Python: do básico à autonomia intermediária `aula 0`

Bernardo Alves Furtado

1 de fevereiro de 2022

Menu do dia: boas-vindas, apresentações, curso, instalaçoes, interfaces, HelloWorld!

Introdução
- Apresentação

Motivação
- Simulação, Modelagem

Hands-on
- Instalações
- Exercícios iniciais

Bernardo Alves Furtado

- Pesquisador produtividade CNPq 2014–
- Ph.D Utrecht University, 2009
- Co-tutorship UFMG Dr. Economia Regional
- Arquiteto, urbanista, mestre em Geografia/GIS
- Professor (1988/2003/2006–)
- Ipea: 2009–2013 Políticas urbanas
- Ipea: 2014– Sistemas complexos e Modelagem Baseada em Agentes (ABM)

ipea Institute for Applied Economic Research

Contatos e links

- furtadobb@gmail.com
- researchgate.net/profile/Bernardo-Furtado-2
- GitHub/BAFurtado/PythonClassIpea
- sites.google.com/view/bernardo-alves-furtado/home

Apresenção alunos

Objetivos

1. Autonomia, flexibilidade
2. Alcançar nível intermediário (leitura, compreensão, programas simples, execução, bibliotecas)
3. Ou seja: compreensão e utilização de Funções, Classes e Módulos. Persistência

Python

- Instalação, operacionalização
- **Funções**
- Estruturas: listas, dicionários, files
- Condicionantes e operadores
- *Loops*. Exercícios
- Bibliotecas. **DataFrames. Manipulação dados.**
- Persistência. Saídas e leituras
- Classes. OOP
- **Plots**
- Exercícios. Exemplos. Projeto

Dos Memoriais!

- 6: Dados
- 2: Webcrawler/Web
- Pesquisas Domiciliares
- Linguagem Natural
- 5: Rotinas
- Consistência
- Visualizações
- 3:Manipulação dados
- Indicadores

Dinâmica das aulas

- As aulas serão **síncronas**, ministradas por via on-line, com **câmeras ligadas**, às terças-feiras, de 18:30 às 22:30
- As aulas serão **hands on!**
- A informação é simples. A operacionalização, nem tanto: `pycheckio`
  - Como se, sabendo as palavras, escrever ou interpretar um texto, não é tão imediato... :—

Exercícios e Avaliação

- Exercícios – 70%, pelo menos: https://py.checkio.org/class/pythonipea2022/
  - First exercise: https://py.checkio.org/en/mission/multiply-intro/
- Trabalho final – de acordo com o memorial.
  - Apresentação proposta 3 minutos, próxima aula
  - Algo concreto. Para o seu trabalho, ou proposta de texto, relatório, ou uma análise.
  - Completo

Python: referências básicas

- GitHub.com/BAFurtado/PythonClassIpea
- Think Python [3]
- greenteapress.com/wp/think-python-2e/
- Think Complexity [2]
- greenteapress.com/wp/think-complexity-2e/

+ Referências

- Wes McKinney, Python for Data Analysis 3rd edition. Beijing: O'Reilly Media, 2022.
- Mark Lutz, Programming Python O'Reilly Media, Inc., 2010.
- Mark Lutz, Learning Python O'Reilly Media, Inc., 2013.
- Vários livros em Português. Vários autores. Kindle Unlimited
- Código estruturado e comentado em português: github.com/fernandofeltrin/PYTHON

Vantagens python

A ferramenta depende do propósito [4]
Se a necessidade é uma regressão, por exemplo: `R, stata, SAS, python, MATLAB` ...
`python`

1. Alto nível $-- >$ fácil de compreender, ler, manipular.
2. Versátil, generalista: ML, web, data, SQL, network.
3. **Full programming language**. Equivalente a Java (ou `C++`), porém, simples.
4. Free. Libraries, libraries, libraries ... and support

