

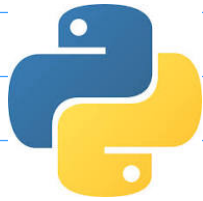
Python vs. Outras linguagens de programação

- Python:

- Interpretada

- "Devagar"

- Não-tipada:



Exemplo:

```
i = 0
```

```
a = 0.0
```

- Sintaxe simples

- Alocação automática da memória

- Fácil de Utilizar:

- "democratização" da programação

- prototipagem rápida de código

- Sistemas de gerenciamento de pacotes (bibliotecas):

Exemplo instalação de pacotes:

a) `pip install numpy`

b) `conda install numpy`

c) `conda install numpy=2.3.1`

C, C++, FORTRAN:

- Compilado

- "Rápido"

- Tipadas



Exemplo:

```
int i = 0;
```

```
float a = 0.0;
```

- Sintaxe complexa

- Gerenciamento manual memória

- Difícil de utilizar:

- Instalação de um compilador

- Difícil compreensão

- Declaração das variáveis

- Gerenciamento de memória

- etc...

- Bibliotecas:

- Download manual

- Compilação manual

- Checar/Resolver dependências com outras bibliotecas

- "Linkar" bibliotecas na compilação

Exemplo (by ChatGPT) mostrando a diferenças entre Python e C++:

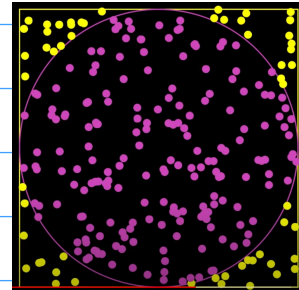
Monte Carlo Integration - <https://www.youtube.com/watch?v=8276ZswRw7M>



Sure, I can help you with that! The Monte Carlo method is a statistical technique that uses random sampling to estimate numerical results. To calculate the area of a circle using the Monte Carlo method, you can follow these steps:

1. Generate random points within a bounding square that contains the circle.
2. Check if each point falls within the circle using the circle equation: $x^2 + y^2 \leq r^2$.
3. Calculate the ratio of points within the circle to the total number of points generated. This ratio is proportional to the ratio of the circle's area to the square's area.
4. Use the ratio to estimate the area of the circle.

Here's a Python code that does this using the NumPy package:



```
import numpy as np

def monte_carlo_circle_area(radius, num_points):
    # Generate random points within a square that contains the circle
    points = np.random.uniform(-radius, radius, size=(num_points, 2))

    # Calculate the distances of points from the origin
    distances = np.linalg.norm(points, axis=1)

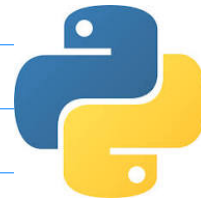
    # Count the points inside the circle
    points_inside_circle = np.sum(distances <= radius)

    # Calculate the estimated circle area
    circle_area = (points_inside_circle / num_points) * (4 * radius**2)

    return circle_area

# Define the radius and the number of random points
radius = 1.0
num_points = 1000000

# Calculate the estimated circle area using Monte Carlo method
estimated_area = monte_carlo_circle_area(radius, num_points)
print("Estimated Circle Area:", estimated_area)
```



GSL:
GNU Scientific Library (GSL)
<https://www.gnu.org/software/gsl/>

```
#include <iostream>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_math.h>

int main() {
    const double radius = 1.0;
    const size_t num_points = 1000000;

    // Initialize the random number generator
    gsl_rng* rng = gsl_rng_alloc(gsl_rng_default);

    // Initialize variables to count points inside the circle
    size_t points_inside_circle = 0;

    // Generate and check random points
    for (size_t i = 0; i < num_points; ++i) {
        double x = gsl_rng_uniform(rng) * 2 * radius - radius;
        double y = gsl_rng_uniform(rng) * 2 * radius - radius;

        if (x*x + y*y <= radius*radius) {
            points_inside_circle++;
        }
    }

    // Calculate the estimated circle area
    double square_area = 4 * radius * radius;
    double circle_area = (static_cast<double>(points_inside_circle) / num_points) * square_area;

    std::cout << "Estimated Circle Area: " << circle_area << std::endl;

    // Free the random number generator
    gsl_rng_free(rng);

    return 0;
}
```

