Introduction to Python language

Jisu Kim, Ph.D

What are your backgrounds?

What programming language do you normally use?

Materials

Slides + Codes are available here:

`https://github.com/jisukimmmm/NCCR_MWQTA_2024`

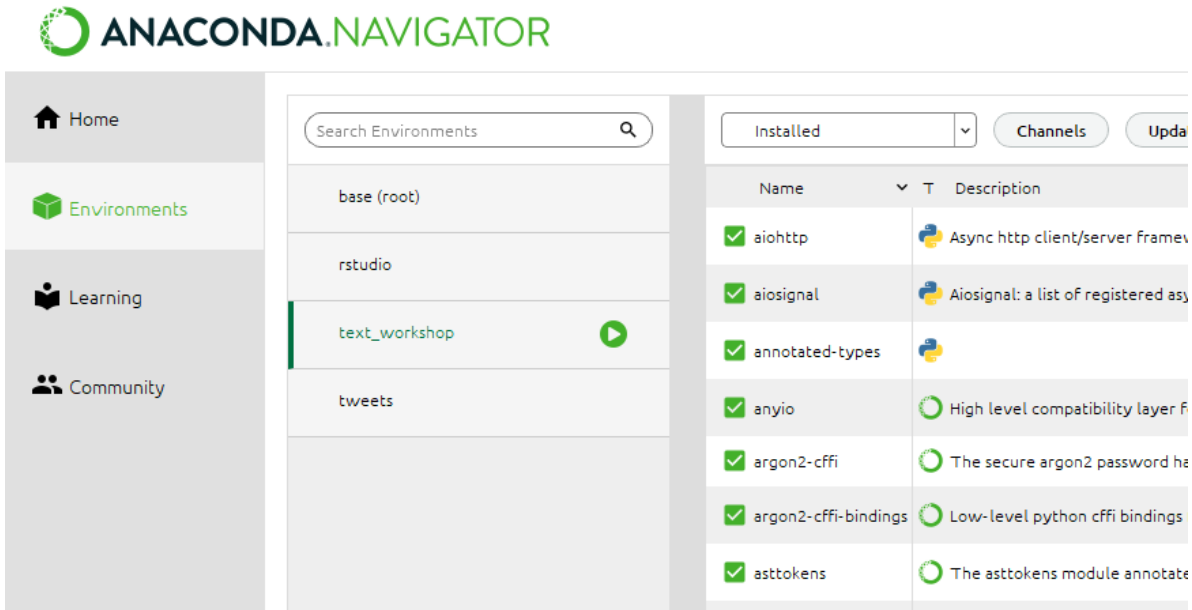
Running Python: Installing it yourself

There are many ways to install Python on your laptop/PC/etc.

- `https://www.python.org/downloads/`
- `https://www.anaconda.com/download/*`
- `https://www.spyder-ide.org/`

Anaconda is the most popular option

"Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment."



