Programming and Data Analytics
Module 1
Lecture 1: Course Introduction

Outline

1. Course introduction
2. Sneak preview of Module 2
3. Let's Kahoot!
4. Overview to programming

Note on the 2-modules structure

2-modules structure: http://bit.ly/PDASSSA21_22

As you know, this course is the first module of a teaching unit of two modules. Intuitively

- M1: Module 1 focuses on programming
- M2: Module 2 focuses on data analysis and machine learning

Students can attend single modules.
M1 gives the necessary background for M2

These slides focus on M1

Note on the 2-modules structure

2-modules structure: http://bit.ly/PDASSSA21_22

As you know, this course is the first module of a teaching unit of two modules. Intuitively

- M1: Module 1 focuses on programming
- M2: Module 2 focuses on data analysis and machine learning

Students can attend single modules.
M1 gives the necessary background for M2

These slides focus on M1

Previous edition, A.Y. 2019/2020

M1 is a re-edition of course **Introduction to Programming in Python** held by us last year for the Allievi Ordinari of SSSA

- If you have attended it, and plan to attend M2, you can skip to it

Note on the 2-modules structure

2-modules structure: http://bit.ly/PDASSSA21_22

As you know, this course is the first module of a teaching unit of two modules. Intuitively

- M1: Module 1 focuses on programming
- M2: Module 2 focuses on data analysis and machine learning

Students can attend single modules.
M1 gives the necessary background for M2

These slides focus on M1

Previous edition, A.Y. 2020/2021

M1 is a re-edition of **Intro to Programming & Data Processing**

- If you have attended it, and plan to attend M2, you can skip to it

As suggested by your colleagues, **this year M2 has twice the hours**

Course Responsible

- Course responsible: Andrea Vandin
  - ★ andrea.vandin@santannapisa.it
  - ★ Tenure-track Assistant Professor in Computer Science at Institute of Economics & EMbeDS @ SSSA, Adjunct Associate Professor at DTU Technical University of Denamrk
  - ★ Former Associate Professor in Computer Science at DTU Technical University of Denmark
    - ▶ Responsible for: *Programming in C++ for non-computer scientists*, ∼250 students
- Co-lecturer: Daniele Licari
  - ★ daniele.licari@santannapisa.it
  - ★ EMbeDS Data Scientist
  - ★ Great academic & industrial experience in Python, data analysis, machine learning, natural language processing, . . .

Related courses

We also teach related courses to two more School of Excellence:

- PhD students of Scuola Superiore Normale, Pisa
- PhD Students in Computer Science of Gran Sasso Science Institute, L'Aquila

Course References & Material

- Webpages of the course:
  - ★ http://bit.ly/PDASSSA21_22
    - Slides and examples from the lectures, further materials and links
    - Weekly coding assignments
- Suggested books:
  - ★ M. Lutz, Learning Python;
  - ★ W. McKinney, Python for Data Analysis.
- Well-done tutorial: https://docs.python.org/3/tutorial/
- Software
  - ★ Python: https://www.python.org/
  - ★ Python editor: JupyterLab https://jupyter.org/
  - ★ Setup your machine: http://bit.ly/PDASSSA21_22

Tentative Course Description - M1

This module will

Introduce students to the fundamental principles of structured programming, with applications to data processing and analysis.

- It starts from basic notions of programming (data types, collections, control structures, functions & modules, OOP),
- Progresses to data processing functionalities (loading, manipulation, and visualization of CSV data).

Tentative Course Description - M1

This module will

Introduce students to the fundamental principles of structured programming, with applications to data processing and analysis.

- It starts from basic notions of programming (data types, collections, control structures, functions & modules, OOP),
- Progresses to data processing functionalities (loading, manipulation, and visualization of CSV data).

A student who has met the objectives of the course will

acquire an understanding of the issues involved in computer programming, to be able to make informed decisions. The student will be able to write simple to medium python programs of various nature, including those for reading, manipulating and visualizing data.

