit.NoWrap)
        # Layouts
        self.mainsizer = QVBoxLayout()
        self.mainsizer.addWidget(self.buDo)
        self.mainsizer.addWidget(self.teTsk)
        self.setLayout(self.mainsizer)
        # Callbacks
        self.buDo.clicked.connect(self.do_task)

    def do_task(self):
        self.teTsk.append('Done!')
```



© Copyright 2022 Cyril Keime & Arnaud Bêche

ANNEXE

© Copyright 2022 Cyril Keime & Arnaud Bêche

Packages de référence

Numpy	calcul matriciel
SciPy	algorithmes
Matplotlib	tracé de graphiques 2D/3D
Pandas	exploitation de données tabulées
PyQt	interfaces graphiques
PyTorch	deep learning
Scikit Learn	toolbox optimisation et machine learning
Scikit Image	toolbox Image Processing
OpenCV	toolbox Video Processing
PyBind11	encapsulation de codes C/C++

© Copyright 2022 Cyril Keime & Arnaud Bêche