Ainda é necessário baixar a biblioteca gsl, compilá-la e linkar no compilador!

```
g++ -I/path/to/gsl/headers -c your_program.cpp -o your_program.o
g++ your_program.o -o your_program -lgsl -lgslcblas -lm
./your_program
```

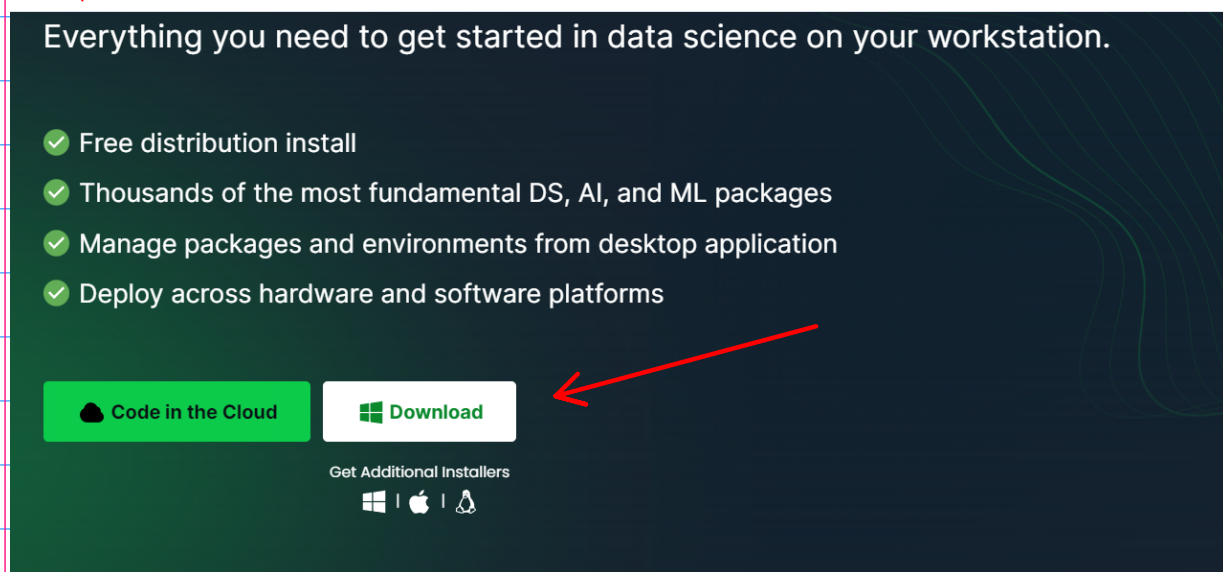
Download e Instalação do Python Anaconda

Poderíamos muito bem usar o Python "padrão", disponível em www.python.org.

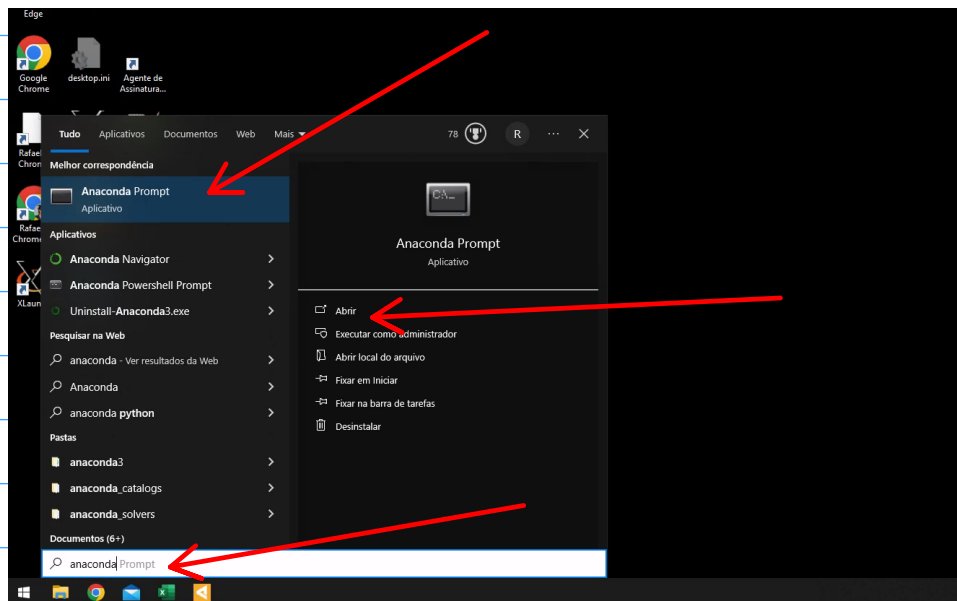
No entanto, para utilização em aplicações científicas, é aconselhado a utilização da distribuição disponibilizada pela Anaconda Inc.

