

# NCAR Python Intro Tutorial

## Beginner Python for Scientists by Julia Kent

<https://ncar.github.io/python-tutorial/tutorials/beginner.html#reading-a-text-file>

14.10.2020 - CLASS 1

### FIRST PYTHON SCRIPT

\* Criar um diretório para o curso

```
cd iCloud\ Drive\ \ (Arquivo\)\  
cd Documents/Py/  
mkdir NCAR  
cd NCAR/
```

\* Criar um ambiente do conda para o curso

Criar um env para cada projeto — diferentes versos podem impedir scripts velhos  
“instala vários Pythons, um p/ cada ambiente, de maneira que cada projeto tem suas versões

```
conda create --name NCAR_tutorial python
```

\* Iniciar o ambiente

```
conda activate NCAR_tutorial  
>> Initialized empty Git repository in /Users/snati/iCloud Drive  
(Arquivo)/Documents/Py/NCAR/.git/
```

\* Cria um git vazio

\* A Git repository tracks changes made to files within your project. It looks like a **.git/** folder inside that project.

```
git init .
```

\* Adicionar o dado que iremos usar à pasta

\* Download sample data from the CU Boulder weather station:

```
(NCAR_tutorial) snati:data snati$ curl -k0 https://sundowner.colorado.edu/weather/atoc8/wxobs20170821.txt  
[ % Total      % Received % Xferd  Average Speed   Time    Time       Time  Current  
   Dload  Upload   Total   Spent    Left   Speed  
100 116k 100 116k    0     0 43827      0  0:00:02  0:00:02 --:--:-- 43811
```

Path completo:

```
cd iCloud\ Drive\ \ (Arquivo\)/Documents/Py/NCAR/
```

\* Download via comando curl

```
curl -k0 https://sundowner.colorado.edu/weather/atoc8/  
wxobs20170821.txt  
>>
```

\* Criamos um diretório git que ainda está vazio. Verificar:

```
git status
```

```
>>On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
./
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

\* Adiciona o dado ao git

```
git add wxobs20170821.txt
```

```
>>On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   wxobs20170821.txt
```

```
(NCAR_tutorial) snati:data snati$ git commit -m "Adding sample data file"
```

```
*** Please tell me who you are.
```

```
Run
```

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

```
to set your account's default identity.
```

```
Omit --global to set the identity only in this repository.
```

```
fatal: unable to auto-detect email address (got 'snati@snati.(none)')
```

\* Git pede identificação do usuário:

```
git config --global user.email "natalia3.silva@usp.br"
```

\* Criar um script python

```
touch firstscript.py
```

\* Diz que ainda não está comprometido, então:

```
git commit -m "Adding sample data file"
```

```
>>[master (root-commit) 4c169dd] Adding sample data file
```

```
1 file changed, 578 insertions(+)
```

```
create mode 100644 data/wxobs20170821.txt
```

\* Após a identificação, conferimos:

```
git status
```

```
>>On branch master
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
firstscript.py
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

```
(NCAR_tutorial) snati:data snati$ git log
```

```
commit 4c169ddc527c415e34943cb17d7a8b5d3ee2e8d9 (HEAD -> master)
```

```
Author: Natalia Silva <natalia3.silva@usp.br>
```

```
Date: Wed Oct 14 16:27:25 2020 -0300
```

Adding sample data file

\* Abrir o script no seu editor de texto escolhido.

No curso, digita-se:

```
nano firstscript.py
```

Aqui abriremos manualmente no sublime.

\* Iniciando tudo no dia seguinte....

Abrir o diretório no terminal

```
cd iCloud\ Drive\ \ (Arquivo\)/Documents/Py/NCAR/
```

\* Ativar conda e iniciar git

```
conda activate NCAR_tutorial
```

```
git init .
```

```
>>Reinitialized existing Git repository in /Users/snati/iCloud Drive (Arquivo)/Documents/Py/NCAR/.git/
```

\* Verificar se está tudo ok:

```
git status
```

```
>>On branch master
```

```
Changes not staged for commit:
```

```
(use "git add/rm <file>..." to update what will be committed)
```

```
(use "git restore <file>..." to discard changes in working directory)
```

```
deleted: data/wxobs20170821.txt
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
.DS_Store
```

```
firstscript.py
```

