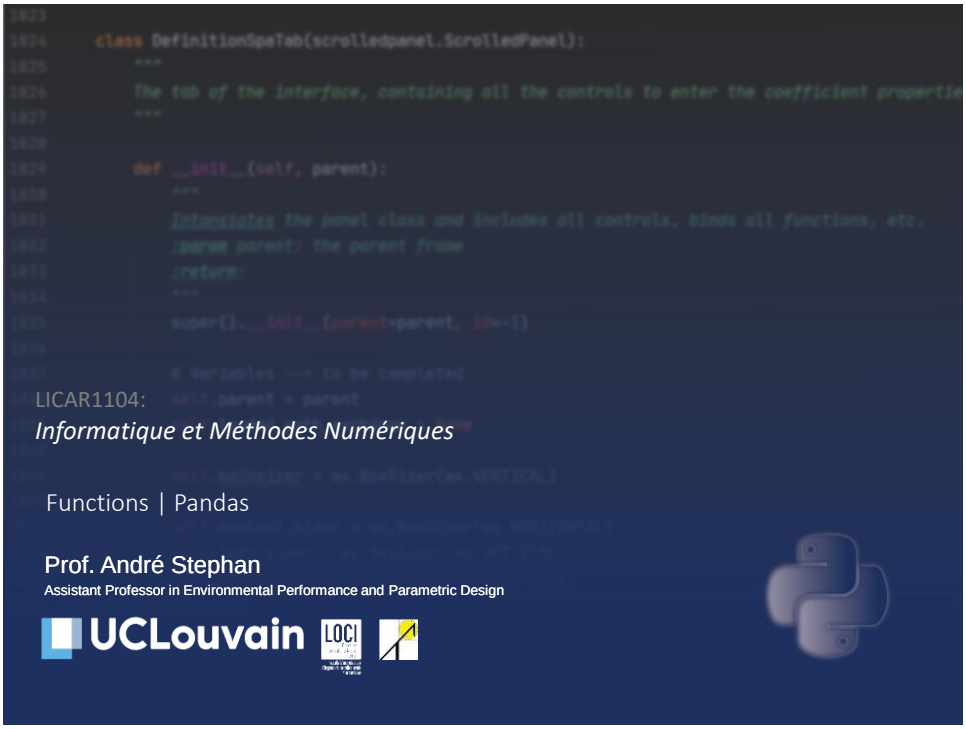


Seminar plan

- Seminar learning outcomes
- Active lecture: functions
- Activity one: calling functions
- Active lecture: introduction to pandas
- Activity two: calculate the embodied energy of a building with pandas

Seminar learning objectives

At the end of this seminar you will be able to:

- Define python and lambda functions
- Demonstrate a basic understanding of pandas dataframes and use them practically

Functions (in python)

- Group of related statements that perform a given task
- Enables a program to be divided into smaller parts
- Action/Verb → Function (e.g. calculate)
- Avoids repetition!!! (remember the PEP08 recommendations)

Functions in python – declaration - call

```
def function_name(argument_1, argument_2, ..., argument_n):
    """docstring"""
    statement_1
    statement_2
    ...
    statement_n
    (return var_1,var_2,...,var_n)
```

```
def power_2(number):
    try:
        return number**2
    except TypeError:
        print("Couldn't produce power 2 of " + str(number) + "
        which is a " + str(type(number)))
power_2(3)
>>> 9
power_2([5])
TypeError "Couldn't produce power 2 of [5] which is a <class
'list'>"
```


Functions in python - docstring

```
def power_2(number):
    """This function tries to produce the power 2 of a given
    number and returns an error if it fails"""
    try:
        return number**2
    except TypeError:
        print("Couldn't produce power 2 of " + str(number) + "
        which is a " + str(type(number)))


print(power_2.__doc__)

>>>"This function tries to produce the power 2 of a given
number and returns an error if it fails"
```

Functions in python - return

```
def absolute_value(number):
    """This function tries to produce the power 2 of a given
    number and returns an error if it fails"""
    try:
        if number >=0:
            return number
        else:
            return -number
    except TypeError:
    print("Couldn't produce the absolute value of " +
    str(number) + " which is a " + str(type(number)))
```

```
absolute_value(-15)
>>>15
absolute_value(124)
>>>124
```


