Introduction to Python

Programming Fundamentals for Artificial Intelligence

Jeremie Nlandu Mabiala

AMMI BootCamp 2025

African Masters of Machine Intelligence

August 5, 2025

AIMS African Institute for Mathematical Sciences SENEGAL

Overview I

Overview II

- Pandas for Data Manipulation
- Matplotlib for Visualization

5. Introduction to Machine Learning with Python
   - Scikit-learn Basics
   - Data Preprocessing

Why Python for AI and Machine Learning?

- **Simplicity and Readability:** Python's clean syntax allows you to focus on problem-solving rather than complex language features
- **Rich Ecosystem:** Comprehensive libraries for data science, machine learning, and scientific computing
- **Industry Standard:** Used by leading tech companies (Google, Netflix, Uber) and research institutions
- **Rapid Prototyping:** Quick development cycle for testing ideas and building prototypes
- **Community Support:** Large, active community with extensive documentation and tutorials

AIMS African Institute for Mathematical Sciences SENEGAL

Python in the AI/ML Pipeline

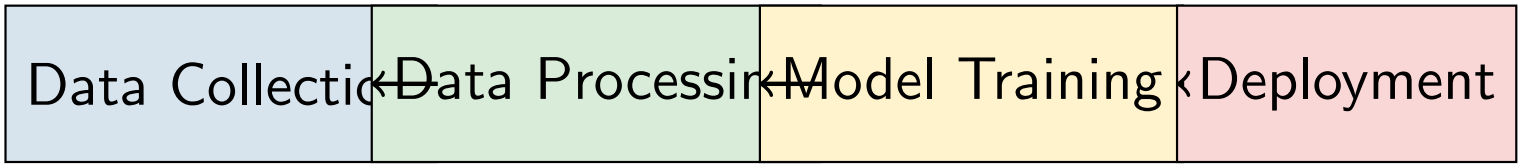

Figure: Python libraries supporting the entire AI/ML workflow

Key Python Libraries for AI/ML

Core Data Science:

- **NumPy**: Numerical computing
- **Pandas**: Data manipulation
- **Matplotlib/Seaborn**: Visualization
- **Jupyter**: Interactive development

Machine Learning:

- **Scikit-learn**: Classical ML
- **TensorFlow/PyTorch**: Deep learning
- **XGBoost**: Gradient boosting

Specialized Domains:

- **OpenCV**: Computer vision
- **NLTK/spaCy**: Natural language processing
- **NetworkX**: Graph analysis
- **Statsmodels**: Statistical modeling

Deployment & Production:

- **Flask/FastAPI**: Web APIs
- **Docker**: Containerization
- **MLflow**: ML lifecycle management