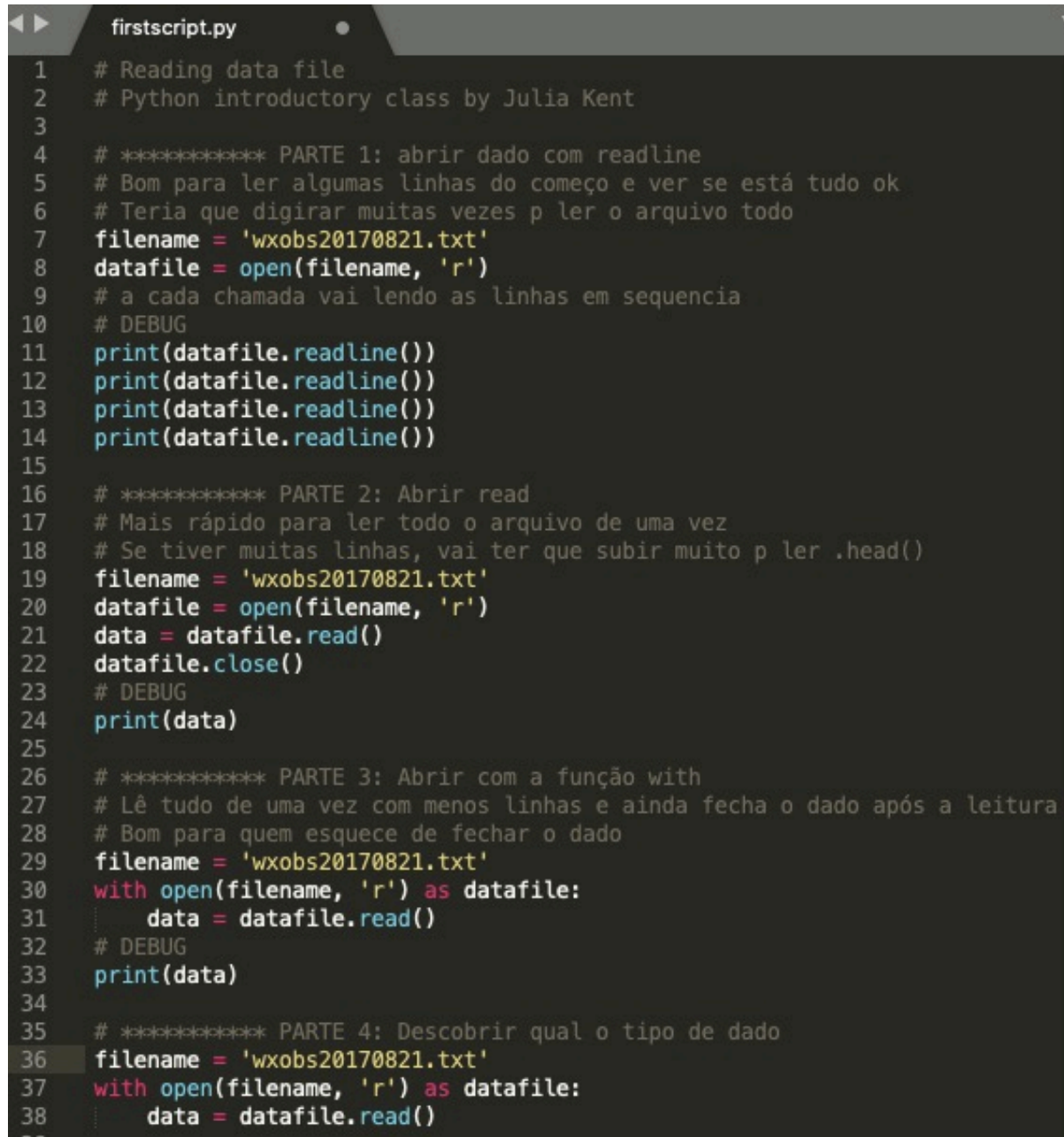
```
wxobs20170821.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

\* Isso corre pois passei os dados e o script da pasta /data para a anterior /NCAR. Portanto, devo adicionar novamente ao git, devido a mudança de diretório

```
git add wxobs20170821.txt
git add firstscript.py
```

- \* Começar a editar o script — aberto no sublime!
- \* Aprendemos a ler um arquivo .txt de 3 maneiras: ler linha por linha `data.readline()`; ler todo o arquivo de uma vez `data.read()`, sendo que ambos exigem que seja fechado depois para evitar de ser corrompido `data.close()`. A terceira forma, usando a função `with`, não precisa que o arquivo seja fechado: `with open(filename, 'r') as datafile: data = datafile.read()`
- \* Também aprendemos a verificar o tipo de arquivo: `type(data)`