Tentative Learning Objectives

A student who has met the objectives of the course will be able to:

- select and use the correct data types and collections for the problem at hand
- use and describe variables, operations, and control structures (if, loops)
- create and use functions and classes
- use libraries for I/O, data manipulation, and data visualization
- use principles of structured program design and methods
- discuss Python-related issues in a clear and concise way, possibly using on-line platforms

Evaluation

- Regular coding assignments
  - ★ Available at http://bit.ly/PDASSSA21_22
    - Every class comes with a set of related assignments
    - Automatic tests for your code to get hints to fix bugs
    - (Soft) deadlines: before the following class
    - We will try to allocate time at the end of classes
    - You will have to send to us your solutions before the exam.
  - ★ A fundamental learning tool of this course
- Oral Exam
  - ★ We will do an oral examination
    - starting from your solutions to the assignments
    - Another reason for doing your assignments!
  - ★ Date: TBD

Tentative Lecture Plan

#	Date	Time	Topic
1	14/02	15:00-17:00	Course introduction
2	16/02	15:00-18:00	Data types & operations
3	18/02	15:00-18:00	Collections
4	21/02	15:00-18:00	Control and Repetition statements
5	25/02	15:00-18:00	Functions
6	28/02	15:00-18:00	Modules & Exceptions & Object Oriented Programming
7	04/03	15:00-18:00	Advanced libraries for data manipulation/visualization
-	TBD	TBD	Exam

Further info

- No previous experience on computer programming required
- Previous experience in writing small programs is advantageous

Further info

- No previous experience on computer programming required
- Previous experience in writing small programs is advantageous
- You will never learn programming if you don't practice it!
  - ★ Therefore you have to regularly do all the assignments

Ideas for an Effective Course

Live Programming & Assignments

We have blocks of 3 hours.

- First part:
  Intro to week's topics & Live programming
  - ★ No slides
  - ★ We use interactive *notebooks* mixing presentation material and code
    - Please have your laptop ready! http://bit.ly/PDASSSA21_22
    - You find code in advance here
- Second part:
  You consolidate your understanding working on the assignments
  - ★ Begin working on the assignments with our live support if needed
  - ★ Complete them offline before next class. Contact us if needed

Ideas for an Effective Course

Live Programming & Assignments

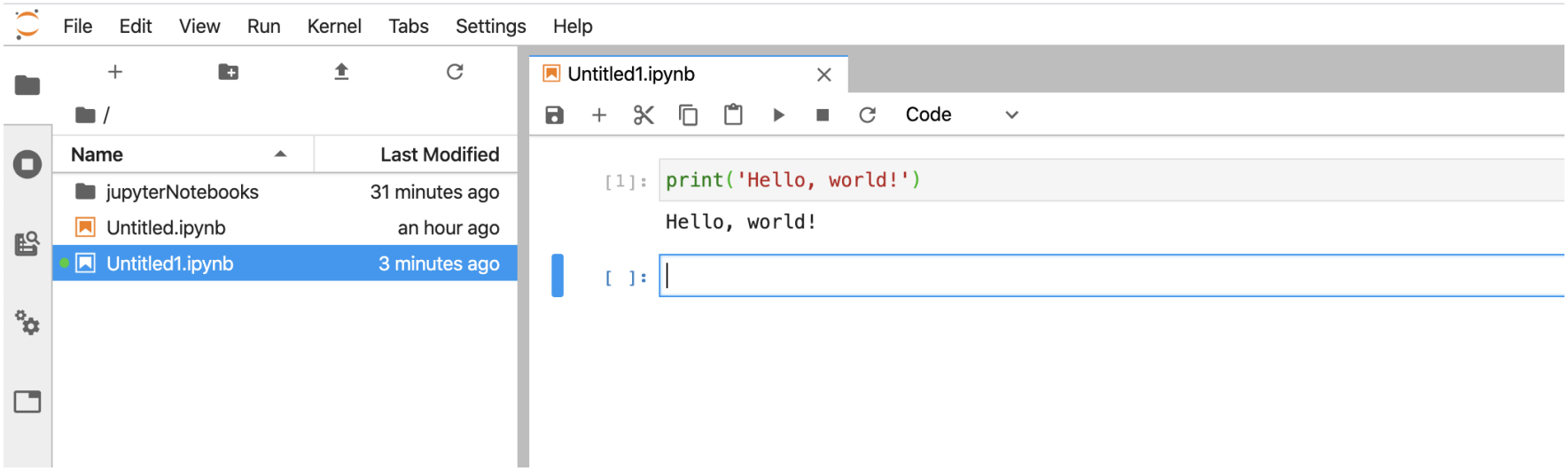
We have blocks of 3 hours.