AIMS African Institute for Mathematical Sciences SENEGAL

Learning Path for AI/ML with Python

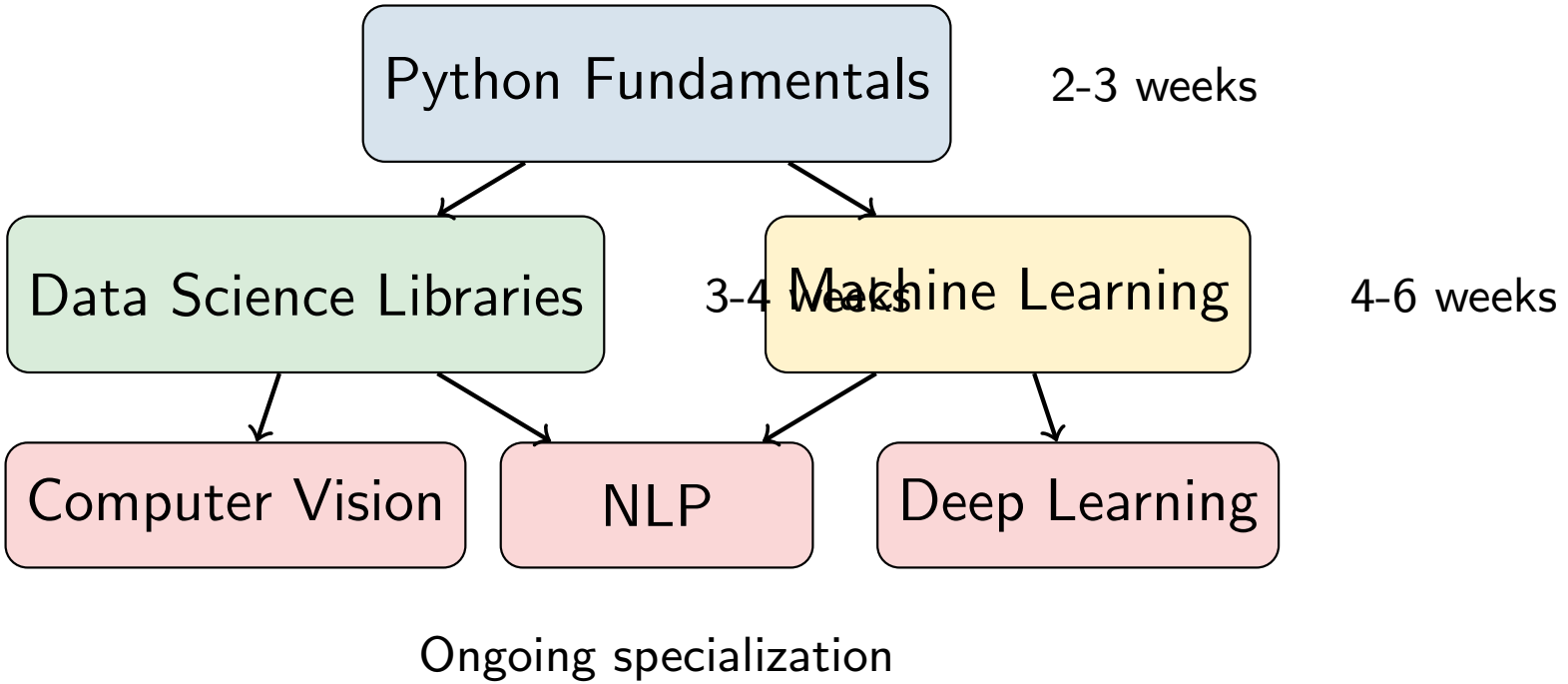

Figure: Recommended learning progression for AI/ML with Python

AIMS African Institute for Mathematical Sciences SENEGAL

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Getting Started with Python

- **Installation Options:**
  - **Anaconda**: Complete data science platform (recommended for beginners)
  - **Python.org**: Official Python distribution
  - **Google Colab**: Browser-based, no installation required
- **Development Environments:**
  - **Jupyter Notebook**: Interactive development and documentation
  - **VS Code**: Versatile editor with Python extensions
  - **PyCharm**: Full-featured Python IDE
  - **Spyder**: Scientific development environment
- **First Steps:**
  - Open a Python interpreter or Jupyter notebook
  - Try the classic first program: `print("Hello, World!")`

AIMS African Institute for Mathematical Sciences SENEGAL

Your First Python Program

Hello World Example

```python
# This is a comment - Python ignores this line
print("Hello, World!")
print("Welcome to Python for AI/ML!")

# Variables and basic operations
name = "AMMI Student"
year = 2025
print(f"Hello {name}, welcome to {year}!")

# Simple calculation
result = 10 + 5 * 2
print(f"The result is: {result}")
```

Interactive Python - REPL

- **REPL**: Read-Eval-Print-Loop for interactive programming
- Great for testing ideas and learning

Python Interactive Session

```
>>> 2 + 3
5
>>> name = "Python"
>>> print(f"I love {name}!")
I love Python!
>>> import math
>>> math.sqrt(16)
4.0
>>> help(print)   # Get help on any function
```

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Python Data Types Overview

Basic Types:

- **int**: Integers (1, 42, -5)
- **float**: Decimal numbers (3.14, -2.5)
- **str**: Text strings ("Hello", 'Python')
- **bool**: True or False

Collection Types:

- **list**: Ordered, mutable [1, 2, 3]
- **tuple**: Ordered, immutable (1, 2, 3)
- **dict**: Key-value pairs {"key":

