

Module 61-12: Option GIS-Python

Introduction to Python

hes.
so
business.

Jean-Paul Calbimonte

School of Management

Bachelor of Science HES-SO (BSc) in Business
Information Technology



> GIS for developers

Why Python for geodata?



- **Free:** no added costs for licensing
- **For coders:** fully programmable geodata manipulation
- **Modular:** libraries adapted to different use-cases
- **Efficiency:** optimized for Big Data analytics
- **Extensibility:** possibility to extend or reuse multiple libraries
- **Flexibility:** options for lots of formats/standards/approaches
- **Open Source:** code reuse/reproducibility/open science
- **Integration:** supported by other tools as QGIS/ArcGIS etc.

> How do we run python?

Different options

In this course:



Option 1. Online notebook: Jupyter Python in renkulab.io

Option 2. Online IDE: VSCodium in renkulab.io

Option 3. VS Code in Local installation

> Options 1 & 2: renkulab.io

Log in with your hevs edu-ID account



Welcome to RenkuLab!

 By continuing, you agree to the updated [RenkuLab Terms of Service](#) and acknowledge that the updated [Privacy Policy](#) applies to you. Please take a moment to review these changes.

Sign up or Log in with

edu-ID

GitHub

ORCID

OR

Log in with your RenkuLab account

Email

Password

[Forgot Password?](#)

Sign In

SWITCH edu-ID

Help EN

Log in to: renkulab.io

Service description:

This is a public beta instance of the Renku platform for reproducible data science operated by the Swiss Data Science Center.

SWITCH edu-ID

E-mail:

Password:

Create account

Login

[Forgot password?](#)

[Options for personal data protection](#)

SWITCH

[About](#) / [Terms of Use](#) / [Legal Notice](#) / [Imprint](#)

> renkulab.io

Got to this project:

<https://renkulab.io/p/jean-paul.calbimonte/gis-dev>

The screenshot shows the RenkuLab interface for a project named 'GIS-Dev'. At the top left is the 'renku' logo. Below it is a project card with a network diagram icon, the title 'GIS-Dev', and a link to 'Add description'. There are two tabs: 'Overview' (selected) and 'Settings'. The main content area is titled 'Sessions 2' and contains two session launcher cards. The first card is for a 'Python environment' with a 'Launch' button. The second card is for 'JupyterLab' with a 'Launch' button circled in red. Below the sessions are sections for 'Data 0' and 'Code Repositories 1', with the 'gis-python' repository having a 'Read & write' status and an 'Edit' button.

Option 1: Launch a Jupyter Notebook session

> renkulab.io Option 1: using Jupyter Lab

Renku UI - Reproducible Data Science | Open Research | Renku

← Back [document] [stop] [trash] [refresh] [warning] GIS-Dev / JupyterLab

File Edit View Run Kernel Git Tabs Settings Help

Launcher

Notebook

Python3

Console

Python3

Other

Terminal Text File Markdown File Python File Show Contextual Help

gis-python 11m ago
lost+found 11m ago

Ready to open the notebooks

> renkulab.io Option 1: using Jupyter Lab

Renku UI - Reproducible Data Science | Open Research | Renku

← Back [Icons] GIS-Dev / JupyterLab

File Edit View Run Kernel Git Tabs Settings Help

Launcher x 01-datatypes.ipynb +

+ × [Icons] Markdown v [Icons] git

Python basics

Basic operators & math

Befre starting, you can input comments using #

```
[ ]: # some comments here  
34 # this is a number
```

Simple expressions results in a value. As in the following arithmetic operations:

```
[ ]: 1+1+34  
[ ]: 3*4.5  
[ ]: 2-3.5  
[ ]: 10/3
```

Other mathematical operatos include the int divison // , modulo % or power **

```
[ ]: 10 // 3
```

Run each cell

> renkulab.io

Got to this project:

<https://renkulab.io/p/jean-paul.calbimonte/gis-dev>

The screenshot shows the project page for 'GIS-Dev' on renkulab.io. At the top left is the 'renku' logo. Below it is a project card with a thumbnail image of a map, the title 'GIS-Dev', and a link to 'Add description'. Below the card are two tabs: 'Overview' (selected) and 'Settings'. The main content area is titled 'Sessions 2' and contains two session launcher cards. The first card is for a 'Python environment' with a 'Launch' button circled in red. The second card is for 'JupyterLab' with a 'Launch' button. Below the sessions are two other sections: 'Data 0' and 'Code Repositories 1'. The 'Code Repositories' section shows a repository named 'gis-python' with a 'Read & write' status and an 'Edit' button.