```
firstscript.py
1  # Reading data file
2  # Python introductory class by Julia Kent
3
4  # ***** PARTE 1: abrir dado com readline
5  # Bom para ler algumas linhas do começo e ver se está tudo ok
6  # Teria que digitar muitas vezes p ler o arquivo todo
7  filename = 'wxobs20170821.txt'
8  datafile = open(filename, 'r')
9  # a cada chamada vai lendo as linhas em sequencia
10 # DEBUG
11 print(datafile.readline())
12 print(datafile.readline())
13 print(datafile.readline())
14 print(datafile.readline())
15
16 # ***** PARTE 2: Abrir read
17 # Mais rápido para ler todo o arquivo de uma vez
18 # Se tiver muitas linhas, vai ter que subir muito p ler .head()
19 filename = 'wxobs20170821.txt'
20 datafile = open(filename, 'r')
21 data = datafile.read()
22 datafile.close()
23 # DEBUG
24 print(data)
25
26 # ***** PARTE 3: Abrir com a função with
27 # Lê tudo de uma vez com menos linhas e ainda fecha o dado após a leitura
28 # Bom para quem esquece de fechar o dado
29 filename = 'wxobs20170821.txt'
30 with open(filename, 'r') as datafile:
31     data = datafile.read()
32 # DEBUG
33 print(data)
34
35 # ***** PARTE 4: Descobrir qual o tipo de dado
36 filename = 'wxobs20170821.txt'
37 with open(filename, 'r') as datafile:
38     data = datafile.read()
39
```

- \* A cada mudança no script, rodamos no terminal para testar:  
`python firstscript.py`

\* Ao final da aula, adicionamos e organizamos a pasta do Git:

```
git status
```

```
>> On branch master
```

**Changes to be committed:**

(use "git restore --staged <file>..." to unstage)

new file: firstscript.py

new file: wxobs20170821.txt

Changes not staged for commit:

(use "git add/rm <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

deleted: data/wxobs20170821.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

.DS\_Store

\* Note que a resposta incluía a sessão “changes to be committed”. Ou seja, precisamos de fato adicionar as mudanças à pasta Git. Para isso:

```
git commit -m "Adding script file"
```

O que vem entre “ ” descreve qual foi a mudança feita ao commit. Dessa vez adicionamos o script ao diretório.

```
>>[master 3f9caba] Adding script file
2 files changed, 614 insertions(+)
create mode 100644 firstscript.py
create mode 100644 wxobs20170821.txt
```

\* Verificando:

```
git status
```

```
>> On branch master
```

Changes not staged for commit:

(use "git add/rm <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

deleted: data/wxobs20170821.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

.DS\_Store

**no changes added to commit** (use "git add" and/or "git commit -a")

\* E então vemos o que tem no diretório Git

```
git log
```

```
>> commit 3f9caba301d13b95d722f94d3ceb8c7f6d270040 (HEAD -> master)
```

```
Author: Natalia Silva <natalia3.silva@usp.br>
```

```
Date: Thu Oct 15 11:58:41 2020 -0300
```

```
Adding script file
```

```
commit 4c169ddc527c415e34943cb17d7a8b5d3ee2e8d9
```

Author: Natalia Silva <natalia3.silva@usp.br>

Date: Wed Oct 14 16:27:25 2020 -0300

Adding sample data file

\* Ou de maneira simplificada:

```
git log --oneline
```

```
>>3f9caba (HEAD -> master) Adding script file
```

```
4c169dd Adding sample data file
```

**28.10.2020 - CLASS 2**

## **FIRST PYTHON SCRIPT — PART 2**

\* Voltar para o diretório do curso:

```
cd iCloud\ Drive\ \ (Arquivo\)/Documents/Py/NCAR/
```

\* E ativar o ambiente conda — para o caso de não lembrar qual o nome do ambiente, conferimos:

```
conda env list
```

```
>> # conda environments:
```

```
#
```

```
base * /Users/snati/anaconda3
```

```
MESTRADO /Users/snati/anaconda3/envs/MESTRADO
```

```
NCAR_tutorial /Users/snati/anaconda3/envs/NCAR_tutorial
```

\* Então:

```
conda activate NCAR_tutorial
```

```
>> (NCAR_tutorial) snati:NCAR snati$
```

\* Abrir o script no editor de texto escolhido (aqui, Sublime Text):

```
# ***** PARTE 5: Initialize data variable with dictionary
data = []
filename = 'wxobs20170821.txt'
with open(filename, 'r') as datafile:
    #
    for _ in range(3): # same as loop in a list [0, 1, 2]
        # print(_). # Just to see what is the _, which is the same as i or j...
        datafile.readline() # read one line three times
    #
    for line in datafile: # Read and parse all the file but the header
        # print(line)
        datum = line.split() # Faz uma lista de "listas"
        data.append(datum) # Adiciona uma lista ao final do dicionário

# # DEBUG
# for datum in data:
#     print(datum)
```

- \* Aprendendo a usar listas e loop para ler cada linha do dado. No primeiro *for*, lemos apenas as 3 primeiras linhas (*range(3)*). No segundo *loop for*, lemos todo o arquivo, e adicionamos cada linha do arquivo ao final da lista/dicionário *data*.
- \* Aprendendo a selecionar os elementos da lista através de indexação. Na figure,

```
# # DEBUG
print(data[0]) # first index
print(data[-1]) # negative index works for reverse indices in list
print(data[0:10]) # start and stop at these indices