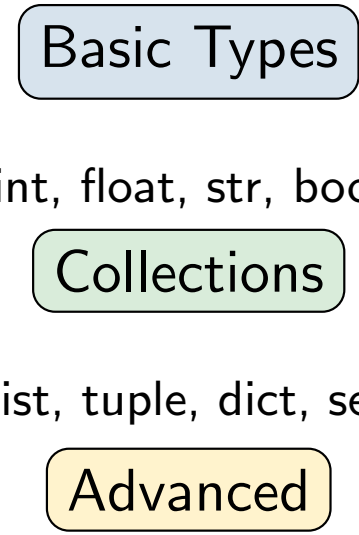

Figure: Python type hierarchy

Numbers and Basic Operations

Numeric Types and Operations

```python
# Integer operations
a = 10
b = 3
print(f"Addition: {a + b}")          # 13
print(f"Subtraction: {a - b}")       # 7
print(f"Multiplication: {a * b}")    # 30
print(f"Division: {a / b}")          # 3.333...
print(f"Floor division: {a // b}") # 3
print(f"Modulus: {a % b}")           # 1
print(f"Exponentiation: {a ** b}") # 1000

# Float operations
pi = 3.14159
radius = 5.0
area = pi * radius ** 2
```

Strings and Text Processing

String Operations

```
# String creation
name = "Alice"
greeting = 'Hello'
message = """This is a
multi-line string"""

# String formatting
age = 25
formatted = f"My name is {name} and I am {age} years old
    "
print(formatted)

# String methods
text = "  Python Programming  "
print(text.strip())                    # Remove whitespace
```

Variables and Assignment

Variable Assignment and Naming

```python
# Variable assignment
x = 5
y = 10
z = x + y

# Multiple assignment
a, b, c = 1, 2, 3
first_name, last_name = "John", "Doe"

# Variable naming conventions (PEP 8)
student_name = "Alice"        # Good: snake_case
CONSTANT_VALUE = 3.14159      # Good: UPPER_CASE for
    constants
class_size = 30               # Good: descriptive names

```

Boolean Logic and Comparisons

Boolean Operations

```python
# Boolean values
is_student = True
is_graduated = False

# Comparison operators
x, y = 10, 20
print(x == y)      # False (equal)
print(x != y)      # True (not equal)
print(x < y)       # True (less than)
print(x <= y)      # True (less than or equal)
print(x > y)       # False (greater than)
print(x >= y)      # False (greater than or equal)

# Logical operators
age = 22
```

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Conditional Statements - if/elif/else

Conditional Logic

```
# Basic if statement
score = 85

if score >= 90:
    grade = "A"
    print("Excellent work!")
elif score >= 80:
    grade = "B"
    print("Good job!")
elif score >= 70:
    grade = "C"
    print("Satisfactory")
else:
    grade = "F"
    print("Need improvement")
```

Loops - for and while

Iteration Structures

```
# For loop with range
print("Counting from 0 to 4:")
for i in range(5):
    print(i)

# For loop with list
fruits = ["apple", "banana", "orange"]
print("\nFruits in our basket:")
for fruit in fruits:
    print(f"- {fruit}")

# For loop with enumerate (index + value)
print("\nIndexed fruits:")
for index, fruit in enumerate(fruits):
    print(f"{index}: {fruit}")
```

Loop Control and List Comprehensions

Advanced Loop Techniques

```
# Loop control statements
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print("Even numbers:")
for num in numbers:
    if num % 2 == 0:
        print(num)

print("\nNumbers less than 6:")
for num in numbers:
    if num >= 6:
        break  # Exit the loop
    print(num)

print("\nOdd numbers (using continue):")
```

Nested Loops and Patterns

Nested Loop Examples

```python
# Multiplication table
print("Multiplication Table (3x3):")
for i in range(1, 4):
    for j in range(1, 4):
        product = i * j
        print(f"{i}x{j}={product:2d}", end="   ")
    print()  # New line after each row

# Pattern printing
print("\nStar pattern:")
for i in range(1, 6):
    print("*" * i)

# Matrix creation using nested loops
matrix = []
```