- First part:
  Intro to week's topics & Live programming
  - ★ No slides
  - ★ We use interactive *notebooks* mixing presentation material and code
    - Please have your laptop ready! http://bit.ly/PDASSSA21_22
    - You find code in advance here
- Second part:
  You consolidate your understanding working on the assignments
  - ★ Begin working on the assignments with our live support if needed
  - ★ Complete them offline before next class. Contact us if needed

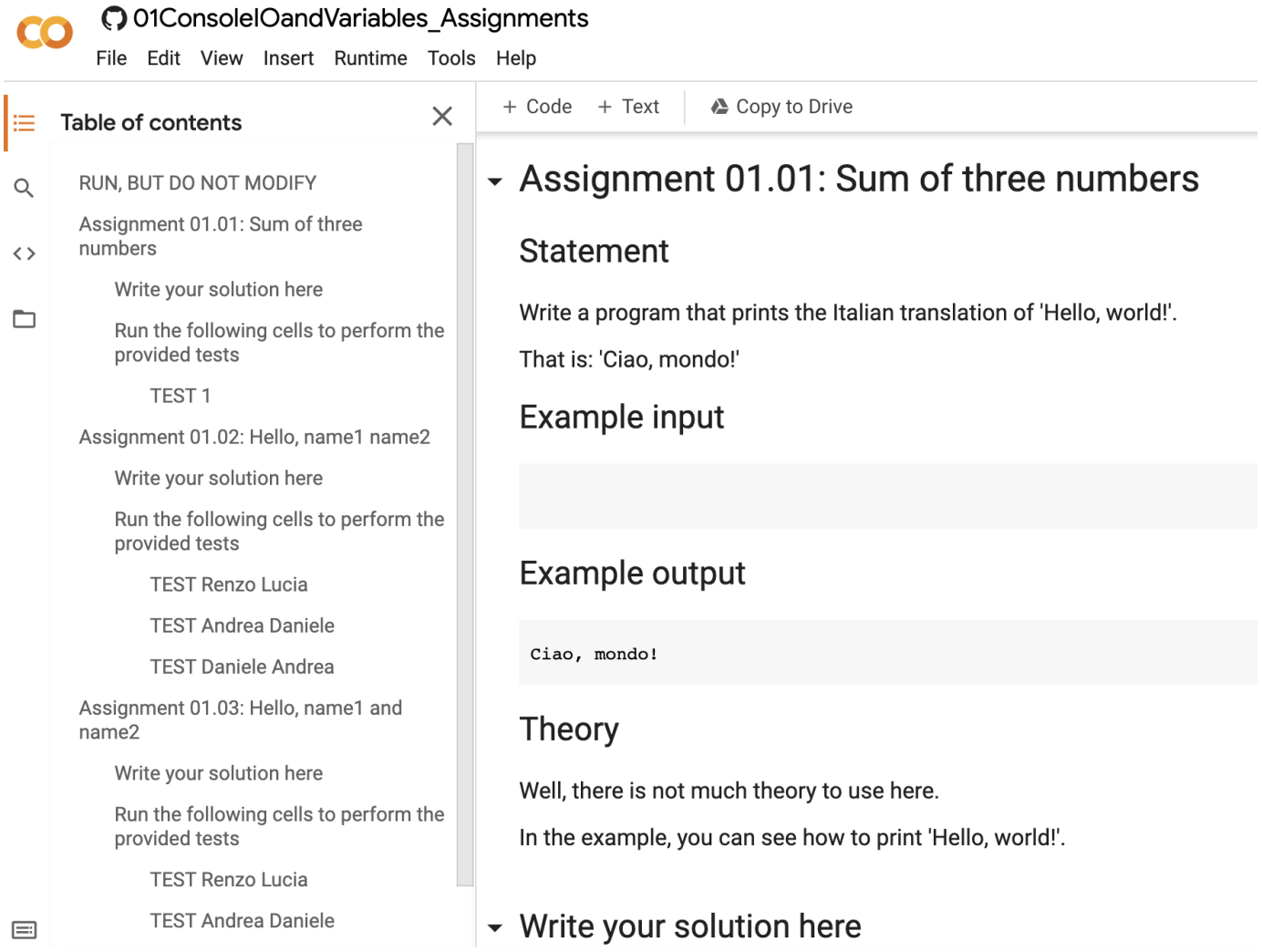
However, we have the ambitious goal of covering many topics necessary to introduce you to programming and data analytics in just 20 hours. Hence we might skip some second parts.