# A better way to look at the output:
for datum in data[0:10]:
    print(datum) # imprime cada linha de uma vez

# If you don't specify your start, it will always initialize in index 0:
for datum in data[:10]:
    print(datum)

# You can also set the step in your range:
for datum in data[0:10:2]:
    print(datum) # imprime a cada 2 linhas

# Print the fifth element of the ninth row in data:
print(data[8][4])

# Since we're dealing with strings, we can select which character we want.
# For instance, the first character of the fifth element in the ninth row:
print(data[8][4][0])

# Print the five first indices of row nine
print(data[0][:5]) # header
print(data[8][:5])

# Print every index in row 8, but with a step of two
print(data[0][::2]) # header
print(data[8][::2])
```

vemos diversos exemplos de como acessar as linhas, e os as colunas (elementos de cada linha).

Lembrar que os índices se dão por: **data [ linha ] [ coluna ] [caractere ]**

- \* E finalmente, salvamos as alterações no Git:

- \* Adiciona o script ao Git

```
git add firstscript.py
```

- \* Commit o script adicionado

```
git commit -m "Criando listas e aprendendo indexação"
```

```
>> [master 1e53f20] Criando listas e aprendendo indexação
1 file changed, 66 insertions(+), 4 deletions(-)
```

- \* Conferir as mudanças feitas ao Git. Duas mudanças da primeira aula e mais a adicionada hoje.

```
git log
```



```
>> commit 1e53f20ab91577087c4c3f1e6ba630d7a2379cb4 (HEAD -> master)
Author: Natalia Silva <natalia3.silva@usp.br>
Date: Thu Oct 29 11:55:46 2020 -0300
```

Criando listas e aprendendo indexação

```
commit 3f9caba301d13b95d722f94d3ceb8c7f6d270040
Author: Natalia Silva <natalia3.silva@usp.br>
Date: Thu Oct 15 11:58:41 2020 -0300
```

Adding script file

```
commit 4c169ddc527c415e34943cb17d7a8b5d3ee2e8d9
Author: Natalia Silva <natalia3.silva@usp.br>
Date: Wed Oct 14 16:27:25 2020 -0300
```

Adding sample data file

- \* Lendo cada coluna do arquivo como uma lista, e adicionando tudo na variável como dicionário.
- \* Cada linha do arquivo corresponde à uma observação e cada coluna, à uma variável.

```
# ***** PARTE 7: É difícil saber qual coluna corresponde à que variável.
# É mais interessante abrir o dado como um dicionário, ao invés de lista
# Ao invés de buscar por índice na row, podemos buscar por variável (tempo, T)
# Ao invés de colchetes, abrimos chaves. Ao invés de lista, dicionário...
# Initialize my data variable
data = {'date': [], 'time': [], 'tempout': []}
filename = "wxobs20170821.txt"
with open(filename, 'r') as datafile:
    #
    for _ in range(3): # Read the first three lines (header)
        datafile.readline() # Se não ler as primeiras linhas vai dar erro --?
    #
    # Read and parse the rest of the file
    for line in datafile:
        split_line = line.split()
        data['date'].append(split_line[0])
        data['time'].append(split_line[1])
        data['tempout'].append(split_line[2])

# DEBUG
print(data['time'])
```

Desta maneira, adicionamos cada variável à sua lista específica, ou seja, atribuímos à coluna a uma variável.

- \* Novamente, atualizamos as alterações no Git:

```
(NCAR_tutorial) snati:NCAR snati$ git add firstscript.py
```

```
(NCAR_tutorial) snati:NCAR snati$ git commit -m "Analisar time-series
com dicionário"
```

```
>> [master 714ffc4] Analisar time-series com dicionário
1 file changed, 22 insertions(+)
```

```
(NCAR_tutorial) snati:NCAR snati$ git log
```



```
>> commit 714ffc4888f5a4db0bed7be23979f1a143b032cc (HEAD -> master)
Author: Natalia Silva <natalia3.silva@usp.br>
Date: Thu Oct 29 12:20:47 2020 -0300
```

### Analisar time-series com dicionário

\* Vamos ler o arquivo criando um dicionário, assim como anteriormente. Mas agora

```
# ***** PARTE 9: E se quisermos adicionar mais variáveis ao dicionário?
# Os índices [0, 1, 2] correspondem à coluna no dado
columns = {'date': 0, 'time': 1, 'tempout': 2}

# Queremos ler a coluna de temperaturas como float
# Data types for each column (only if non-string)
types = {'tempout': float}

# Não vamos mais estabelecer quais as variáveis/listas dentro do dicionário.
# Vamos usar um loop for que fará isso automaticamente
data = {}
for column in columns:
    data[column] = []
# Faz um dicionário com as variáveis como listas vazias.
# Mesmo que o comando da parte 8