Control Flow Best Practices

- **Readable Conditions:**
  - Use descriptive variable names: `is_valid` instead of `flag`
  - Combine related conditions: `if 18 <= age <= 65:`
- **Loop Efficiency:**
  - Prefer `for` loops over `while` when possible
  - Use `enumerate()` instead of `range(len())`
  - Consider list comprehensions for simple transformations
- **Avoid Common Pitfalls:**
  - Don't modify a list while iterating over it
  - Use `break` and `continue` judiciously
  - Be careful with infinite loops in `while` statements
- **Code Style:**
  - Use 4 spaces for indentation (PEP 8)
  - Keep line length under 79 characters

AIMS African Institute for Mathematical Sciences SENEGAL

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Lists - Dynamic Arrays

List Operations

```
# Creating lists
empty_list = []
numbers = [1, 2, 3, 4, 5]
mixed = [1, "hello", 3.14, True]
nested = [[1, 2], [3, 4], [5, 6]]

# List methods
fruits = ["apple", "banana"]
fruits.append("orange")           # Add to end
fruits.insert(1, "grape")         # Insert at index
fruits.extend(["kiwi", "mango"])  # Add multiple items
print(fruits)  # ['apple', 'grape', 'banana', 'orange',
    'kiwi', 'mango']

# Removing items
```

List Slicing and Indexing

Advanced List Operations

```
# List slicing
data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

print(data[0])        # First element: 0
print(data[-1])       # Last element: 9
print(data[2:5])      # Elements 2 to 4: [2, 3, 4]
print(data[:3])       # First 3 elements: [0, 1, 2]
print(data[7:])       # From index 7 to end: [7, 8, 9]
print(data[::2])      # Every 2nd element: [0, 2, 4, 6, 8]
print(data[::-1])     # Reverse: [9, 8, 7, 6, 5, 4, 3, 2,
    1, 0]

# List copying
original = [1, 2, 3]
shallow_copy = original[:]          # or original.copy()
```

Tuples - Immutable Sequences

Tuple Operations

```python
# Creating tuples
empty_tuple = ()
single_item = (42,)  # Note the comma for single item
coordinates = (10, 20)
rgb_color = (255, 128, 0)
mixed_tuple = (1, "hello", 3.14)

# Tuple unpacking
x, y = coordinates
print(f"x: {x}, y: {y}")

red, green, blue = rgb_color
print(f"RGB: ({red}, {green}, {blue})")

# Multiple assignment using tuples
```

Dictionaries - Key-Value Pairs

Dictionary Operations

```python
# Creating dictionaries
empty_dict = {}
student = {
    "name": "Alice",
    "age": 22,
    "major": "Computer Science",
    "gpa": 3.8
}

# Accessing and modifying
print(student["name"])              # Alice
print(student.get("age", 0))        # 22 (with default)
student["age"] = 23                 # Update value
student["graduation_year"] = 2025   # Add new key
```

Sets - Unique Collections

Set Operations

```python
# Creating sets
empty_set = set()  # Note: {} creates an empty dict, not set
numbers = {1, 2, 3, 4, 5}
duplicates = {1, 2, 2, 3, 3, 3}  # Automatically removes duplicates
print(duplicates)  # {1, 2, 3}

# Set operations
set_a = {1, 2, 3, 4}
set_b = {3, 4, 5, 6}

print(f"Union: {set_a | set_b}")                    # {1, 2, 3, 4, 5, 6}
print(f"Intersection: {set_a & set_b}")             # {3, 4}
```

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Defining Functions

Function Basics

```
# Basic function definition
def greet(name):
    """This function greets someone."""
    return f"Hello, {name}!"

# Calling the function
message = greet("Alice")
print(message)  # Hello, Alice!

# Function with multiple parameters
def calculate_area(length, width):
    """Calculate rectangle area."""
    area = length * width
    return area
```

Modules and Packages

- **Importing Modules:**
  - `import math` - Import entire module
  - `from math import sqrt` - Import specific function
  - `import numpy as np` - Import with alias
- **Standard Library Modules:**
  - **math**: Mathematical functions
  - **random**: Random number generation
  - **datetime**: Date and time handling
  - **os**: Operating system interface
  - **json**: JSON data handling
- **Creating Your Own Modules:**
  - Save functions in a `.py` file
  - Import using the filename (without `.py`)
  - Use `if __name__ == "__main__":` for executable scripts

AIMS African Institute for Mathematical Sciences SENEGAL

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Working with Files in Python

