Introduction to Python

Dr. Daniel Genkins
Digital Library Architect
Digital Lab, Heard Libraries

Fundamentals of Python

Data structures

What is a list?

- An ordered collection of items (elements) that can be of any data type
- Lists are mutable, meaning their elements can be changed
- Useful for storing sequences of related items
- Can contain mixed data types, including other lists
- Common use cases
  - Managing collections of data such as a list of names or numbers
  - Iterating over elements to perform operations
  - Dynamic arrays where items can be added or removed

What is a dictionary?

- An unordered collection of key-value pairs
- Keys are unique and used to access corresponding values
- Efficient for storing and retrieving data based on keys
- Keys can be of any immutable type (e.g., strings, numbers, tuples)
- Common use cases
  - Storing configuration settings or user information
  - Creating databases of items with unique identifiers
  - Counting occurrences of items using keys

What is a tuple? What is a set?

- A tuple is an ordered collection of items that are immutable
    - once created, elements cannot be changed
- Useful for storing related items that should not be modified
- Can contain mixed data types
- Common Use Cases:
    - Storing coordinates, dates, or other fixed collections of items
    - Using as keys in dictionaries when immutability is required

- A set is an unordered collection of unique items
- Sets are mutable, but items must be immutable
- Automatically removes duplicate items
- Common use cases
    - Storing unique elements from a list
    - Performing set operations like union, intersection, and difference

Exercises with data structures!

AI for programming

How can AI help you program?

- Code completion
  - AI tools can predict and complete code snippets based on context
  - Speeds up coding by reducing keystrokes and minimizing syntax errors
- Bug Detection
  - Automated identification of common coding errors and potential bugs
  - Early detection helps maintain code quality and reduces debugging time
- Code generation
  - Generate boilerplate code, repetitive structures, or entire functions
  - Useful for quickly prototyping and scaffolding new projects
- Documentation
  - Automatically generate documentation comments and explanations for code
  - Helps maintain up-to-date and consistent documentation

What isn’t AI good at (yet)?

- Complex problem solving
  - AI tools are limited in solving complex, abstract problems that require deep understanding and creativity
  - Human intuition and problem-solving skills are still crucial
- Context-specific decisions
  - Struggles with tasks that require a deep understanding of project-specific context
  - Important architectural and design decisions should not be solely reliant on AI
- Original algorithm design
  - Designing new algorithms and data structures often requires innovative thinking beyond the capabilities of AI tools
  - AI can assist but not replace the need for original thought and expertise
- Ethical and secure coding
  - AI may inadvertently suggest insecure or unethical coding practices
  - Human oversight is necessary to ensure adherence to ethical standards and security best practices

Control structures and functions

Dr. Daniel Genkins
Digital Library Architect
Digital Lab, Heard Libraries

Python control structures

If statements

What are if statements?

- A conditional statement that executes a block of code if a specified condition is true, allowing for decision-making in programs
- Common use cases
    - Checking user input or validating data
    - Implementing decision-making logic in algorithms
    - Managing different outcomes in games or simulations

Loops

What are for loops?

- A control flow statement for iterating over a sequence (e.g., list, tuple, dictionary, set, string) that executes a block of code multiple times, once for each item in the sequence
- Common use cases
    - Iterating through elements in a data structure
    - Performing operations on each item in a list or array
    - Generating repetitive outputs or patterns

What are while loops?

- A control flow statement that executes a block of code as long as a specified condition is true
- Common use cases
  - Waiting for user input or external events
  - Running tasks until a certain condition is met
  - Implementing algorithms that require repeated operations until convergence

Functions

What are functions?

- A block of reusable code designed to perform a specific task
- Can be called (invoked) with a set of arguments to execute its code
- Breaks down complex programs into simpler, manageable pieces
- Encourages code reuse and reduces redundancy
- Common use cases
  - Performing specific tasks like calculations or data processing
  - Organizing code logically into separate functional units
  - Facilitating debugging and testing by isolating specific parts of the code

Examples of and exercises with functions

Working with packages and files

Dr. Daniel Genkins
Digital Library Architect
Digital Lab, Heard Libraries

Packages

What are packages?

- Collections of modules bundled together to provide specific functionality
  - Modules are files containing Python code (functions, classes, variables)
- Packages allow you to reuse code across different projects
- Avoid reinventing the wheel by using pre-built packages
- Helps in organizing code into manageable parts
- Makes complex projects more maintainable
- Python's standard library includes packages for various tasks (e.g., os for operating system interactions, math for mathematical functions)

Working with files

Why work with files?

- Allows storing data for later use
- Advantages: efficiency, interoperability, and automation
- Common use cases
    - Text files: Reading and writing plain text data (e.g., logs, configuration files)
    - CSV files: Handling comma-separated values for tabular data
    - Binary files: Reading and writing non-text data (e.g., images, executable files)

Examples of and exercises about working with packages and files

More packages and files

Beginner Python exercises

Recap, review, and putting it all together

Want to create your own Python package?

You’ve learned a lot in 3 weeks

Python intro recap

- **Variables:** Store data values using **assignment statements** like `x = 5`. Variables can hold different types of data including integers, floats, strings, and more.
- **Data types:** Python supports various data types such as **integers** (`int`), **floating-point numbers** (`float`), **strings** (`str`), and **booleans** (`bool`). Understanding data types is essential for working with data efficiently.
- **Basic operations:** Python includes mathematical operations such as **addition** (`+`), **subtraction** (`-`), **multiplication** (`*`), and **division** (`/`). There are also **comparison operators** (`==`, `!=`, `<`, `>`, `<=`, `>=`) for comparing values.

Python intro recap

- **Lists:** Store ordered collections of items using **lists**, which are mutable and can hold different data types.
- **Dictionaries:** Store data in **key-value pairs** using **dictionaries**. Dictionaries are mutable and are used for fast lookups.
- **Tuples:** Immutable sequences similar to lists, created using parentheses.
- **Sets:** Unordered collections of unique elements. Useful for membership tests and eliminating duplicates.

Python intro recap

- **Control structures:** Python uses **if statements** to execute code conditionally. **Elif** and **else** provide additional branches for complex conditions.
- **Loops:** Python supports **for loops** for iterating over sequences (e.g., lists, strings) and **while loops** for repeating actions while a condition holds true.
- **Functions:** Create reusable blocks of code using the `def` keyword. Functions can take parameters and return values.

Python intro recap

- **File I/O:** Python can **read from** and **write to** files. The open( ) function is used to access files.
- **Packages and modules:** Extend Python's functionality by importing built-in and third-party packages.
- **Error handling:** Manage errors using **try-except blocks** to prevent crashes and handle exceptions.

Python intro review