Live Programming

Find the JupyterLab notebooks at http://bit.ly/PDASSSA21_22



Colab



- Each lecture comes with a set of simple coding assignments
  - ★ Links available in the wiki page for slides and further material

Colab

- Colab is a Google service similar to Google docs
  - ★ but for python notebooks.
  - ★ no installation required
- Each set of assignments is actually a python notebook
- We implemented in Colab autograding functionalities
  - ★ to test your solution
- If you prefer, you can also download them as jupyter notebooks

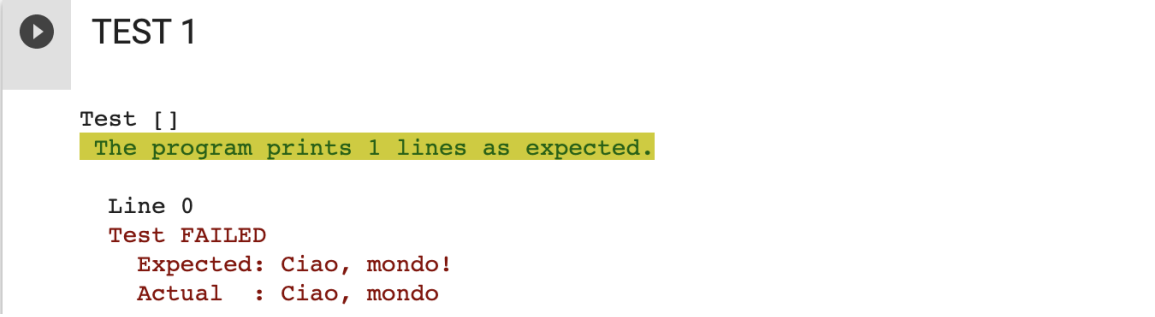
Colab: auto-testing

Write your solution here

- Do not change the first line (`def ...():`)
- Maintain the given indentation
- You can run some tests by yourself by decommenting the last line

```
[3] def asgn01_01Hello_world():
        # This program prints 'Hello, world!':
        #print('Hello, world!')

        # Can you change it so that it prints the same,
        #but in Italian?


        print('Ciao, mondo')

    #You can test independently your solution by executing the following line
    #asgn01_01Hello_world()
```

Run the following cells to perform the provided tests



Colab: auto-testing

Write your solution here

- Do not change the first line (`def ...():`)
- Maintain the given indentation
- You can run some tests by yourself by decommenting the last line

```
[ ]  def asgn01_01Hello_world():
         # This program prints 'Hello, world!':
         #print('Hello, world!')

         # Can you change it so that it prints the same,
         #but in Italian?


         print('Ciao, mondo!')

     #You can test independently your solution by executing the following line
     #asgn01_01Hello_world()
```