Ela possui um ambiente para pacotes mais adequado para nossas aplicações, contando com uma ferramenta nativa para download de pacotes (conda) e virtualização de ambientes.

<https://www.anaconda.com/download>



Depois de instalado, cheque se está tudo OK. Procure por 'anaconda' na barra de busca do Windows:



Clique em "Abrir". Se tudo estiver certo, aparecerá um terminal:

1. Digite `python` e aperte Enter. Esse console deverá aparecer.
2. Esse é o console do Python. Experimente realizar um cálculo qualquer.
3. Para sair, digite `exit()` ou pressione `Ctrl+D`

A screenshot of the Anaconda Prompt terminal window. The window title is 'Anaconda Prompt'. The prompt is '(base) C:\Users\rafaelfc>'. The user has entered the command 'python', which has opened a Python 3.11.4 shell. The prompt is now '>>>'. The user has entered the following code: `a = 2`, `b = 3`, `c = a + b`, and `print(c)`. The output of the code is '5'. The user has then entered `exit()`, and the prompt has returned to '(base) C:\Users\rafaelfc>'. A red bracket on the left side of the image groups the first three steps of the list above with this screenshot.

Agora já temos o Anaconda Python instalado. Precisamos apenas ajustar mais algumas configurações. Vamos ver esses detalhes nas próximas páginas...

Anaconda Python: Criação de um ambiente virtual

Como mencionando anteriormente, é muito simples baixar e utilizar pacotes em Python. Isso é ótimo, pois é muito simples testar e compartilhar bibliotecas. Entretanto, isso pode resultar em problemas, pois a depender do projeto, os pacotes podem diferir em suas versões.

Como forma de aliviar esse problemas, e isolar as dependências de cada projeto, são utilizados ambientes virtuais.

A ambiente padrão do Python é o `virtualenv` (venv), mas aqui como estamos utilizando a distribuição da Anaconda, vamos utilizar o `conda env`.

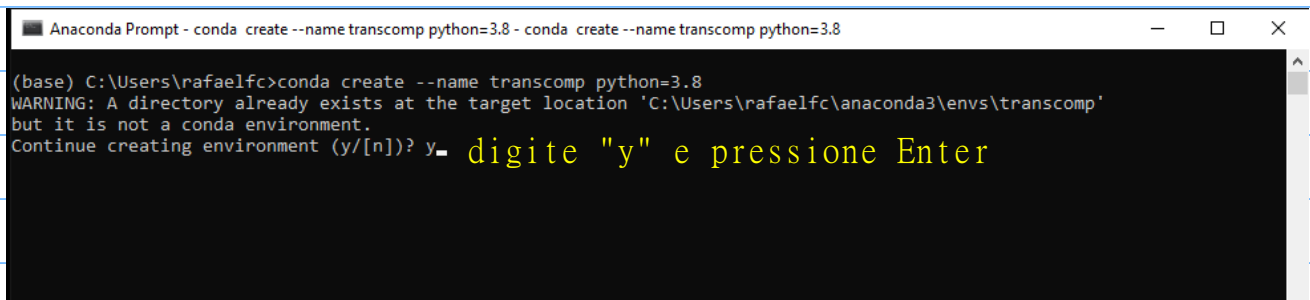
Vamos agora criar o nosso ambiente virtual, que será utilizado ao longo da disciplina.