# Read and parse the data file
filename = "wxobs20170821.txt"
with open(filename, 'r') as datafile:

    # Read the first three lines (header)
    for _ in range(3):
        datafile.readline()

    # Read and parse the rest of the file
    for line in datafile:
        split_line = line.split()
        for column in columns:
            i = columns[column]
            t = types.get(column, str)
            # data['float'] -> str
            # tenta achar os dados do tipo float, mas se não achar, lê como str
            value = t(split_line[i]) # como fizemos float(split_line[coluna])
            data[column].append(value) # append automático com o dado no
            # formato correto para cada variável/lista

# DEBUG
print(data['date'])
print(data['tempout'])
```

será de forma automatizada, com um loop para cada coluna/variável e também com uma função que determina se a lista será str ou float.

\* Novamente, atualizamos as alterações no Git:

```
(NCAR_tutorial) snati:NCAR snati$ git add firstscript.py
```

```
(NCAR_tutorial) snati:NCAR snati$ git commit -m "Criar time-series
dicionári automatizado"
```

```
>> [master 714ffc4] Criar time-series dicionári automatizado 1 file
changed, 22 insertions(+)
```

```
(NCAR_tutorial) snati:NCAR snati$ git log -- oneline
```

```
>> 3919d82 (HEAD -> master) Criar time-series dicionário automatizado
714ffc4 Analisar time-series com dicionário
1e53f20 Criando listas e aprendendo indexação
3f9caba Adding script file
4c169dd Adding sample data file
```

### 11.11.2020 - CLASS 3

### FIRST PYTHON SCRIPT (FUNCTIONS) — PART 3

- \* Vamos construir uma função que devolve índice para ventos. Precisaremos dos dados de velocidade e direção do vento que constam no file.txt.
- \* Lendo o cabeçalho do file, procuramos qual o índice dos dados que queremos:

```
186
187 # ***** PARTE 10: Vamos construir uma função que lê
188 # velocidade e direção do vento, e retorna um índice
189 columns = {'date': 0, 'time': 1, 'tempout': 2}
190
191 # Queremos ler a coluna de temperaturas como float
192 # Data types for each column (only if non-string)
193 types = {'tempout': float}
194
195 # Não vamos mais estabelecer quais as variáveis/listas dentro do dicionário.
196 # Vamos usar um loop for que fará isso automaticamente
197 data = {}
198 for column in columns:
199     data[column] = []
200 # Faz um dicionário com as variáveis como listas vazias.
201 # Mesmo que o comando da parte 8
202
203 # Read and parse the data file
204 filename = "wxobs20170821.txt"
205 with open(filename, 'r') as datafile:
206
207     # Read the first three lines (header)
208     for _ in range(3):
209         headerline = datafile.readline()
210         print(headerline)
211
```

- \* Descobrimos que os dados de Velocidade do vento (WS) e direção (D), que serão usados na fórmula, correspondem aos índices 7 e 8.

- \* Então vamos ler as colunas 7 e 8, assim como fizemos com temperatura e datas anteriormente.

```
# ***** PARTE 11: Com o print do header no passo 10,  
# descobrimos que os dados de vento correspondem aos índices 7 e 8.  
# Agora vamos lê-los e adicioná-los ao dicionário  
columns = {'date': 0, 'time': 1, 'tempout': 2, 'windspeed': 7}  
# Velocidade do vento na coluna 7  
  
# Data types for each column (only if non-string)  
types = {'tempout': float, 'windspeed': float}  
# Velocidade do vento vem no formato de número  
  
data = {}  
for column in columns:  
    data[column] = []  
  
# Read and parse the data file  
filename = "wxobs20170821.txt"  
with open(filename, 'r') as datafile:  
  
    # Read the first three lines (header)  
    for _ in range(3):  
        datafile.readline()  
  
    # Read and parse the rest of the file  
    for line in datafile:  
        split_line = line.split()  
        for column in columns:  
            i = columns[column]  
            t = types.get(column, str)  
            # data['float'] ----> str  
            # tenta achar os dados do tipo float, mas se não achar, lê como str  
            value = t(split_line[i])  
            data[column].append(value)  
  
# DEBUG  
print(data['windspeed'])
```

- \* Salvamos as alterações no Git

```
(NCAR_tutorial) snati:NCAR snati$ git add firstscript.py
```

```
(NCAR_tutorial) snati:NCAR snati$ git commit -m "Lendo dados de  
velocidade do vento (WS)"
```

```
>> [master ba7da99] Lendo dados de velocidade do vento (WS)  
1 file changed, 79 insertions(+), 18 deletions(-)
```

```
(NCAR_tutorial) snati:NCAR snati$ git log --oneline
```

```
>> ba7da99 (HEAD -> master) Lendo dados de velocidade do vento (WS)  
d506afb Adicionar a apostila do curso ao Git  
3919d82 Analisar time-series com dicionário  
714ffc4 Analisar time-series com dicionário  
1e53f20 Criando listas e aprendendo indexação  
3f9caba Adding script file
```

### 4c169dd Adding sample data file

- \* Salvamos as alterações no Git
- \* Agora escrevendo propriamente a função:

```
# Definimos uma função, a qual o nome explicita o que faz e entre
# () quais argumentos ela irá usar para os cálculos
def compute_windchill(t, v):

    a, b, c, d = 35.74, 0.6215, 35.75, 0.4275

    v16 = v ** 0.16

    wci = a + (b * t) - (c * v16) + (d * t * v16)

    # Nenhuma outra variável definida dentro da função fica disponível para
    # o resto do código (ex. se chamar a, b, c, d e v16 fora da função
    # não vai existir). Apenas a variável de retorno pode ser usada depois
    return wci

# # DEBUG: Running the function to comput wci
# # Vamos rodar nossa função usando zip.
# # Zip pega os elementos de interesse da matriz e ignora os outros.
# Aqui, 5 está fora da matriz, então é ignorado
# for i, j in zip([1, 2], [3, 4, 5]):
#     print(i, j)
windchill = []
for temp, windspeed in zip(data['tempout'], data['windspeed']):
    windchill.append(compute_windchill(temp, windspeed))
```

- \* Dentre os dados inseridos/definidos dentro da função, apenas o valor retornado pode ser chamado!! Se chamar a, b, c, v16 não vai funcionar
  - \* Finalmente, vamos comparar os dados de Wind Chill calculados com os valores que já vêm no dado — no tutorial, eles explicam que a grande diferença é devido à temperatura: esse cálculo de WC não é adequado para as temperaturas do dado.. mas vamos usá-la pois é só um exemplo
- Aqui também printamos os resultados usando f'{} que possibilita escolher quantas casas decimais vão aparecer e o formato do output
- \* Nosso script para cálculo de Wind Chill está pronto



```

# ***** PARTE 13: Comparar o valor de wind chill calculado com o que
# já veio no dado (se olharmos no header, veremos que esse dado vem na
# coluna 13 — índice 12)

columns = {'date': 0, 'time': 1, 'tempout': 2, 'windspeed': 7, 'wchill': 12}

# Data types for each column (only if non-string)
types = {'tempout': float, 'windspeed': float, 'wchill': float}

data = {}
for column in columns:
    data[column] = []

# Read and parse the data file
filename = "wxobs20170821.txt"
with open(filename, 'r') as datafile:

    # Read the first three lines (header)
    for _ in range(3):
        datafile.readline()

    # Read and parse the rest of the file
    for line in datafile:
        split_line = line.split()
        for column in columns:
            i = columns[column]
            t = types.get(column, str)
            # data['float'] —> str
            # tenta achar os dados do tipo float, mas se não achar, lê como str
            value = t(split_line[i])
            data[column].append(value)

# Definimos uma função, a qual o nome explicita o que faz e entre
# () quais argumentos ela irá usar para os cálculos
def compute_windchill(t, v):

    a, b, c, d = 35.74, 0.6215, 35.75, 0.4275

    v16 = v ** 0.16

    wci = a + (b * t) - (c * v16) + (d * t * v16)

    # Nenhuma outra variável definida dentro da função fica disponível para
    # o resto do código (ex. se chamar a, b, c, d e v16 fora da função
    # não vai existir). Apenas a variável de retorno pode ser usada depois
    return wci

# # DEBUG: Running the function to comput wci
# # Vamos rodar nossa função usando zip.
# # Zip pega os elementos de interesse da matriz e ignora os outros.
# Aqui, 5 está fora da matriz, então é ignorado
# for i, j in zip([1, 2], [3, 4, 5]):
#     print(i, j)
windchill = []
for temp, windspeed in zip(data['tempout'], data['windspeed']):
    windchill.append(compute_windchill(temp, windspeed))

# Comparar os valores de WChill calculados e dos dados
# for wc_data, wc_comp in zip(data['wchill'], windchill):
#     print(f'{wc_data: .5f} {wc_comp: .5f} {wc_data - wc_comp: .5f}')

# Reescrevendo os ultimos comandos para melhorar a apresentação
# Vamos incluir prints antes do loop como header para auxiliar na visualização
print(' DATE      TIME      WC_ORIG      WC_COMP      WC_DIFF')
print('-----')
# Colocamos as linhas como marcações para cada output

zip_data = zip(data['date'], data['time'], data['wchill'], windchill)
for date, time, wc_orig, wc_comp in zip_data:
    wc_diff = wc_orig - wc_comp
    print(f'{date} {time: >6} {wc_orig: 9.6f} {wc_comp: 9.6f} {wc_diff: 10.6f}')