Package management

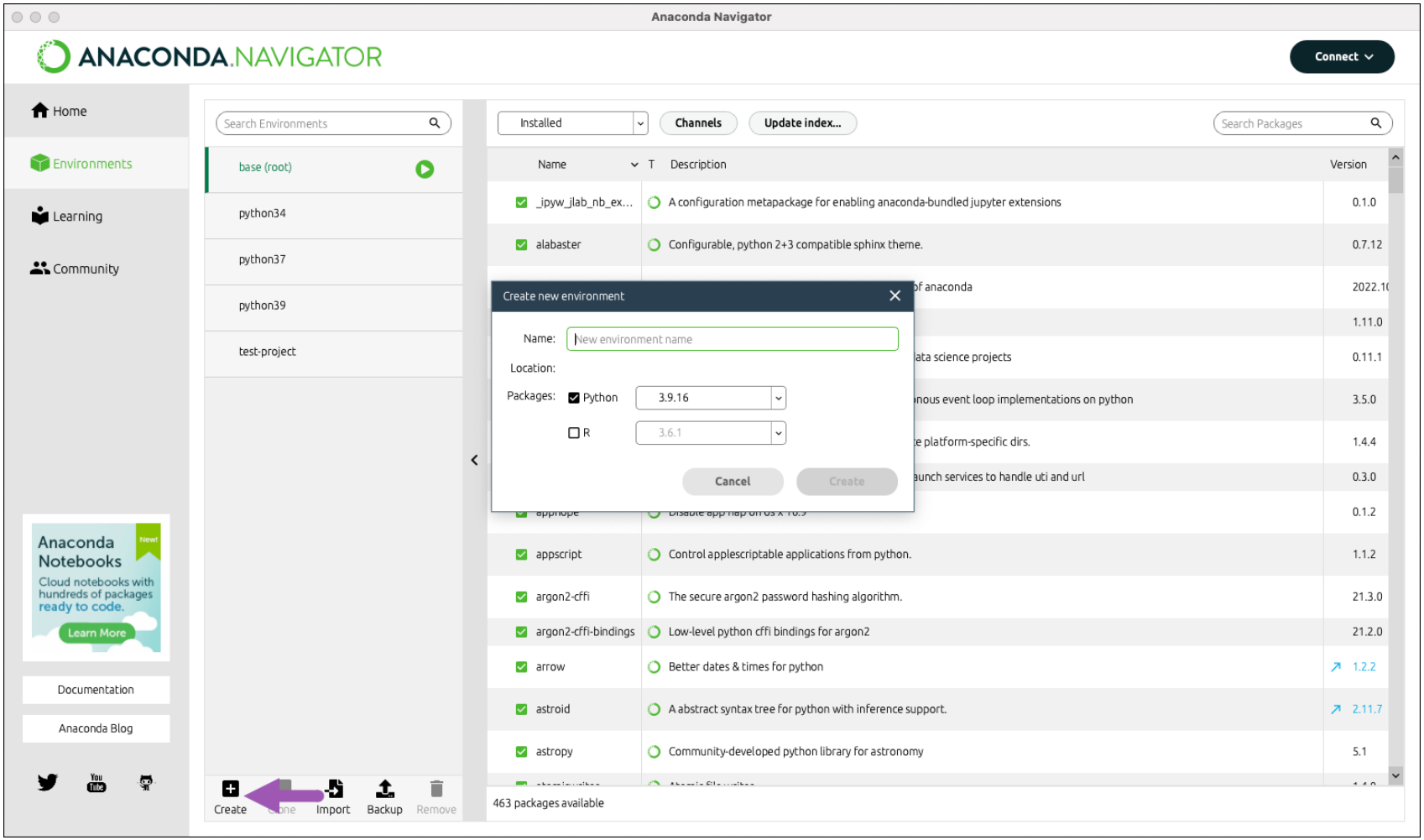
"Packages are managed separately for each environment. Changes you make to packages only apply to the active environment."[1]

"You can set up your own libraries and dependencies without affecting the system Python"

[1] https://docs.anaconda.com/free/navigator/tutorials/manage-environments

Package management

Create an environment:



There are also many software applications for Python

- Jupyter notebook `https://jupyter.org/`
- Jupyter Lab `https://jupyterlab.readthedocs.io/en/latest/`
- Visual Studio `https://visualstudio.microsoft.com/`
- ...

Why Python?

- Simplicity and readability
- Extensive libraries
- Community support
- Comparison with other languages (e.g., Java, C++)
- Most up to date - recent development libraries include AI, ML etc
- Open source - Free!

Zen of Python[2]

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. 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.[d] 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!

[2] Software engineer Tim Peters

Brief history of Python[3]

Dutch Programmer Guido Van Rossum in the late 1980s. And, Python's first version (0.9.0) was released in 1991.

The name of the Python programming language was inspired by a British Comedy Group Monty Python.