Primeiro, abrimos o terminal do "Anaconda prompt" e digitamos o seguinte comando:

`conda create --name cfd python=3.11`

versão
do
Python

comando para criar o amb. virtual nome do amb. virtual



```
Anaconda Prompt - conda create --name transcomp python=3.8 - conda create --name transcomp python=3.8
(base) C:\Users\rafaelfc>conda create --name transcomp python=3.8
WARNING: A directory already exists at the target location 'C:\Users\rafaelfc\anaconda3\envs\transcomp'
but it is not a conda environment.
Continue creating environment (y/[n])? y_ digite "y" e pressione Enter
```

```
Anaconda Prompt - conda create --name transcomp python=3.8 - conda create --name transcomp python=3.8

environment location: C:\Users\rafaelfc\anaconda3\envs\transcomp

added / updated specs:
- python=3.8

The following packages will be downloaded:

package ----- | ----- build -----
openssl-3.0.10 | h2bbff1b_2 | 7.4 MB
----- | -----
Total: 7.4 MB

The following NEW packages will be INSTALLED:

ca-certificates pkgs/main/win-64::ca-certificates-2023.05.30-haa95532_0
libffi pkgs/main/win-64::libffi-3.4.4-hd77b12b_0
openssl pkgs/main/win-64::openssl-3.0.10-h2bbff1b_2
pip pkgs/main/win-64::pip-23.2.1-py38haa95532_0
python pkgs/main/win-64::python-3.8.17-h1aa4202_0
setuptools pkgs/main/win-64::setuptools-68.0.0-py38haa95532_0
sqlite pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0
vc pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel pkgs/main/win-64::wheel-0.38.4-py38haa95532_0

Proceed ([y]/n)?
```

digite "y" e pressione Enter

Se tudo estiver certo, o terminal apresentará uma mensagem similar a essa mostrada aqui abaixo:

```
Anaconda Prompt - conda create --name transcomp python=3.8

setuptools pkgs/main/win-64::setuptools-68.0.0-py38haa95532_0
sqlite pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0
vc pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel pkgs/main/win-64::wheel-0.38.4-py38haa95532_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
openssl-3.0.10 | 7.4 MB | 0% D
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/openssl-3.0.10-h2bbff1b_2.conda HTTP/1.1" 200 7786531

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

#
# To activate this environment, use
#
#     $ conda activate transcomp
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\rafaelfc>
```

A sintaxe de utilização é simples, como mostrado na janela acima.

Para ativar o ambiente: conda activate transcomp

Para desativar o ambiente: conda deactivate

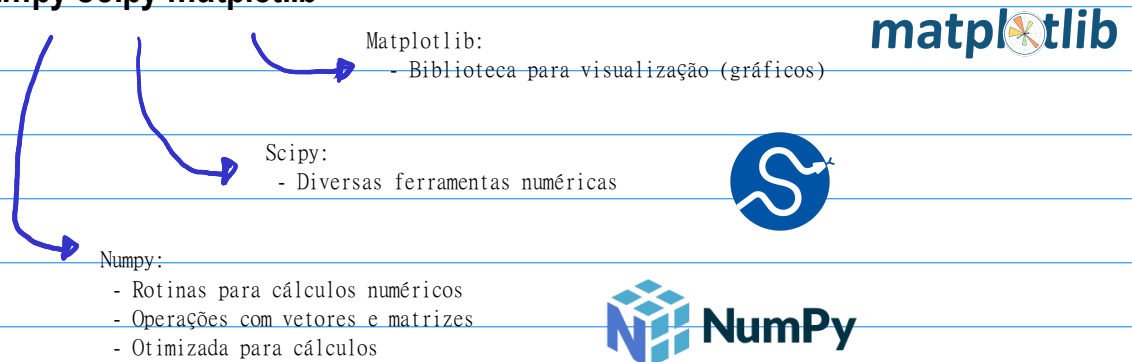
Agora vamos ativar o ambiente e configurar os pacotes necessários:

```
(base) C:\Users\rafaelfc>conda activate transcomp  
(transcomp) C:\Users\rafaelfc>
```

Note que quando ativamos o ambiente, há a mudança de (base) para (cfd).