# # *****

```

- \* Vamos salvar tudo no Git e passar para o “Google Drive” dos códigos, o GitHub.
- \* Primeiro renomear nosso script

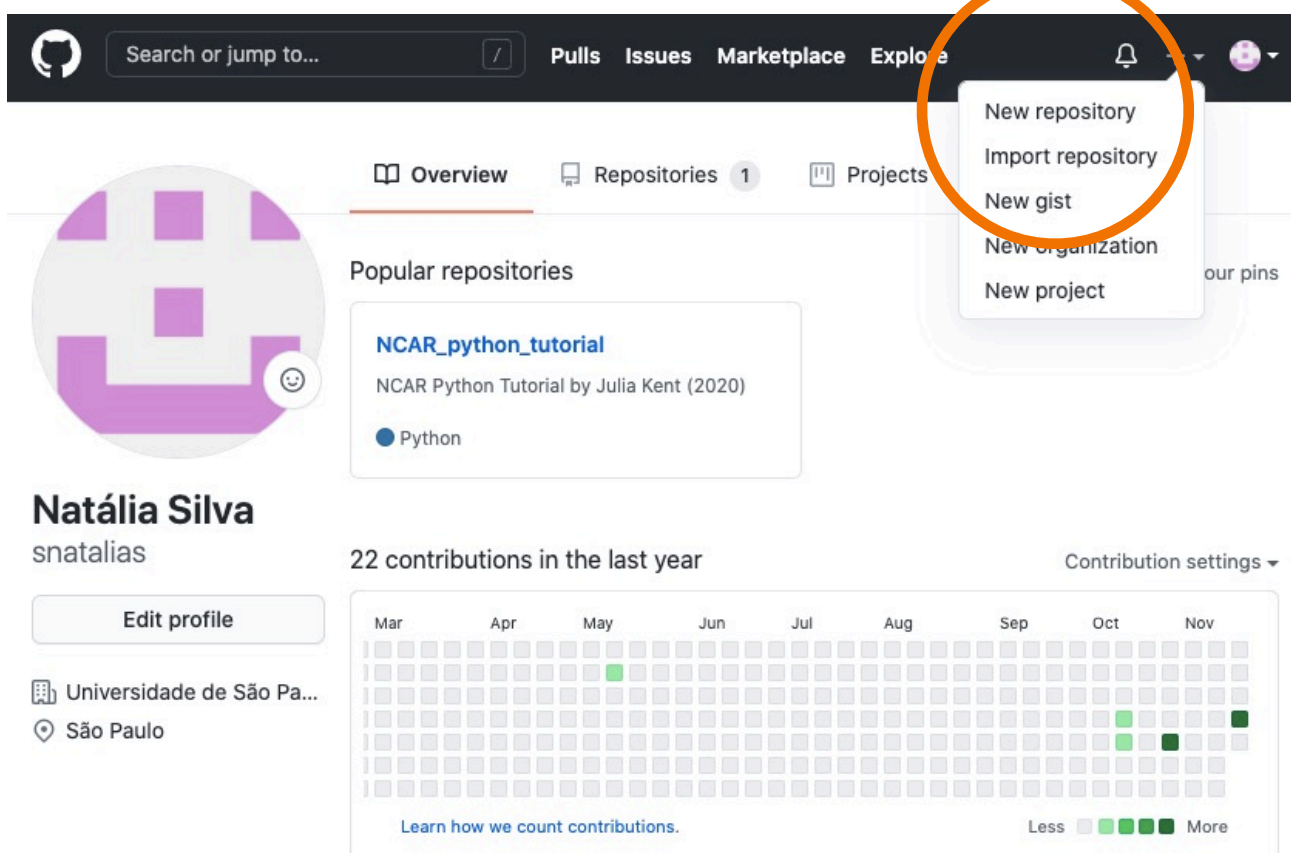
```
(NCAR_tutorial) snati:NCAR snati$ git add firstscript.py
```