Illustração: python x R

```r
# INFORMATION ENTROPY
# THE UNCERTAINTY CONTAINED IN A PROBABILITY DISTRIBUTION IS THE AVERAGE LOG-PROBABILITY OF AN EVENT
# MENOS SUOMATÓRIA DE PI * LOG(PI)
p <- c( 0.3 , 0.7 )
-sum( p*log(p) )

# R Code 7.13
```

rethinking_book7.R

formation_entropy.py

```python
from math import log
from typing import List


def information_entropy_calculus(prob: List) -> float:
    """
    :param prob: probabilities of an event
    :return: information theory -- the uncertainty contained in a probability distribution
    """
    ie = -sum(i * log(i) if i > 0 else 0 for i in prob)
    print(f'Information Entropy is {ie:.04f}')
    return ie


def kl_divergence(prob1, prob2):
    kl = sum(prob1[i] * log(prob1[i]/prob2[i])
             if (prob1[i] > 0) and (prob1[i] > 0) else 0
             for i in range(len(prob1)))
    print(f'Information Entropy is {kl:.04f}')
    return kl


if __name__ == '__main__':
    p = [.3, .7]
    information_entropy_calculus(p)
```

Figure 1: https://github.com/BAFurtado/causal_salad_2021

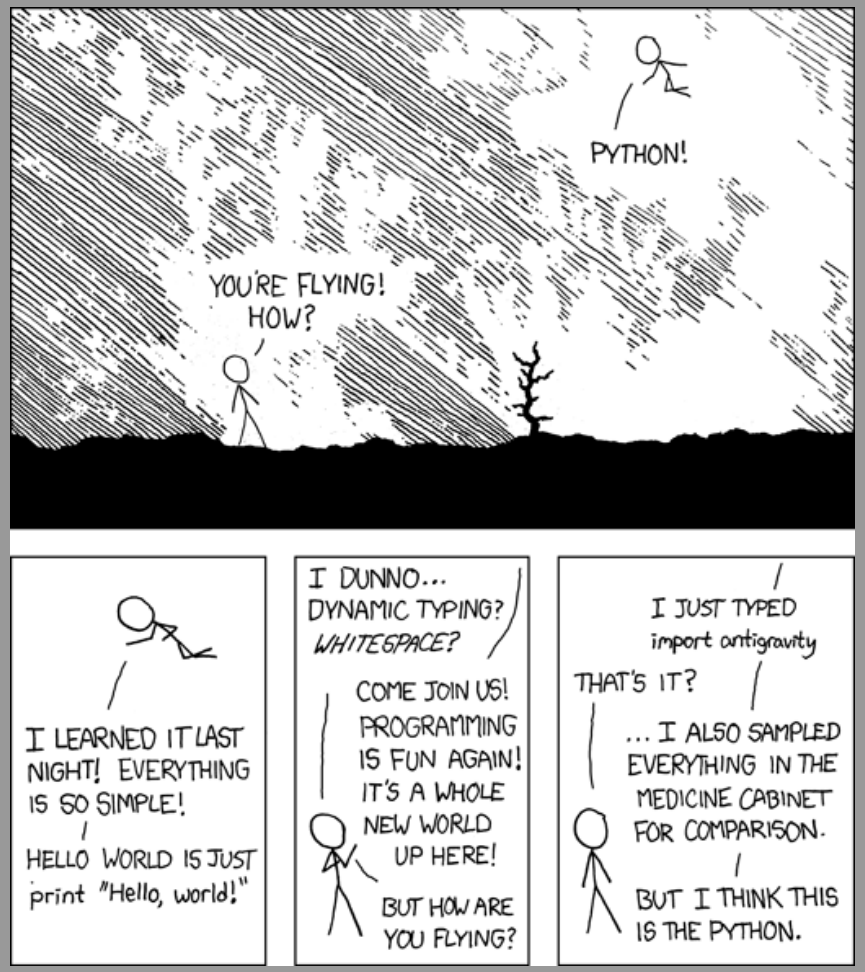
import this I

The Zen of Python, by Tim Peters
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.

import this II

There should be one – and preferably only one – obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea – let's do more of those!

python


YOU'RE FLYING! HOW?
PYTHON!
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

What I actually (think) I do:

Computational models can help us translate observations into an anticipation of future events, act as a testbed for ideas, extract value from data and ask questions about behaviours. The answers are then used to understand, design, manage and predict the workings of complex systems and processes, from public policy to autonomous systems. Models have spread far beyond the domains of engineering and science and are used widely in diverse areas from finance and economics, to business management public policy and urban planning. Increasing computing power and greater availability of data have enabled the development of new kinds of computational model that represent more of the details of the target systems. These allow us to do virtual what if? experiments—even changing the rules of how this detail operates—before we try things out for real. [1]

Instability in the Stable Marriage Problem

Problema original [5]
https://www.hindawi.com/journals/complexity/2018/7409397/

- Método

Método I

2. Methods

We start with the classical scenario with N male and M females to match pairwise. Here, we assume that everyone knows all people from the opposite gender and that there is a wish list for each person which represents the ranking of all persons from the other gender to her/his preference. Following previous research models [11, 13, 17], a reasonable and simple assumption is that all wish lists are randomly established and irrelevant. We define an energy function for each person, which is equal to the ranking of their eventual partner in their wish list. The lower energy one has, the happier the person is. When $N = M$, it is the conventional SMP. Here, we extend the SMP to groups with different sizes. When $N \neq M$, obviously, there will be some people who will remain single. For these persons, their energy is defined as one worse than the bottom of the wish list; that is to say, the energy is $M + 1$ for single men and $N + 1$ for women.

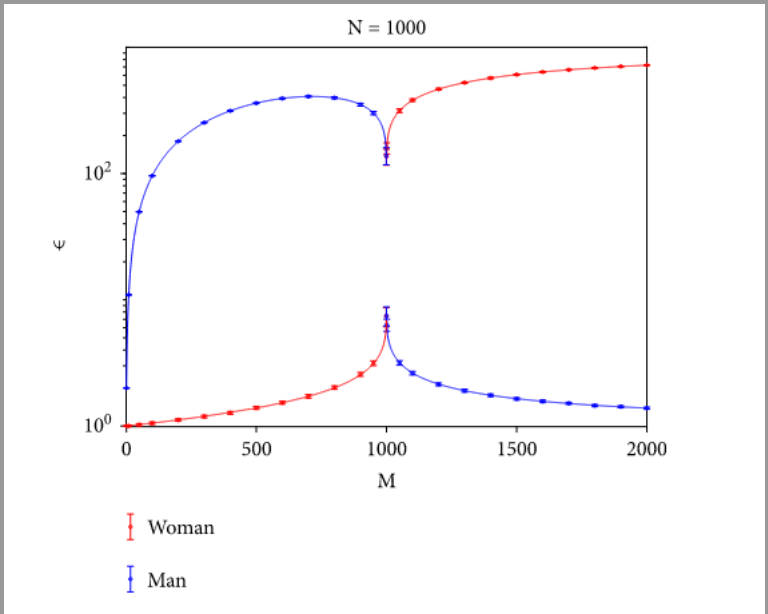
Método II

The G-S algorithm runs as follows: unengaged men will 6
continue to send proposals to women, and women keep the 7
one she prefers between the suitor and her provisional part- 8a
ner. The process stops when no man issues proposal again,
either all men are engaged or the unengaged men are rejected 8b
by everyone. For $N \leq M$, this means that all men are engaged.
For the case of $N > M$, M men are engaged and the remaining
$N - M$ men are still single.