Vamos agora instalar as primeiras bibliotecas/pacotes via conda:

conda install numpy scipy matplotlib



Para testar se está tudo certo, vamos abrir um console e importar as bibliotecas:

```
Anaconda Prompt - conda create --name transcomp python=3.8 - conda install numpy scipy matplotlib - python  
(transcomp) C:\Users\rafaelfc>python abrindo o console  
Python 3.8.17 (default, Jul 5 2023, 20:44:21) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import numpy as np importando numpy  
>>> import scipy importando scipy  
>>> import matplotlib.pyplot as plt importando matplotlib  
>>> _  
  
Se nenhuma mensagem aparecer, deu tudo certo!
```

Ok, agora já temos o nosso ambiente virtual criado e organizado.

Precisamos agora de uma ferramenta para desenvolver e testar os nossos códigos. Para isso utilizaremos um Ambiente de Desenvolvimento Integrado (IDE, Integrated Development Environment)

Nesse curso utilizaremos o PyCharm.

Outras opções são o Spyder, VSCode, etc...

A seguir vamos configurar o ambiente de desenvolvimento no PyCharm, partindo de nosso ambiente virtual criado a partir das ferramentas do Anaconda (conda env).



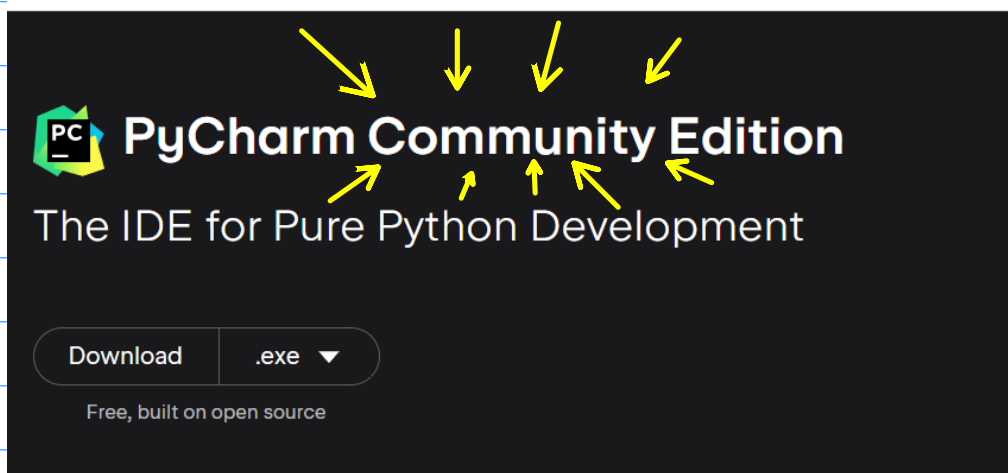
PyCharm: Configurando o ambiente virtual

Para baixar o PyCharm, precisamos acessar o seguinte endereço:

<https://www.jetbrains.com/pycharm/download/?section=windows>

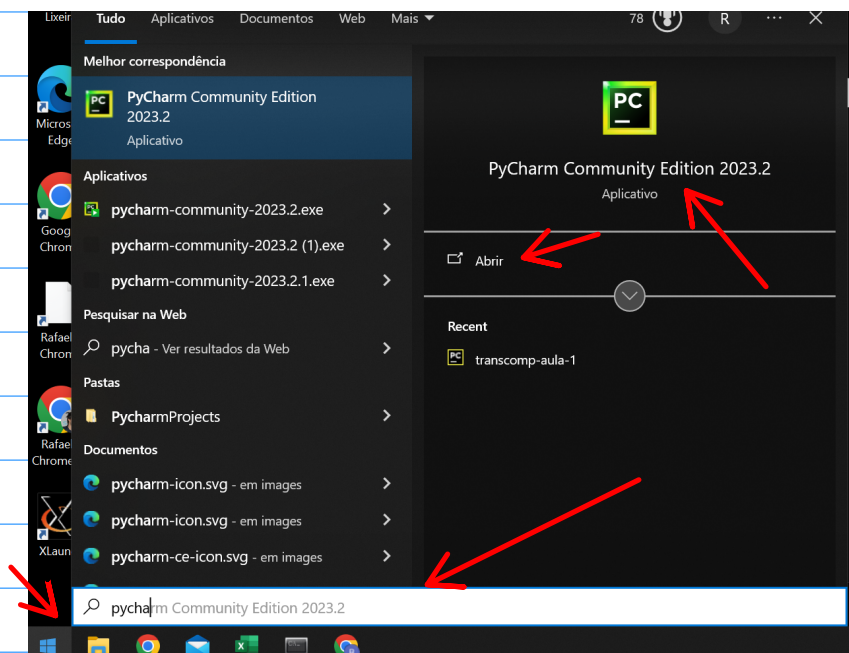
Importante! Certifique-se que está instalando a "Community Edition", que é gratuita.*