Option 2: Launch a VS Codium session

> renkulab.io Option 2: using VS Codium online

The screenshot shows the VS Codium online interface. At the top, there's a navigation bar with "Renku UI - Reproducible Data Science | Open Research | Renku" and a breadcrumb trail "GIS-Dev / Python environment". Below this is a toolbar with icons for back, home, refresh, and warning, along with a search bar containing "work". The main area is divided into three panes: a file explorer on the left, a file browser at the top, and a code editor. The file explorer shows a tree structure under "WORK" with folders like ".local", ".venv", ".vscode", "gis-python", and "notebooks". The file browser shows open files: "01-datatypes.ipynb", "README.md", "books.py", "mats.py", and "08-files.ipynb". The code editor displays the content of "01-datatypes.ipynb", which includes a title "Python basics", a subtitle "Basic operators & math", and a code cell with the following content:

```
Befre starting, you can input comments using #  
  
# some comments here  
# this is a number  
[ ]
```

Below the code cell, there's a text block: "Simple expressions results in a value. As in the following arithmetic operations:" followed by two code cells:

```
[ ] 1+1+34  
[ ] 3*4.5
```

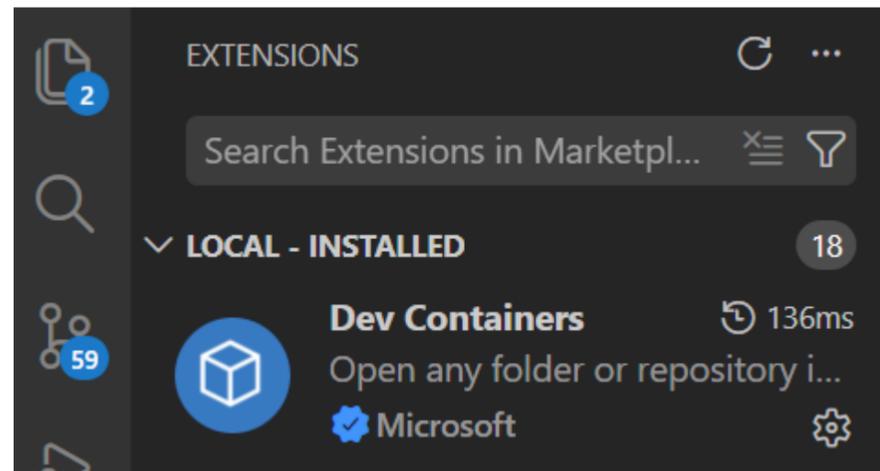
A red arrow points from the text "Run each cell" to the "Execute Cell" button (a play icon) next to the first code cell.

Run each cell

> Option 3: VSCode on your machine

You will need:

- VSCode
with Dev Containers extension



- Docker Desktop

> Option 3: VSCode on your machine

Got to this project:

<https://renkulab.io/p/jean-paul.calbimonte/gis-dev>

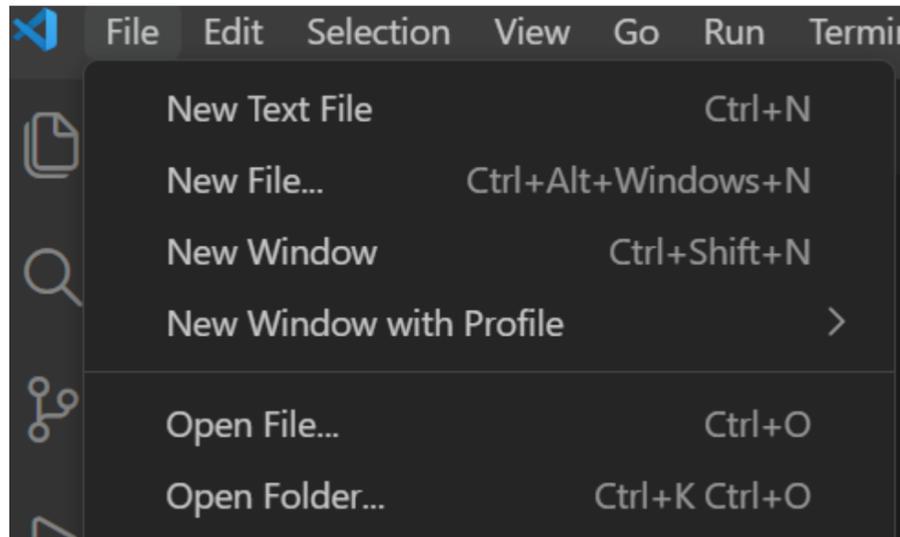
The screenshot shows the Renku UI interface. The main page displays the project 'GIS-Dev' with an 'Add description' link and tabs for 'Overview' and 'Settings'. Under 'Sessions', there are two entries: 'Python environment' and 'JupyterLab', both with 'Build succeeded' status. At the bottom, there are sections for 'Data' and 'Code Repositories'. The 'Code Repositories' section is circled in red and shows a repository named 'gis-python' with 'Read & write' permissions.

A modal window is open on the right, showing details for the 'gis-python' repository. It includes the repository name, the URL <https://github.com/jpcik/gis-python>, and the Git command: `git clone https://github.com/jpcik/gis-python.git`. The permissions section shows 'Read & write' selected, with 'Pull: Yes' and 'Push: Yes' options. A red arrow points from the Git command in the modal to the text 'Git clone the repository in your machine'.

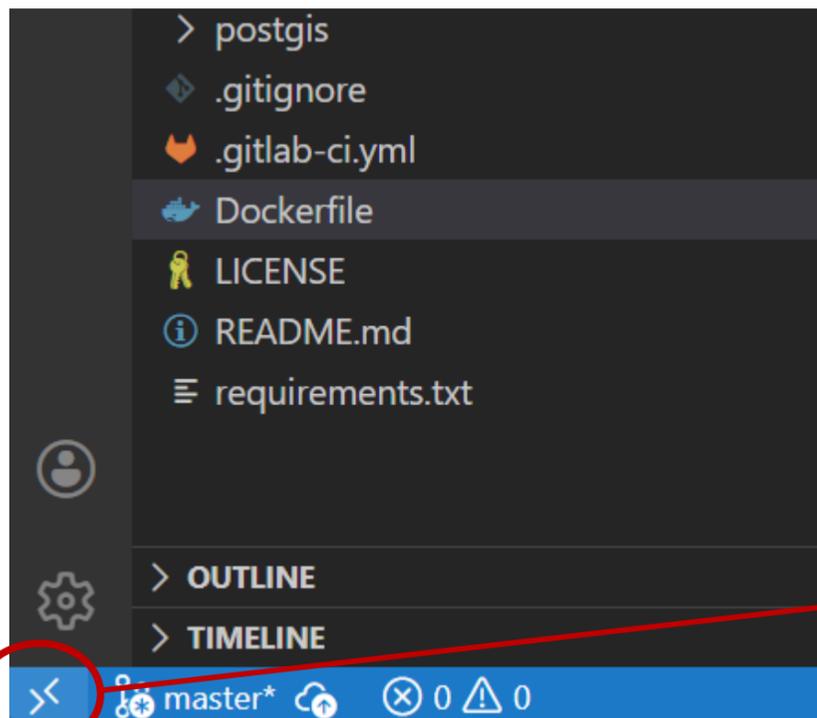
Git clone the repository in
your machine