```
(NCAR_tutorial) snati:NCAR snati$ git commit -m "Output formatting comparision"
```

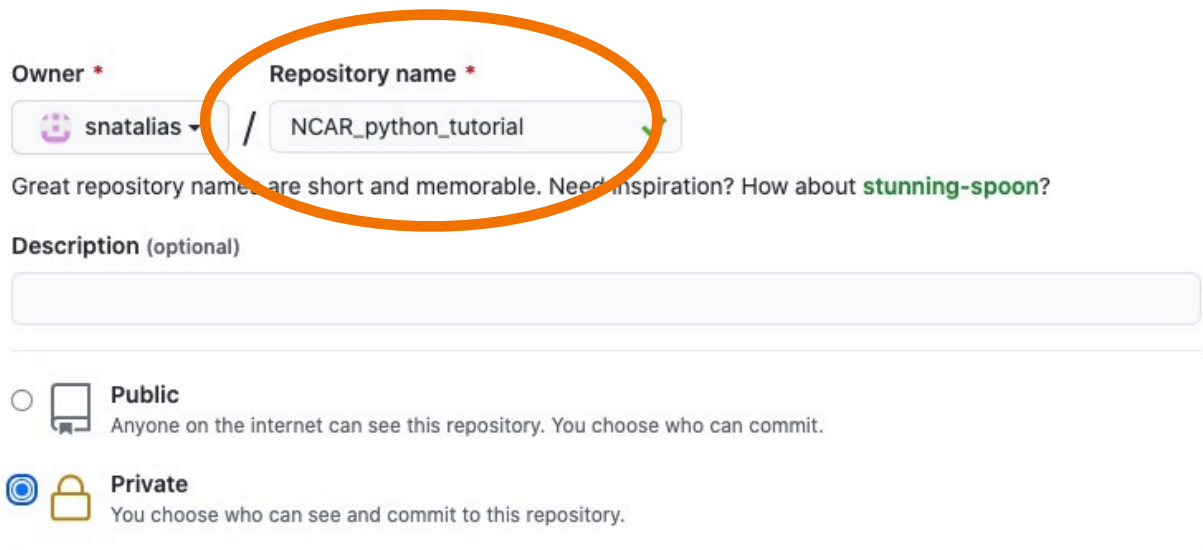
```
(NCAR_tutorial) snati:NCAR snati$ git mv firstscript.py windchill.py
```

```
(NCAR_tutorial) snati:NCAR snati$ git commit -m "Renaming the script"
```

- \* No site do GitHub:

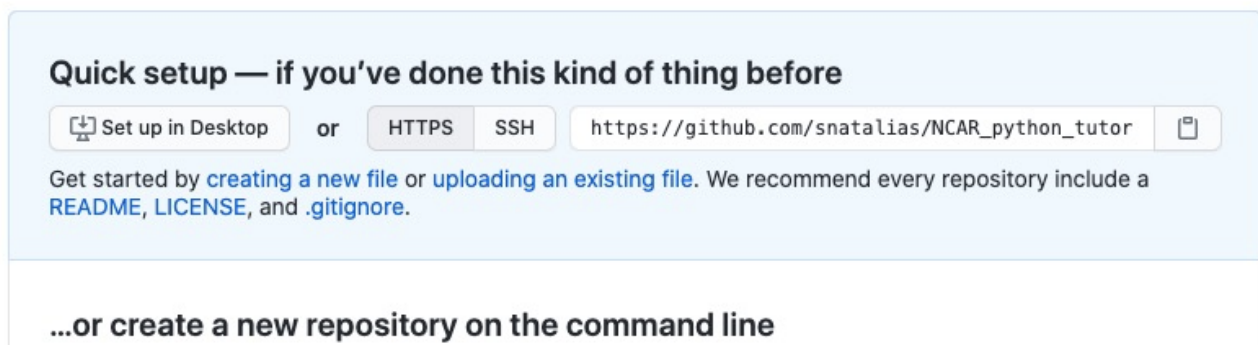


- \* Criar um novo repositório e escolher se é publico/privado -- vou deixar público





- \* Copiar o link do repositório para adicionarmos conteúdo via terminal



- \* Link entre a página do repositório no GitHub e o PC

```
(NCAR_tutorial) snati:NCAR snati$ git remote add origin https://github.com/snatalias/NCAR_python_tutorial.git
```

Criei o repositório, exclui e tentei criar de novo p/ praticar. Deu erro após o comando acima:

```
>> fatal: remote origin already exists.
```

P/ resolver:

```
(NCAR_tutorial) snati:NCAR snati$ git remote set-url origin https://github.com/snatalias/NCAR_python_tutorial.git
```

- \* Conferir

```
(NCAR_tutorial) snati:NCAR snati$ git remote -v
```

```
>> origin    https://github.com/snatalias/python_tutorial.git (fetch)
origin      https://github.com/snatalias/python_tutorial.git (push)
```

- \* Pelo push, vamos passar os files para o Git. Entendendo o comando push:  
git push <nome do remoto> <nome do branch local>:refs/heads/<nome do branch remoto>

- \* Qual o nome do local? Vamos passar o main para o Git:

```
(NCAR_tutorial) snati:NCAR snati$ git show-ref
```

```
>> 331c6dda359ca5cdfb77abe7f60cd121896ae58a refs/heads/main
331c6dda359ca5cdfb77abe7f60cd121896ae58a refs/remotes/origin/master
```

```
(NCAR_tutorial) snati:NCAR snati$ git push origin main
```

```
>> Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Delta compression using up to 4 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (30/30), 795.01 KiB | 2.75 MiB/s, done.
Total 30 (delta 11), reused 0 (delta 0)
remote: Resolving deltas: 100% (11/11), done.
To https://github.com/snatalias/NCAR_python_tutorial.git
 * [new branch]      main -> main
```