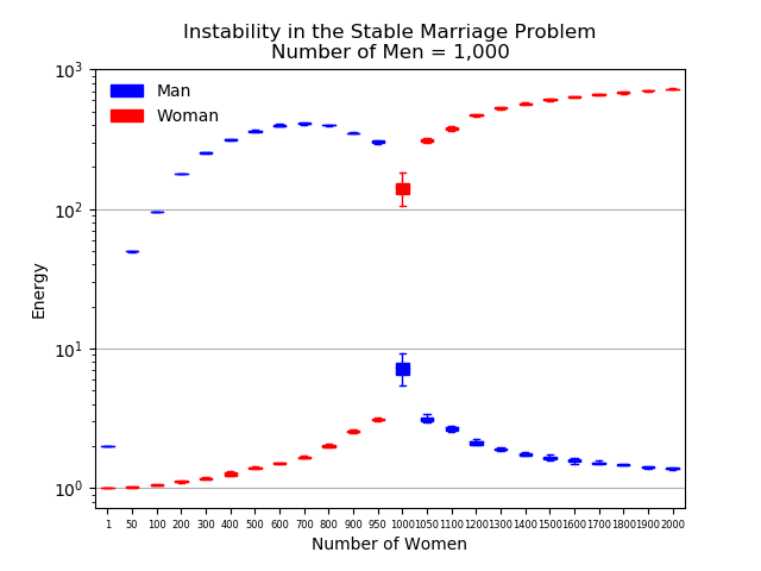
Replicação com variação

- https://arxiv.org/abs/1902.09226
- https://github.com/BAFurtado/HISMP

Método III



(a) Original



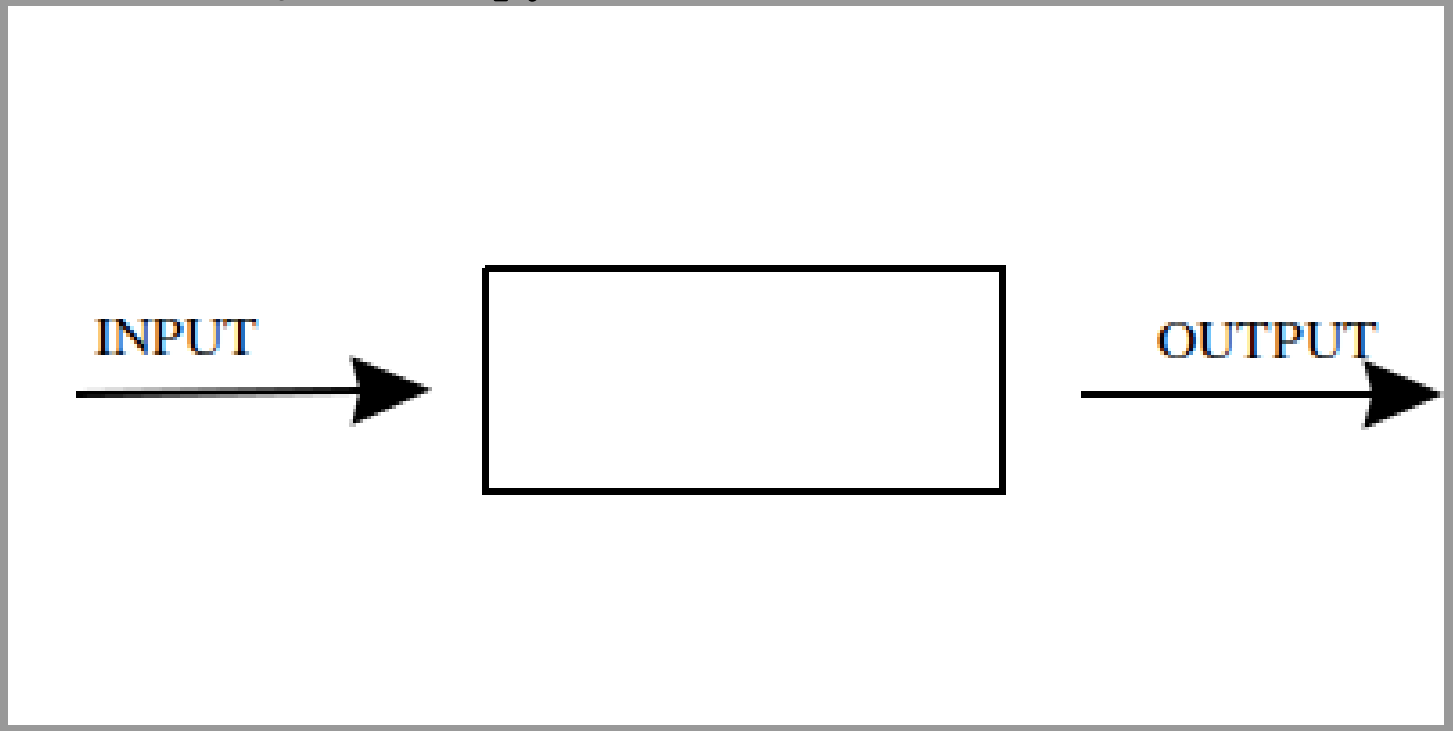
(b) Replication

PyCharm Community e Anaconda