> Option 3: VSCode on your machine

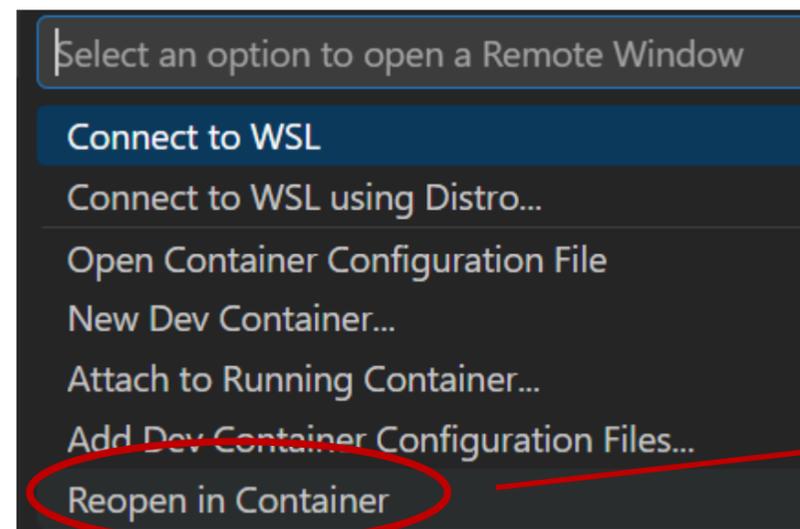
Open the folder in VSCode



Click on the bottom corner button



click



Then reopen
in container

> Option 3: VSCode on your machine

You are ready to go

The screenshot shows the Visual Studio Code (VS Code) interface running in a Dev Container. The Explorer sidebar on the left shows a project structure for 'GIS-PYTHON [DEV CONTAINER: PYTHON DEV ...]' with folders like '.devcontainer', '.renku', 'data', 'django', 'cantons', 'data', 'geoweb', 'demo.html', 'docker-compose.yml', 'Dockerfile', 'init.sh', 'requirements.txt', 'leaflet', and 'notebooks'. The 'notebooks' folder is expanded to show '01-PythonIntro', which contains several Jupyter Notebook files, including '01-datatypes.ipynb' which is currently open.

The main editor area displays the '01-datatypes.ipynb' notebook. The title is 'Python basics' and the subtitle is 'Basic operators & math'. The text reads: 'Befre starting, you can input comments using #'. Below this, there are three code cells:

```
[ ] # some comments here
34 # this is a number
```

Simple expressions results in a value. As in the following arithmetic operations:

```
[ ] 1+1+34
```

```
[ ] 3*4.5
```

> Learning Python

Basics of Python

> Basic operators

$1 + 1$

2

$3 * 4.5$

13.5

$2 - 3.5$

-1.5

$10 / 3$

3.3333333333333335

$3 ** 2$

9

basic arithmetic operations

> Math functions

```
import math
```

```
math.sqrt(81)
```

```
9.0
```

```
math.pi
```

```
3.141592653589793
```

```
math.sin(math.pi/2)
```

```
1.0
```

import library

imported function

> Variables & Basic types

```
a=1
```

```
b=2.4      int
```

```
a+b      float
```

```
3.4
```

```
a==b
```

```
bool
```

```
False
```

```
a==1
```

```
True
```

```
st='we like GIS'  str
```

```
print(st)
```

```
we like GIS
```

```
b=st
```

```
print(b)
```

```
we like GIS
```

```
print('This is number',a,'followed by this string:',b)
```

```
This is number 1 followed by this string: we like GIS
```

> Basic types

```
type (a)
```

```
int
```

```
type (st)
```

```
str
```

```
type (3.42)
```

```
float
```

```
type (a==2)
```

```
bool
```

int
str
float
bool



> Lists

```
[3, 4, 6, 2, 1]
```

unidimensional

```
[3, 4, 6, 2, 1]
```

```
[[2, 3], [5, 6], [4, 3]]
```

multidimensional

```
[[2, 3], [5, 6], [4, 3]]
```

```
list=[3, 4.5, 6, 2, 2.1]
```

```
len(list)
```

```
5
```

```
list[2]
```

```
6
```

```
list[-1]
```

```
2.1
```

indices

```
type(list[1])
```

```
float
```

types

> Lists

```
list=[3,4.5,6]
del list[2]
print(list)
[3, 4.5]
```

remove item

```
list.append(44)
print(list)
[3, 4.5, 44]
```

append item

```
list.reverse()
print(list)
[44, 4.5, 3]
```

reverse list

```
list.sort()
print(list)
[3, 4.5, 44]
```

sort list

```
list[0]=55
print(list)
[55, 4.5, 44]
```

replace item

> Lists & Loops

```
string='strange'  
len(string)  
7
```

```
print(string[2])  
r
```

```
for ch in string:  
    print(ch)
```

```
s  
t  
r  
a  
n  
g  
e
```

iterate over chars

```
count=0  
for ch in string:  
    print(ch, count)  
    count=count+1  
print(count)  
s 0  
t 1  
r 2  
a 3  
n 4  
g 5  
e 6  
7
```