PyCharm



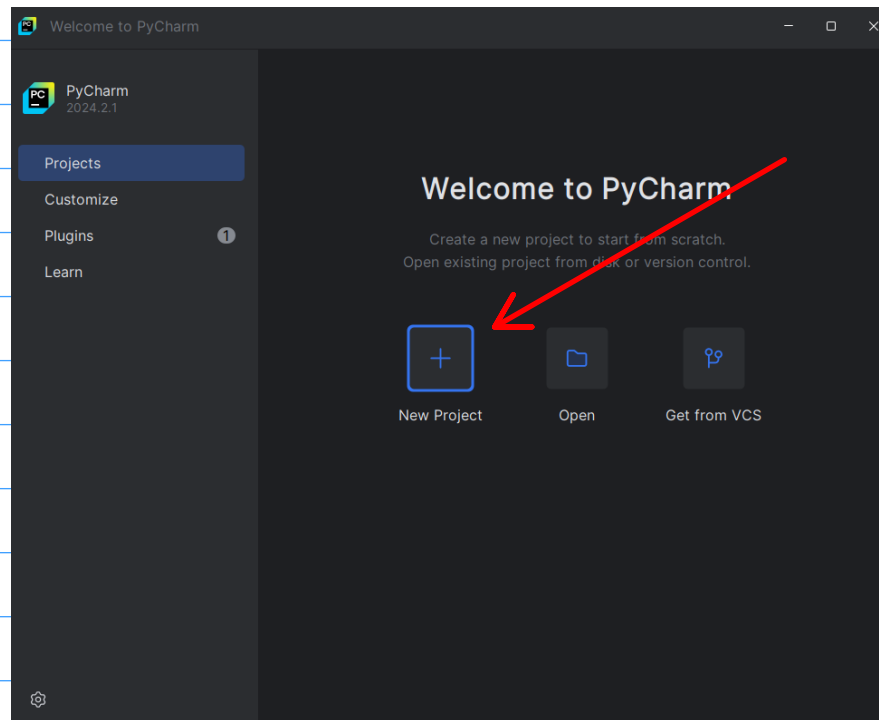
Feito o download, vamos configurar a IDE.

Para abrir o PyCharm, busque-o na barra de ferramentas do Windows.