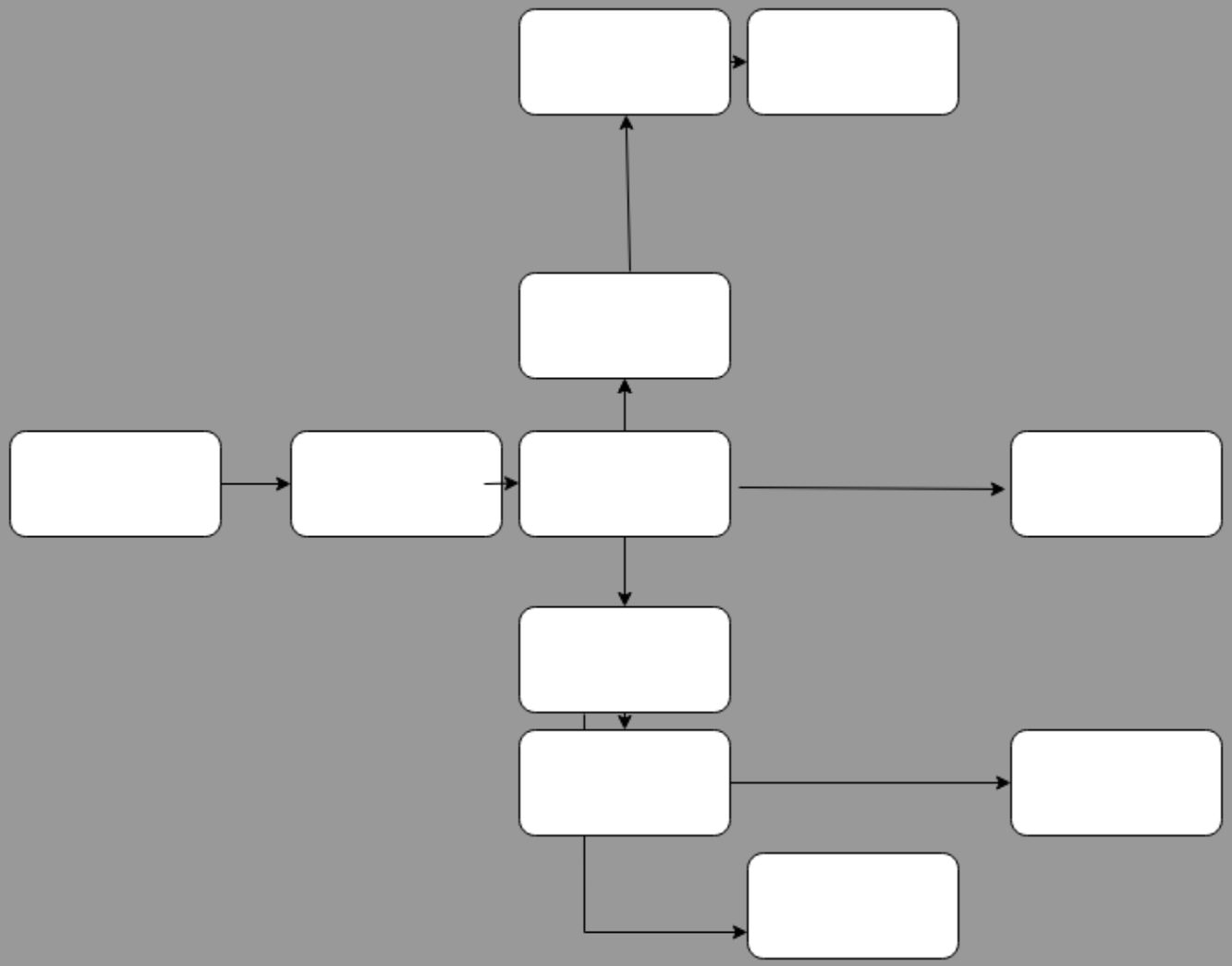
- **PyCharm**: ide, interface, ambiente, **R**
  https://www.jetbrains.com/pycharm/download/
- **Conda**: `python` e suas bibliotecas, libraries
  https://www.anaconda.com/download
- Miniconda
  https://docs.conda.io/en/latest/miniconda.html