Functions in python – good to know

```
def hi():
    print("Hi")
    return None
```

```
hi()
>>> Hi

hi
>>> <function hi at  0x000002EB47314C0  >
```

Functions in python – scope

```
def test():
    a = 20
    print('Value of a inside function ', a)

a = 40
test()
>>>'Value of a inside function 20'


print(a)

>>> 40
```


Lambda functions in python

- Short anonymous function
- Practical for very short expressions
- Multiply arguments, one expression

```
lambda arg_1,...,arg_n: <result to return>

z = lambda x,y: (x+y)**2
z(2,3)

>>>25
```

Activity one: call functions in python

https://www.wooclap.com/LICAR1104

LICAR1104:
Informatique et Méthodes Numériques

Break

Any questions, come and see me

- *"pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language"* (pandas.org)
- Started in 2008 (2009 open-access version)
- Wraps around numpy nd-**arrays**, turning them into **DataFrame**s (numpy 2-d array on steroids) with a plethora of additional methods
- Dataframes can receive an index and column labels,

- Pandas DataFrame is its core object
- Stores different types of variables! (like python data structures)
- Basically it's a matrix with 2-dimensions, and 1-n indexes, most often 1
- A DataFrame can have names for its rows (indexes) and columns (labels)
- Axis 0 is ? Rows
- Axis 1 is ? Columns

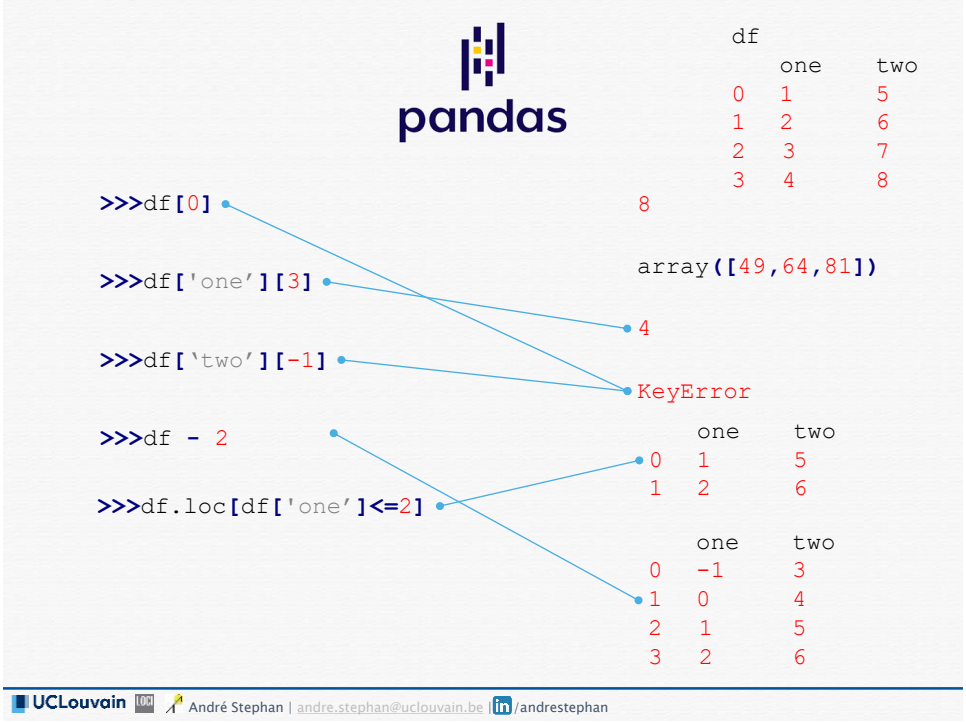
```
import pandas as pd
df = pd.DataFrame({'one': [1,2,3,4], 'two': [5,6,7,8]})
>>> df
```

```
   one  two
0    1    5
1    2    6
2    3    7
3    4    8
```


```
df
   one  two
0    1    5
1    2    6
2    3    7
3    4    8
```

```
df['one']
>>>
0    1
1    2
2    3
3    4
df.loc[0]
>>>  one 1
     two 5
     Name: 0, dtype: int64
df.iloc[0]
>>>  one 1
     two 5
     Name: 0, dtype: int64
```

```
df[['one','two' ]]
>>>
   one  two
0    1    5
1    2    6
2    3    7
3    4    8
```




```
df = pd.read_excel('filename.xlsx', index_col=0, header=0)

df = pd.read_csv('filename.csv', index_col=0, header=0)
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_excel.html

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

- Multiplying a dataframe by another:

```
df1.multiply(df2, axis='index')
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.multiply.html

Activity two: using pandas to calculate the embodied environmental flows of a building

- Download a reworked version of the original EPiC database https://doi.org/10.26188/5dc228ef98c5a as excel (.xlsx) from: **https://www.dropbox.com/s/0p5wfvjrmny8u8l/EPiC%20Database%202019.xlsx?dl=0**
- Put the file in your working python directory
- **Read the excel file as a dataframe (named *epic_df*), use only columns 2,3,4,5,6, use column 0 as the index_col, use excel's row 6 as the header**
- Download the bill of quantities of Belgian Passive House from: **https://www.dropbox.com/s/cxyp2wd8hgozdns/ph_boq.xlsx?dl=0**
- **Read the excel file as a dataframe (named *boq*), use column 0 as the index_col, use row 0 as the header**
- **Calculate the embodied energy, water and greenhouse gas emissions of each material in the building (note: materials in the boq are all in the correct units)**
- **Calculate the embodied energy, water and greenhouse gas emissions of the whole building.**

15 mins

15 mins

Activity two: using pandas to calculate the embodied environmental flows of a building

- **Read the excel file as a dataframe (named epic_df), use columns 2,3,4,5,6, use column 0 as the index_col, use excel's row 6 as the header**

```
import pandas as pd
epic_df = pd.read_excel('EPiC Database 2019.xlsx',
header=5, usecols=[2,3,4,5,6], index_col=0)
```


Activity two: using pandas to calculate the embodied environmental flows of a building

- **Read the excel file as a dataframe (named boq), use column 0 as the index_col, use row 0 as the header**

```
boq = pd.read_excel('ph_boq.xlsx', index_col=0)
```

Activity two: using pandas to calculate the embodied environmental flows of a building

- **Calculate the embodied energy, water and greenhouse gas emissions of each material in the building**

Keeps only the relevant columns for the calculation

Filters out all unnecessary materials from the database

```
res =
epic_df.loc[boq.index][['EE','EW','EGHG']]
.multiply(boq['Delivered quantity'],
axis=0)
```

Multiplies by the quantities of materials in the building

Activity two: using pandas to calculate the embodied environmental flows of a building

- **Calculate the embodied energy, water and greenhouse gas emissions of the whole building**

`sum(res.EE)`		**1971319.8 MJ = 1 971 GJ**
`sum(res.EW)`		**4747755.4 L = 47 478 kL**
`sum(res.EGHG)`		**150339.3 $kgCO_2e$ = 150 tCO_2e**

```
sum(res['EE'])
sum(res['EW'])
sum(res['EGHG'])
```

What if you had to write a function that…

- Receives a boq file formatted like the one you just saw
- Automatically calculates the Embodied energy, water and greenhouse gas emissions
- Returns a dataframe with materials as the index and three columns of EE, EW, and EGHG as well as a dictionary containing IEE, IEW, and IEGHG as keys, and the total initial embodied energy, water and greenhouse gas emissions as values

Function that calculates initial embodied flows from the EPiC database for a given boq file

```
def calc_embodied_flows(boq_file:str,epic_db_file_path:str =
'EPiC Database 2019.xlsx'):
    epic_df = pd.read_excel(epic_db_file_path, header=5,
    usecols=[2,3,4,5,6], index_col=0)
    boq = pd.read_excel(boq_file, index_col=0)
    res =
    epic_df.loc[boq.index][['EE','EW','EGHG']].multiply(boq['D
    elivered quantity'], axis=0)
    total_ef = {flow: sum(res[flow]) for flow in
    ['EE','EW','EGHG']}
    return res, total_ef

a,b = calc_embodied_flows('ph_boq.xlsx')
```

UCLouvain André Stephan | andre.stephan@uclouvain.be | /andrestephan

Next time: Week 12

Basics of:

Object-oriented programming