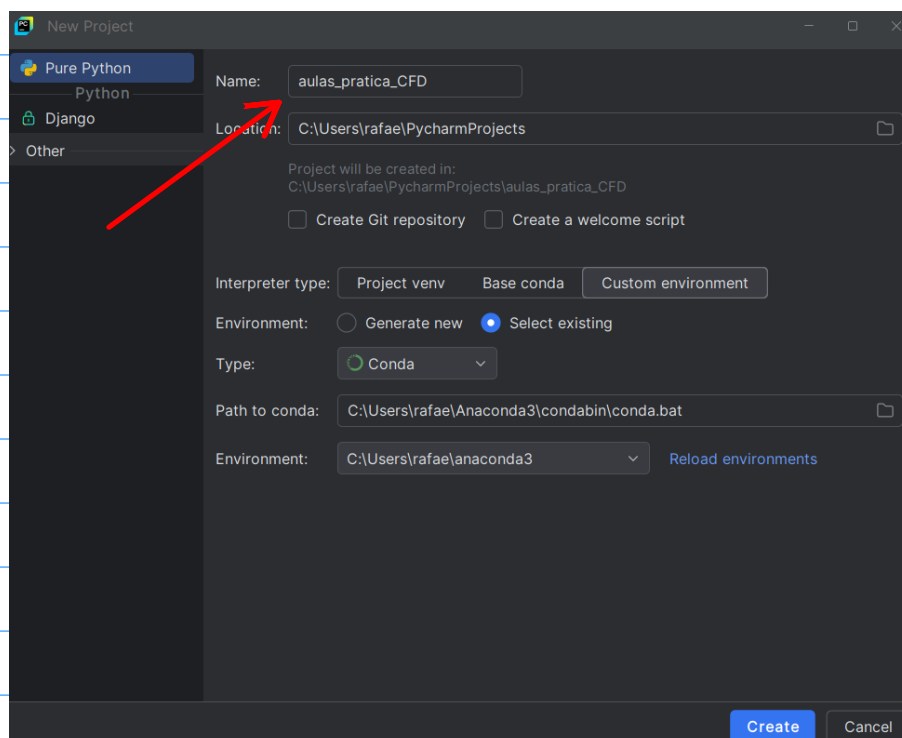


* Você podem optar pela versão teste também. Nesse caso, ao cadastrarem uma conta com e-mail @ufsc.br, vocês tem direito a versão Full. O bom dela é poder rodar Jupyter Notebooks na IDE.

Feito isso, vamos criar nosso primeiro projeto:

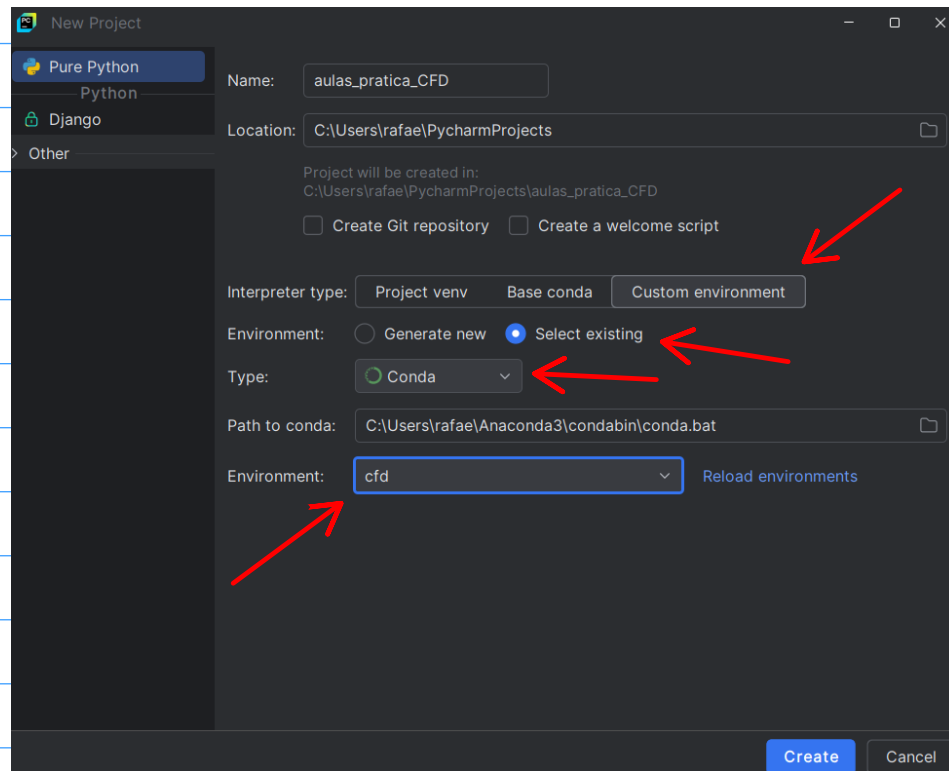


Vamos dar o nome de nosso projeto de "aulas_pratica_CFD":



Agora vamos configurar o ambiente virtual:

1. Utilize a opção "Custom environment", depois selecionar "Select existing", "Type" escolher conda e por fim definir o "Environment" como "cfd".



Se tudo deu certo, o ambiente está pronto, resultando nessa tela abaixo