Run the following cells to perform the provided tests

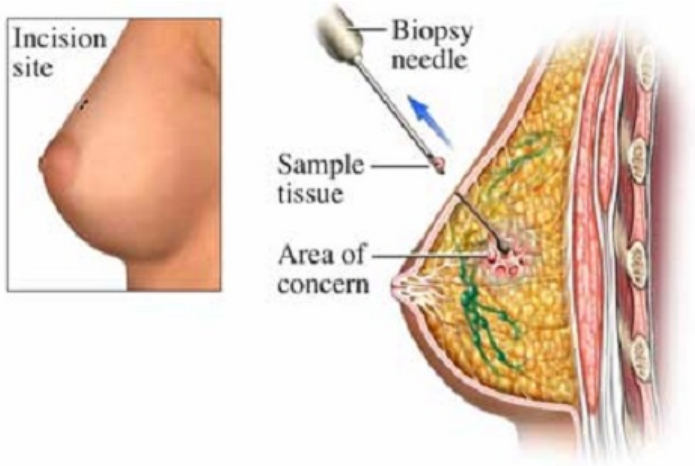
TEST 1

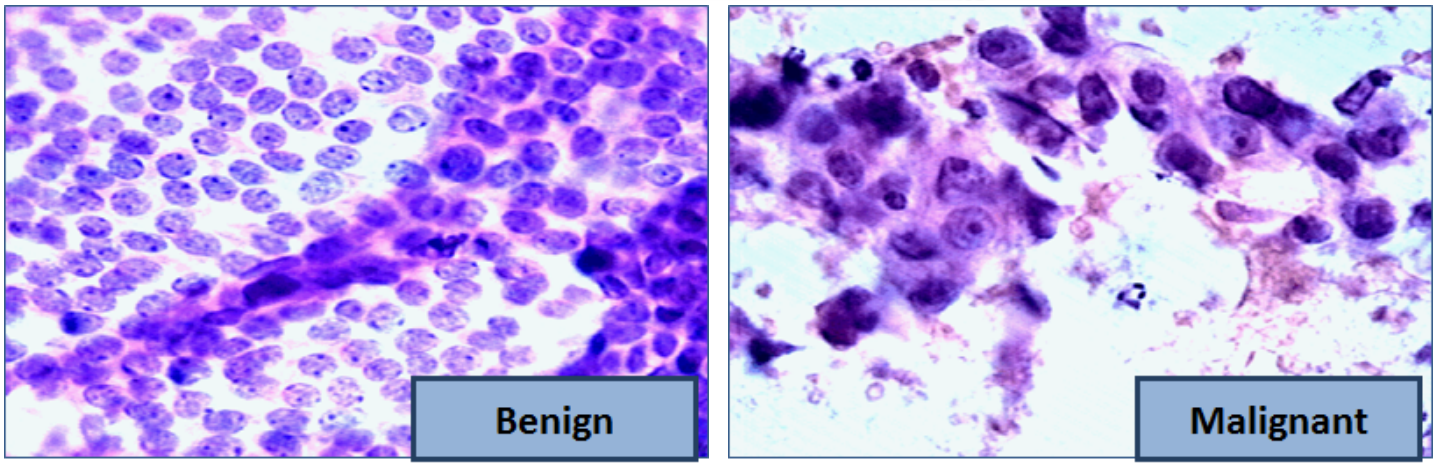
```
Test []
  The program prints 1 lines as expected.
 Line 0
  Expected and actual output match: Ciao, mondo!

Test PASSED!
```

Outline

Sneak preview of Module 2

Starting from the competences developed in the first module, we will study how to apply data analysis techniques from Machine learning





Can we classify them automatically?

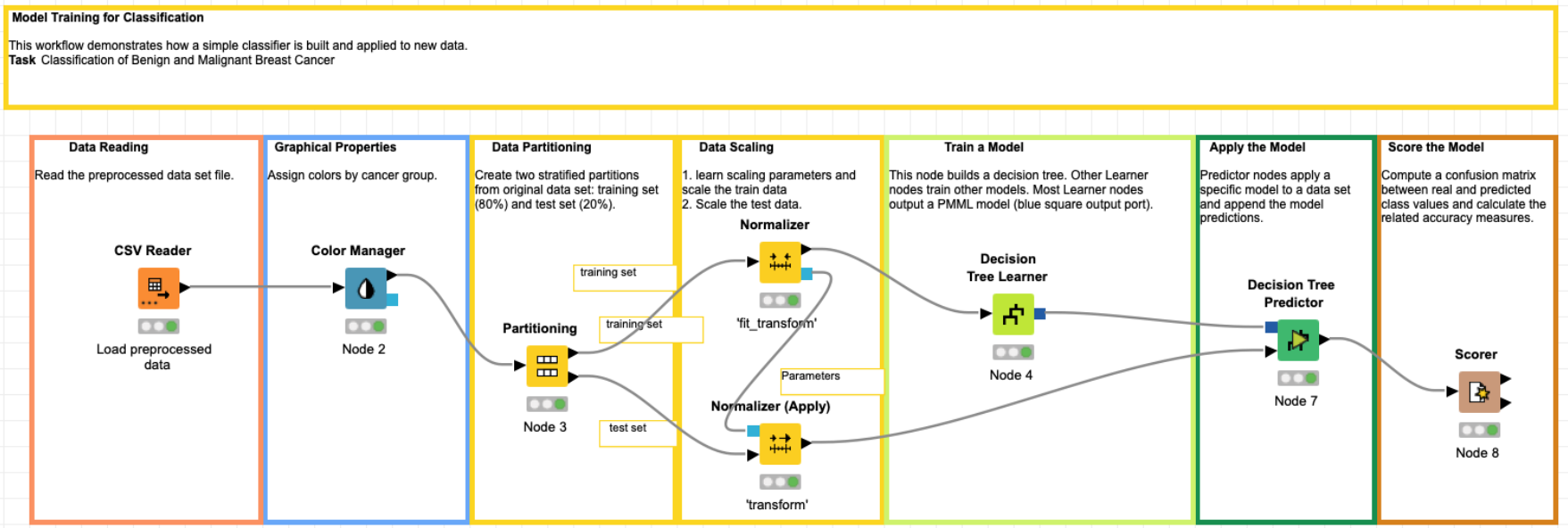
Sneak preview of Module 2

We will go through a classic pipeline for these data analysis tasks

- with emphasis on data pre-processing.

