Féidearthachtaí as Cuimse
Infinite Possibilities

Programming for Analytics

Lecture 1: Introduction

Bojan Božić
School of Computer Science
TU Dublin, Grangegorman

bojan.bozic@tudublin.ie

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Slides adapted from Sarah Jane Delany and book slides from: Fundamentals of Machine Learning for Predictive Data Analytics. Kelleher, Mac Namee and D'Arcy

Overview

- Administrivia
- Module Outline
- What is Python?
- Python Development Environments
- Variables, Expressions and Data Types
- How to write Python programs?

About me

- My name is Bojan (born in Croatia)
- PhD in Semantic Web from the University of Vienna
- TU Dublin Lecturer since 2019
- Research Areas: Knowledge Graphs (Semantic Web), Machine Learning (Neural Networks), LLMs

Administrivia

- Lecture and Lab:
  - Tuesday: 1pm to 3pm for lectures
  - Thursday: 1pm to 3pm for labs
  - Attendance for all labs and lectures is mandatory!
  - No lecture on week 5 (27.02.2025)!
- All notes, lecture recordings, tutorials, lab work, and assignments will be available on Brightspace (but can also be found on github: github.com/bozicb/programming-for-analytics)

- For all module queries please contact: bojan.bozic@tudublin.ie

Usage of AI Tools

- You are free to use AI Tools (such as ChatGPT) to help you understand materials or explain coding concepts
- Not recommended to use generated code!
- Keep in mind: Understanding of elementary coding concepts cannot be replaced by code generators and can cost you your future job and career.

Assessment

- Assessment is based on labs + assignment:

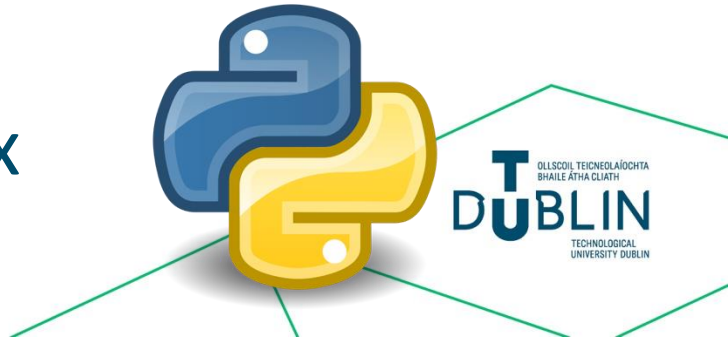
Percentage	Activity
70%	Assignment - due in week 13
30%	5 Lab Assessments (6% each)

Course Overview

Week	Lecture	Learning Outcomes	Lab Number	Lab Topic	Lab Marks	Assignment Specified	Assignment Due
1	Introduction	MLO1, ML02	1	Basic Programming			
2	Programming Constructs	MLO1, ML02	2	Setting up Github			
3	String Manipulation	ML01, ML03	3	String Parsing	6%		
4	Python Data Structures	ML01, ML03	4	Data Structures			
5	Working with Files	ML02, ML03, ML04	5	Reading/writing files	6%		
6	Introduction to NumPy	ML01, ML03	6	NumPy arrays and ops			
7	Review Week	Review Week					
8	Pandas	ML01, ML03	7	Datasets in Pandas	6%	70%	
9	Preprocessing	ML04	8	Data Cleaning			
10	APIs	ML04	9	Fetching and Parsing JSON	6%		
11	Matplotlib and Seaborn	ML05	10	Creating and customising plots			
12	Statistical Analysis	ML05	11	Summary Statistics	6%		
13	Ethics and Summary	All					70%

What is Python and why use it?