Program, Script, Software, App

Tudo é objeto em `python`



Objetos



Exercícios iniciais

- Console
  - $ python
  - >>> print('Hello world')
- `Terminal`
  - $ python hello.py
  - Hello world
- # Let's do it!

python interpreter básico e linha de comando TXT

- **Windows**:
  - `cmd`
  - "Adquirir" APP
  - Abrir (console do python) `>>>`
  - `print('Hello, world!')`
- **Linux**:
  - Todo linux já tem python instalado
  - Abrir `Terminal`

Running python file: hello.py

- Primeiro script: in class. Now.
- Windows: Novo documento texto.
- **Hint:** já vem com o nome *.txt
- `python Documents\hello.txt`

Jupyter

- Jupyter Notebook: Julia, Python, R
- Também possível. No console: ipython
- Segundo script: para próxima aula
- colab.research.google.com

PyCharm Interface

- Arquivos
- Console
- Terminal
- **Debug**
- **Git**
- Python Interpreter/Environment
- Visualizações
- Data
- *.md, R, *.py, Requirements, SQL, databases
- search, replace, PEP checks
- **Conhecimento parâmetros, funções, módulos**

Boas práticas I

DRY: Don't Repeat Yourself

1. Encapsulation

   Wrapping a piece of code up in a function is called encapsulation. One of the benefits of encapsulation is that it attaches a name to the code, which serves as a kind of documentation. Another advantage is that if you re-use the code, it is more concise to call a function twice than to copy and paste the body!

   www.greenteapress.com/thinkpython/html/thinkpython005.html

Boas práticas II

2 Generalization

Adding a parameter to a function is called generalization because it makes the function more general

www.greenteapress.com/thinkpython/html/thinkpython005.html

Noções: int, str, float

No console, digite:

- type(5)
- type('5')
- int('5')
- print(5 + 5)
- print('5' + '5')

input, print–f-string, variable assignment

Digite

- x = int(input('Entre um número:')
- print(f'o número é: {x}')