We will use two alternative approaches

- Python: **main focus**
- Knime: a graphical workflow language



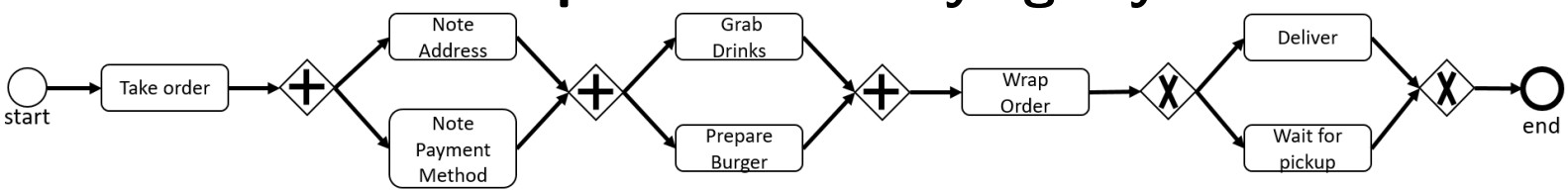
Sneak preview of Module 2

Further advanced research-oriented topics of data-driven analysis like Process Mining

1. ***Lucy*** *takes your order*
2. ***Lucy*** *notes down your address*
3. ***Lucy*** *notes down your preferred payment method*
4. ***Luigi*** *prepares your burger*
5. ***Lucy*** *grabs your can of soda*
6. ***Luigi*** *puts your burger in a box*
7. ***Lucy*** *wraps your order*
8. ***Mike*** *delivers your order*

1. ***Randy*** *takes your order*
2. ***Randy*** *notes down your preferred payment method*
3. ***Randy*** *notes down your address*
4. ***Luigi*** *prepares your burger*
5. ***Luigi*** *puts your burger in a box*
6. ***Randy*** *wraps your order*
7. ***John*** *delivers your order*

What is the process underlying my data?[1]



[1] Example from https://pm4py.fit.fraunhofer.de/

Sneak preview of Module 2

Evaluation

You will do the same on data of interest or on data on titanic sinking

- Would you have survived the sinking of the titanic?

Outline

Let's play a game on Kahoot!

- Using your smartphone or a second monitor
- Visit www.kahoot.it
- Type the code we will give you during the class

Outline

What is a program?

- A sequence of code instructions to control a machine
  - ★ Input/output
  - ★ Mathematical operations
  - ★ Conditional and repetitive executions
- A recipe to instruct a machine to execute instructions.
  - ★ We can't use a *natural language*.
  - ★ We need a **programming language**

Programming languages

The 7 Most In-Demand Programming Languages of 2019

March 15, 2019

Aspiring developers need to know what languages to learn; they need to select the right education and work on a skill set that will impress future employers and land their dream job. So what are the top programming languages? And what is the best one to learn? We've compiled a list for you that highlights the most in-demand programming languages based off current job postings on the market.

Here are the Top 7 programming languages with most job posting on Indeed as of January 2019:

- Java – 65,986 jobs
- Python – 61,818 jobs
- Javascript – 38,018 jobs
- C++ – 36,798 jobs
- C# – 27,521 jobs
- PHP – 16,890 jobs
- PERL – 13, 727 jobs

http://www.codingdojo.com/blog/the-7-most-in-demand-programming-languages-of-2019

Programming languages

https://www.tiobe.com/tiobe-index/

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Feb 2021	Feb 2020	Change	Programming Language	Ratings	Change
1	2	▲	C	16.34%	-0.43%
2	1	▼	Java	11.29%	-6.07%
3	3		Python	10.86%	+1.52%
4	4		C++	6.88%	+0.71%
5	5		C#	4.44%	-1.48%

Programming languages

https://www.tiobe.com/tiobe-index/

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Feb 2021	Feb 2020	Change	Programming Language	Ratings	Change
1	2	▲	C	16.34%	-0.43%
2	1	▼	Java	11.29%	-6.07%
3	3		Python	10.86%	+1.52%
4	4		C++	6.88%	+0.71%
5	5		C#	4.44%	-1.48%