- High-level, interpreted programming language
- Great support for data analysis and machine learning
- Large ecosystems: NumPy, Pandas, Matplotlib, Scikit-learn
- Easy to learn, readable syntax

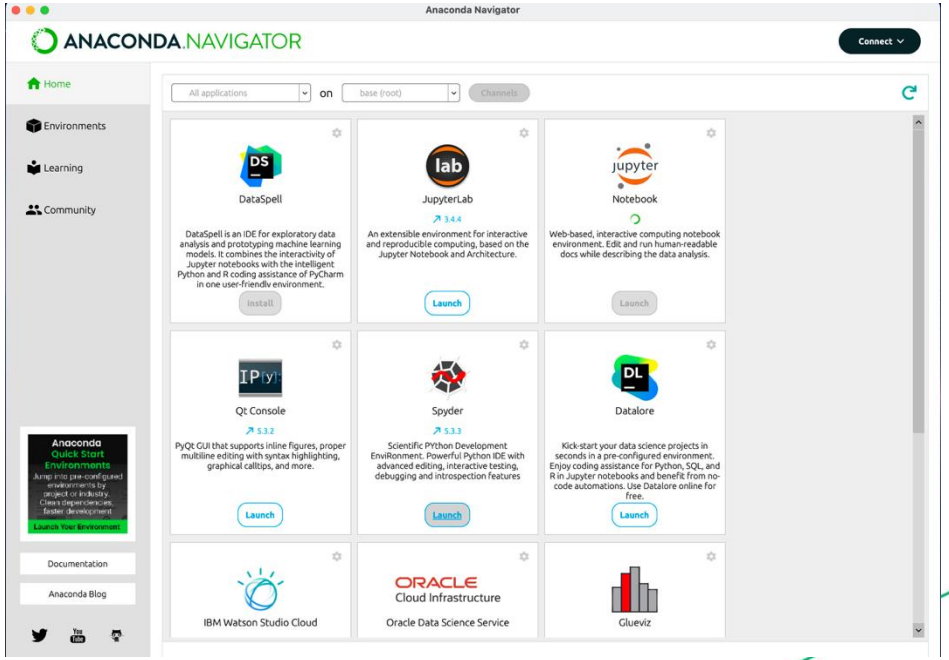


Setting up the Environment

- Install Anaconda (recommended) or Python via python.org
- Use Jupyter notebooks or your Editor/IDE of choice
- Set up Git for version control
- Test your setup by running some code