- **File Operations:** Reading, writing, and processing files
- **Common File Types in Data Science:**
  - **CSV**: Comma-separated values for tabular data
  - **JSON**: JavaScript Object Notation for structured data
  - **TXT**: Plain text files
  - **Excel**: .xlsx files for spreadsheet data

Reading and Writing Files

File I/O Examples

```python
# Writing to a file
with open("example.txt", "w") as file:
    file.write("Hello, World!\n")
    file.write("Python for AI/ML\n")

# Reading from a file
with open("example.txt", "r") as file:
    content = file.read()
    print(content)

# Reading line by line
with open("example.txt", "r") as file:
    for line in file:
        print(line.strip())  # Remove newline characters
```

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Introduction to NumPy

- **NumPy**: Numerical Python - foundation for scientific computing
- **Key Features:**
  - N-dimensional array objects (ndarray)
  - Broadcasting functions
  - Tools for integrating C/C++ and Fortran code
  - Linear algebra, Fourier transform, and random number capabilities

</> NumPy Basics

```python
import numpy as np

# Creating arrays
arr1d = np.array([1, 2, 3, 4, 5])
arr2d = np.array([[1, 2, 3], [4, 5, 6]])
zeros = np.zeros((3, 3))
ones = np.ones((2, 4))
```

Why NumPy for AI/ML?

- **Performance:** 10-100x faster than pure Python lists
- **Memory Efficiency:** Contiguous memory layout
- **Broadcasting:** Vectorized operations without explicit loops
- **Foundation:** Base for pandas, scikit-learn, TensorFlow
- **Interoperability:** Works seamlessly with other libraries

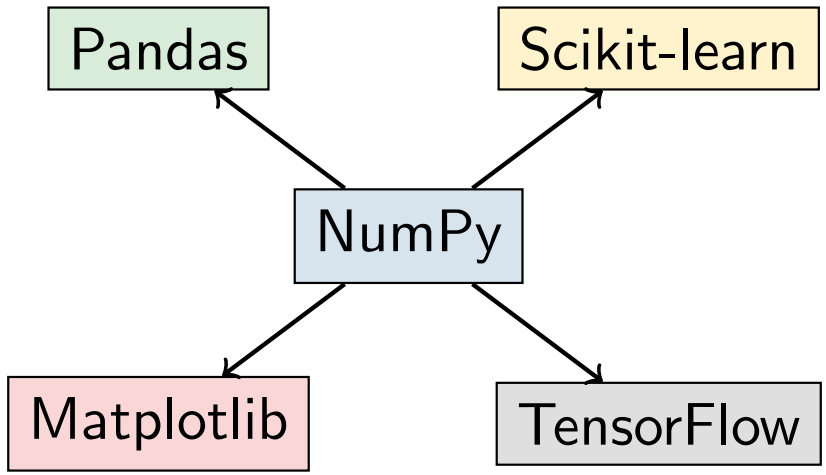

Figure: NumPy as the foundation of the Python data science ecosystem

Section Overview

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Data Visualization with Matplotlib

- **Matplotlib**: Comprehensive plotting library
- **Plot Types:** Line plots, bar charts, histograms, scatter plots, and more
- **Customization:** Colors, labels, legends, annotations
- **Integration:** Works seamlessly with NumPy and Pandas

Creating Basic Plots

Matplotlib Examples

```
import matplotlib.pyplot as plt
import numpy as np

# Line plot
x = np.linspace(0, 10, 100)
y = np.sin(x)

plt.figure(figsize=(10, 6))
plt.plot(x, y, label='sin(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Sine Wave')
plt.legend()
plt.grid(True)
plt.show()
```

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL

Introduction to Machine Learning with Scikit-learn

- **Scikit-learn**: Simple and efficient tools for data mining and analysis
- **Key Features:**
  - Classification, regression, clustering
  - Model selection and evaluation
  - Data preprocessing
  - Feature selection and extraction
- **Consistent API:** All algorithms follow the same interface pattern

AIMS African Institute for Mathematical Sciences SENEGAL

First Machine Learning Model

Simple Classification Example

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load sample dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Create and train model
```

Section Overview

AIMS African Institute for Mathematical Sciences SENEGAL