Tente: `soma.py`

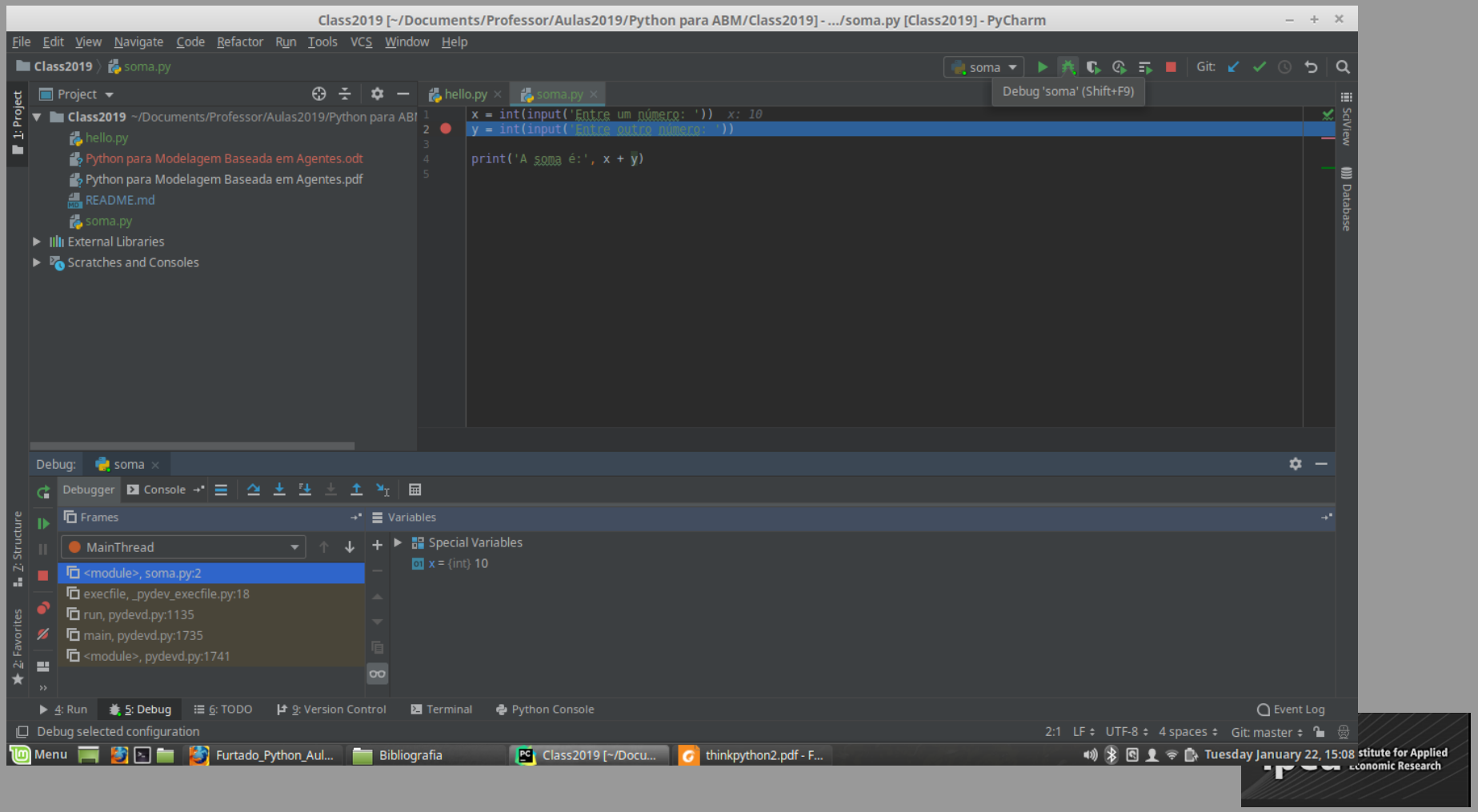
Alguns exercícios

- Descubra esses operadores no console:
- +, -, *, /, **
- Note: order of precedence – PEMDAS
  - Parentheses, Exponentiation, Multiplication, Division, Adition, Subtraction. Da esquerda para a direita
- Quanto é: $(25 * (2 + 23)/54)^2$
- Quanto é: $5 * k$
- minute = 60
- Quanto é: $6 * minute$

Floor division and modulus

- Quantas horas são 200 minutos?
- Como funciona a expressão `if __name__ == '__main__':` no final do script `python`?
- `floor_modulus.py`

Debugging in PyCharm



Python Challenge 0

- http://www.pythonchallenge.com/
- What is the address of the page for Challenge 1?

Exercicios

- Leia o Chapter 1 and 2 do Think Python
- Teste o console, teste o script
- Exercício 1. https://py.checkio.org/en/mission/multiply-intro/

Referências I

[1] Muffy Calder, Claire Craig, Dave Culley, Richard de Cani, Christl A. Donnelly, Rowan Douglas, Bruce Edmonds, Jonathon Gascoigne, Nigel Gilbert, Caroline Hargrove, Derwen Hinds, David C. Lane, Dervilla Mitchell, Giles Pavey, David Robertson, Bridget Rosewell, Spencer Sherwin, Mark Walport, and Alan Wilson. Computational modelling for decision-making: where, why, what, who and how. *Royal Society Open Science*, 5(6):172096, 2018.

[2] Allen B. Downey. *Think Complexity: Complexity Science and Computational Modeling*. O'Reilly Media, Sebastopol, CA, 1 edition edition, March 2012.

[3] Allen B. Downey. *Think Python*. O'Reilly Media, United States of America, 2012.

Referências II

[4] Bruce Edmonds, Christophe Le Page, Mike Bithell, Edmund Chattoe, Volker Grimm, Ruth Meyer, Cristina Montañola Sales, Paul Ormerod, Hilton Root, and Flaminio Squazzoni. Different Modelling Purposes. *Journal of Artificial Societies and Social Simulation, The*, 22:6, June 2019.

[5] Gui-Yuan Shi, Yi-Xiu Kong, Bo-Lun Chen, Guang-Hui Yuan, and Rui-Jie Wu. Instability in Stable Marriage Problem: Matching Unequally Numbered Men and Women. *Complexity*, 2018:5, 2018.