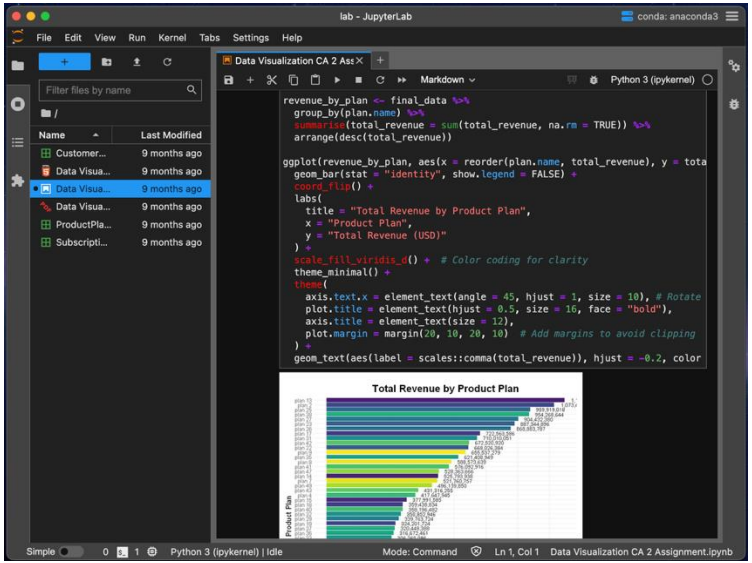
Anaconda

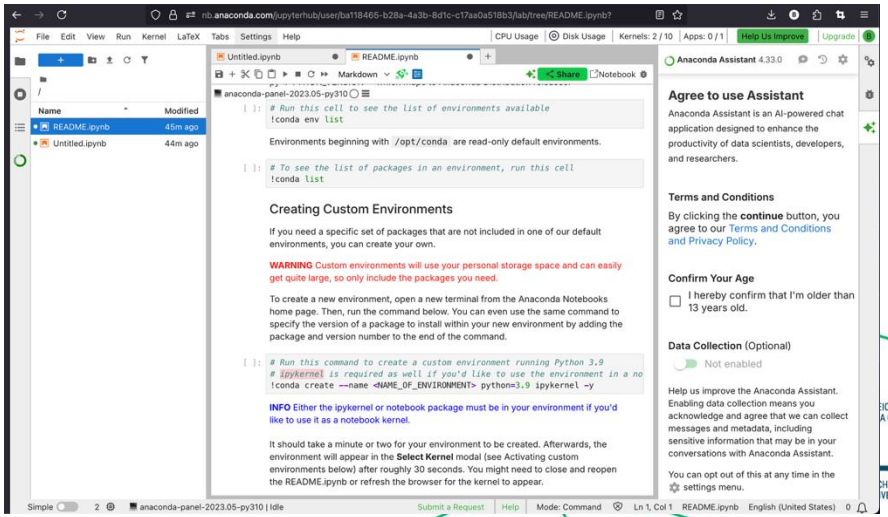
- Swiss knife for data analytics with Python



Jupyter Lab

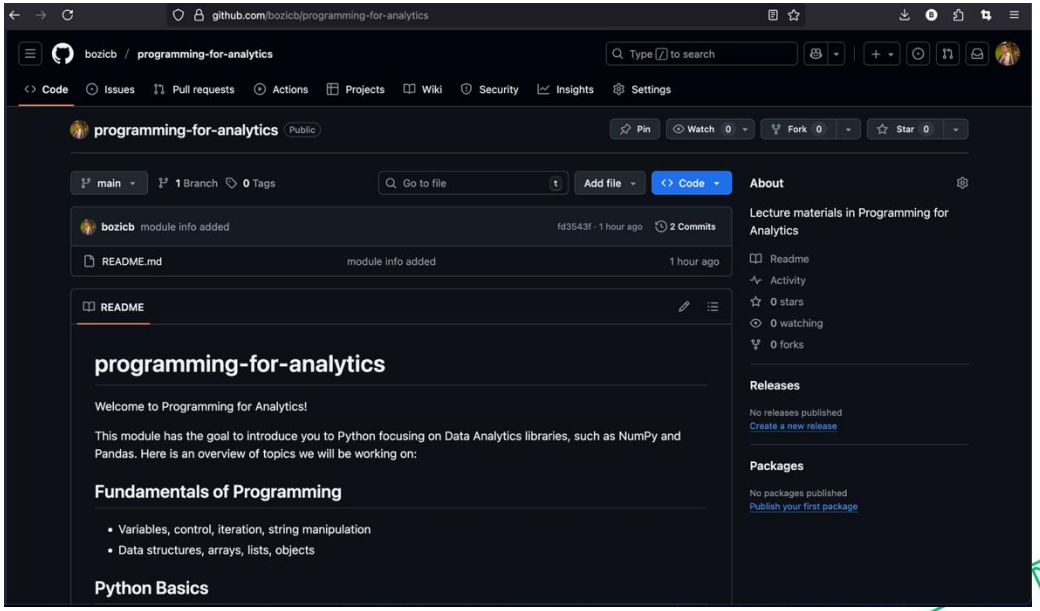
- Data Analytics environment with code and output (including graphs) all in one place





GitHub

- Version control repository for your Python code



Your First Python Script

- Open your Python environment or Jupyter notebook
- Type: `print("Hello World!")`
- Run the cell or script
- Reflect on basic syntax and output

Variables and Data Types

- Variables store values (e.g. `x = 5`)
- Common data types: int, float, str, bool
- Use `type()` to check a variable's data type
- Python is dynamically typed

Expressions and Operators

- Arithmetic operators: +, -, *, /, %, **
- Comparison operators: ==, !=, >, <, >=, <=
- Logical operators: and, or, not
- Use expressions to perform calculations and comparisons

Input and Output

- Use `input()` to get user input
- Use `print()` to display output
- Example:

```
name = input("Enter your name: ")
```

- Example:

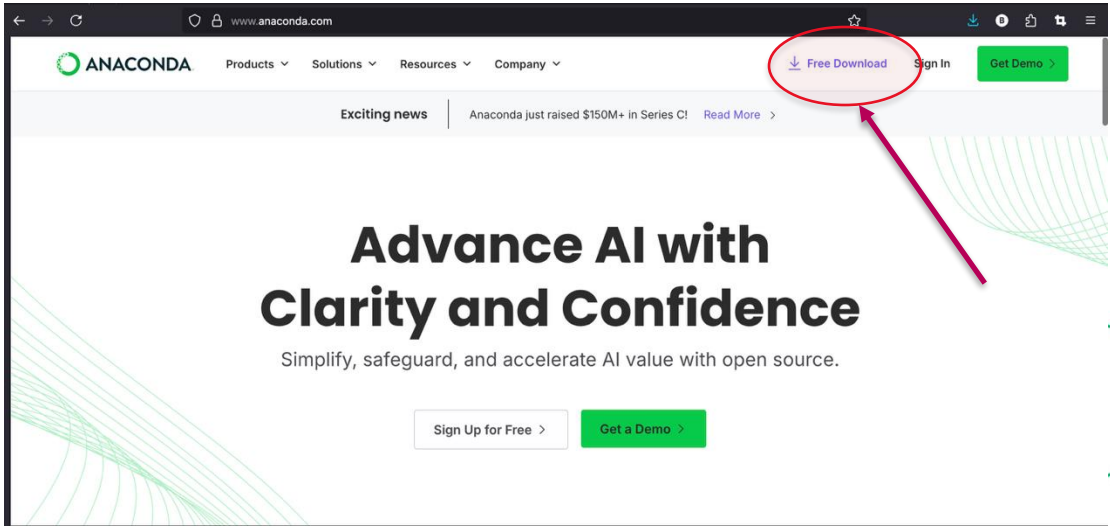
```
print("Hello, ", name)
```

Recap and Next Steps

- We introduced Python and basic programming concepts
- Set up your development environment
- Practice variables, expressions, input/output
- Next week: control structures (if, for, while, etc.), writing functions.

Installing Python with Anaconda

- Sign up and download from: anaconda.com (or use Web UI)
- Includes Pyton, Jupyter Notebooks and essential libraries
- Anaconda navigator for managing environments



Using Jupyter Notebooks

- Web-based (or desktop) environment for interactive coding
- Code cells and markdown cells
- Great for data exploration and analysis
- Shortcut: Shift + Enter to Run a cell

```
[3]: 1 + 1

[3]: 2
```

Github and Version Control

- Track changes to code and collaborate with others
- Key commands: git init, git add, git commit, git push
- We will use GitHub to manage our code!

```
programming-for-analytics on  main [?↑] took 9s
❯ git push
Username for 'https://github.com': bozicb
Password for 'https://bozicb@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 810 bytes | 810.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/bozicb/programming-for-analytics.git
   5e4ca74..fd3543f  main -> main
```

Working with Strings

- Strings are sequences of characters:

```
name = “Karsa Orlong”
```

- Use + to concatenate, * to repeat
- Common methods: `upper()`, `lower()`, `strip()`, `split()`, `replace()`
- Use f-strings for formatting:

```
f”Hello, {name}”
```

Common Python Errors

- Syntax Error: Typo or incorrect code structure
- NameError: Using a variable that hasn't been defined
- TypeError: Incompatible data types in operations
- Use tracebacks to understand where errors happen

Shell

```
$ python example.py
Traceback (most recent call last):
  File "/path/to/example.py", line 4, in <module>
    greet('Chad')
  File "/path/to/example.py", line 2, in greet
    print('Hello, ' + someon)
NameError: name 'someon' is not defined
```

Best Practices for Beginners

- Use clear variable names (e.g. `total_sales` instead of `ts`)
- Write comments to explain your code

```
# like this
```

- Test your code in small pieces
- Save and back up your work regularly using Git

Mini Lab Activity

- Task 1: Write a program that takes a user name + age and prints a message.
- Task 2: Create three variables (int, float, str) and print their types.
- Task 3: Use arithmetic operators to calculate the area of a rectangle.
- Bonus: Use string formatting to print a custom message.

Questions?