May 2021	May 2020	Change	Programming Language	Ratings	Change
1	1		C	13.38%	-3.68%
2	3	▲	Python	11.87%	+2.75%
3	2	▼	Java	11.74%	-4.54%
4	4		C++	7.81%	+1.69%
5	5		C#	4.41%	+0.12%

The Python Programming language

- High-level: almost human readable. Abstracts from hardware
- Beginner-friendly:
  - ★ streamlined syntax
  - ★ it is easy to write your *first programs*
- Free, open-source and multi-platform
- Developed since the 90s, therefore it has
  - ★ A wide community, and its popularity keeps increasing
  - ★ Many predefined software modules

Python programs

- A sequence of python instructions to control a machine
- Python supports the most common programming styles
  - ★ Imperative: Statements are executed in sequence changing the state of the program (the variables)
  - ★ Procedural: The program is structured in reusable units named functions
  - ★ Object-oriented: The program is structured as a collection of interacting objects that send messages to each other.
  - ★ Functional: Statements are not written/executed as an ordered sequence of instructions. A computation is treated as the evaluation of a mathematical function.

Variables

Basic abstraction to represent units of data

A variable has a name and a value

- Names can contain any letter, number, or the underscore _

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Note:

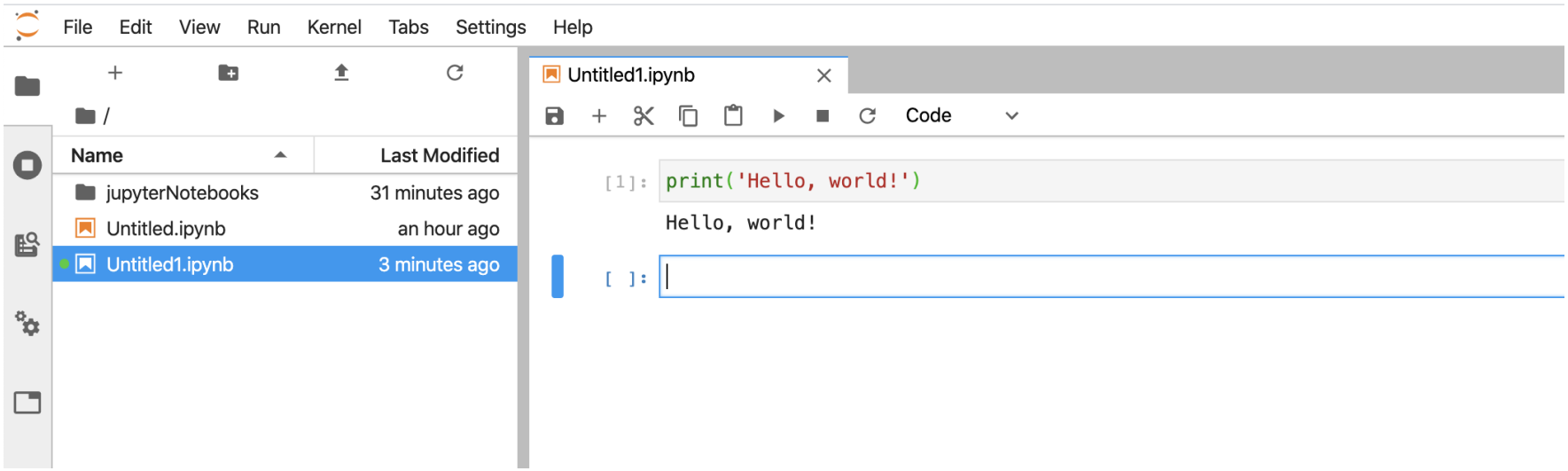
- ★ Cannot start with numbers
- ★ Cannot be a keyword
- ★ Names are case-sensitive

- We assign/update values to variables using assignment statements

```
month_number=3
month_name="April"
print("The number of",month_name,"is",month_number)
month_number=4
print("The number of",month_name,"is",month_number)
```

Live Programming

Find the JupyterLab notebooks at http://bit.ly/PDASSSA21_22



Configure your machine

If you have not done it yet

Follow the instructions in
http://bit.ly/PDASSSA21_22

"But it works . . ."

"Can You Learn To Ski Without Lessons?"

https://www.skibro.com/blog/en/can-you-learn-to-ski-without-lessons/

Most of the times you get to the valley.

The problem is how you get there . . .