> Loops

```
range(4)
```

```
range(0, 4)
```

```
for i in range(4):  
    print(i)
```

```
0
```

```
1
```

```
2
```

```
3
```

iterate range

```
for i in range(2, 14, 3):  
    print(i)
```

```
2
```

```
5
```

```
8
```

```
11
```

range with step

```
name='Aladdin'
```

```
for i in range(len(name)):  
    print(name[i])
```

```
A
```

```
l
```

```
a
```

```
d
```

```
d
```

```
i
```

```
n
```

> Conditions

```
speed=80

if speed > 100:
    print('too fast')
elif speed > 80 and speed <=100 :
    print('speed ok')
else:
    print('too slow')

too slow
```

← if

← elif

← else

> Conditions

```
exam1 = 3.5  
exam2 = 'B'  
  
if (exam1 >=4 and exam2 == 'A') :  
    print('grades are great')  
elif exam1 < 4 and exam2 == 'B':  
    print('grades are poor')  
else:  
    print('grades are mixed')
```

grades are poor

> Functions

```
def calculateArea (length, height) :  
    return length*height
```

```
calculateArea (5, 4)
```

```
20
```

```
calculateArea ('a', 4)
```

```
'aaaa'
```

> Functions

```
def countLetters(strings, letter):  
    count=0  
    for str in strings:  
        count=count+str.count(letter)  
    return count
```

```
countLetters(['day', 'pasta', 'lasagna'], 'a')  
6
```

> Classes

```
class Vehicle(object):  
    wheels=0  
  
    def __init__(self, wheels, maxSpeed=0):           'constructor'  
        self.wheels=wheels  
        self.maxSpeed=maxSpeed  
  
    def fasterThan(self, otherVehicle):             method  
        return self.maxSpeed > otherVehicle.maxSpeed  
  
v1=Vehicle(2)                                       instantiation  
print(v1.maxSpeed)  
0
```

> Classes

```
class Bike(Vehicle):  
    def __init__(self, maxSpeed=0):  
        self.wheels=2  
        self.maxSpeed=maxSpeed
```

```
b1=Bike(30)
```

```
b1.wheels
```

```
b1.maxSpeed
```

```
30
```

```
b2=Vehicle(4, 80)
```

```
b1.fasterThan(b2)
```

```
False
```

> Error

```
def calculateArea(length,height):  
    return length*height
```

```
calculateArea('a','dsd')
```

```
-----  
-----  
TypeError Traceback (most recent call last)  
<ipython-input-2-2ccc00b376b5> in <module>  
----> 1 calculateArea('a','dsd')
```

```
<ipython-input-1-bf7816c63e4e> in calculateArea(length,  
height)
```

```
    1 def calculateArea(length,height):  
----> 2 return length*height
```

```
TypeError: can't multiply sequence by non-int of type 'str'
```

> Assertions

```
def calculateArea(length,height):  
    assert length > 0 , 'length must be positive'  
    assert height > 0 , 'height must be positive'  
    assert type(length) == float  
calculateArea(-3.0,2)  
-----  
(-----  
(AssertionError Traceback (most recent call last)  
<ipython-input-15-f962da30df2f> in <module>  
----> 1 calculateArea(-3.0,2)  
  
<ipython-input-12-f5dade16329c> in calculateArea(length, height)  
      3 #assert type(height) == float  
      4  
----> 5 assert length > 0 , 'length must be positive'  
      6 return length*height  
AssertionError: length must be positive
```

> Questions?

hes.
so
you.

School of Management
Route de la Plaine 2
3960 Sierre

hevs.ch/heg



Thank you for your attention.

swissuniversities