[3] https://www.tutorialspoint.com/python/python_history.htm

Python 2 vs. 3

- Python 2: released in 2000, Python 3 released in 2008
- Python 2 is in "maintenance mode" – no new features are expected
- Py3 is not completely compatible with Py2
- Differences are most negligible
- Choose between the two depending on who you are working with
- Otherwise, choose Python 3 - It is the most updated version

Next Steps

- Practice coding regularly
- Explore Python libraries for specific domains
- Contribute to open-source projects

Questions?

Basic Syntax

Indexing in Python starts at 0, which means that the first element in a sequence has an index of 0, the second element has an index of 1, and so on.

```
1 x=[1,2,3,4]
2 print(x[1])
3
4 Outcome: 2
```

Indentation

- Python uses indentation to define blocks of code.
- Consistent indentation (typically four spaces) is crucial for readability and proper interpretation by the Python interpreter.
- Example:

```
if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

Comments

- Comments in Python start with the # symbol and continue until the end of the line.
- Comments are used to document code, explain functionality, and make it more understandable.
- Example:

```
1 # This is a comment
2 print("Hello, world!")  # This is another
      comment
```

Variables

- Variables are used to store data values.
- Variable names can contain letters, digits, and underscores but cannot start with a digit.
- Variables are case-sensitive.
- Example:

```
x = 5
name = "Alice"
is_valid = True
```

Print Statement

- The `print()` function is used to display output in Python.
- It can take one or more arguments separated by commas.
- Example:

```
1 print("Hello, world!")
2 print("The value of x is", x)
```

Input Statement

- The `input()` function is used to accept user input in Python.
- It displays a prompt message (optional) and waits for the user to enter data.
- The entered data is returned as a string.
- Example:

```
name = input("Enter your name: ")
print("Hello,", name)
```

Data Types

- Numbers
- Strings
- Lists
- Tuples
- Dictionaries

Numbers

- Integers: Whole numbers (e.g., 5, -3, 0)
- Floats: Decimal numbers (e.g., 3.14, -0.001, 2.0)
- Operations: Arithmetic operations (+, -, *, /, //, %)
- Type Conversion: int(), float()

Strings

- Definition: Ordered sequence of characters enclosed in quotes (single or double)
- Examples:

```
name = "Alice"
message = 'Hello, world!'
```

- Operations: Concatenation (+), Repetition (*), Indexing, Slicing
- Common Methods: len(), upper(), lower(), strip(), split()

Lists

- Definition: Ordered collection of items enclosed in square brackets []
- Examples:
  - `numbers = [1, 2, 3, 4, 5]`
  - `names = [‘Alice’, ‘Bob’, ‘Charlie’]`
- Operations: Indexing, Slicing, Append, Extend, Insert, Remove, Pop
- List Comprehension

Tuples

- Definition: Ordered immutable collection of items enclosed in parentheses ()
- Examples:
  - `coordinates = (3, 5)`
  - `colors = ('red', 'green', 'blue')`
- Operations: Indexing, Slicing
- Use cases: Representing fixed collections of items (e.g., RGB color codes)

Dictionaries

- Definition: Unordered collection of key-value pairs enclosed in curly braces
- Examples:
  - `person = {'name': 'Alice', 'age': 30, 'city': 'New York'}`
  - `colors = {'R': 'red', 'G': 'green', 'B': 'blue'}`
- Operations: Accessing, Updating, Adding, Deleting
- Use cases: Storing related data with unique keys (e.g., user information, settings)

Conditional Statements

```
1 if condition:
2     # do something
3 else:
4     # do something else
```

Loops

```
1 for item in iterable:
2     # do something with item
3
4     while condition:
5         # do something
```

Defining Functions

```
1 def greet(name):
2     print("Hello,", name)
3
4 greet("Alice")
```

Advanced Topics

- List Comprehensions
- Error Handling
- Modules and Packages
- File Handling
- Object-Oriented Programming Basics

Useful libraries for plots

- matplotlib `https://matplotlib.org/`
- seaborn `https://seaborn.pydata.org/`