Python's Core Data Types

String (str) 📝

Strings are ordered, immutable sequences of characters used for storing text. They can be created using single, double, or triple quotes.

- **Key Attributes**: None.
- **Key Methods**:
  - .upper(): Converts a string to uppercase.
  - .lower(): Converts a string to lowercase.
  - .strip(): Removes leading and trailing whitespace.
  - .split(delimiter): Splits a string into a list of substrings based on a delimiter.
  - .join(iterable): Joins the elements of an iterable into a single string.
  - .replace(old, new): Replaces occurrences of old with new.
  - len(string): Returns the length of the string.

Integer (int) and Float (float) 🔢

Integers are whole numbers (positive, negative, or zero), while floats are numbers with decimal points. Both are immutable.

- **Key Attributes**: None.
- **Key Methods**: None.
- **Key Operations**: Arithmetic operations like +, -, *, /, ** (exponentiation), and % (modulus).

Tuple (tuple) 📦

Tuples are ordered, immutable collections of heterogeneous items. They are defined using parentheses ().

- **Key Attributes**: None.
- **Key Methods**:
  - .count(item): Returns the number of times item appears in the tuple.
  - .index(item): Returns the index of the first occurrence of item.
  - len(tuple): Returns the number of items in the tuple.

List (list) 📋

Lists are ordered, mutable collections of heterogeneous items. They are defined using square brackets [].

- **Key Attributes**: None.
- **Key Methods**:
  - .append(item): Adds an item to the end of the list.
  - .extend(iterable): Appends all items from an iterable to the end of the list.
  - .insert(index, item): Inserts an item at a specific index.

    - .remove(item): Removes the first occurrence of item.
    - .pop(index): Removes and returns the item at a given index (defaults to the last item).
    - .sort(): Sorts the list in place.
    - len(list): Returns the number of items in the list.

Dictionary (dict) 📚

Dictionaries are unordered collections of key-value pairs, where keys must be unique and immutable. They are defined using curly braces {}.

- **Key Attributes**: None.
- **Key Methods**:
    - .keys(): Returns a view object of the dictionary's keys.
    - .values(): Returns a view object of the dictionary's values.
    - .items(): Returns a view object of key-value pairs.
    - .get(key, default): Returns the value for a key, or a default value if the key is not found.
    - .pop(key): Removes the item with the given key and returns its value.

Set (set) 🧩

Sets are unordered collections of unique, immutable items. They are useful for membership testing and removing duplicates. They are defined using curly braces {} or the set() constructor.

- **Key Attributes**: None.
- **Key Methods**:
    - .add(item): Adds an item to the set.
    - .remove(item): Removes an item from the set.
    - .union(other_set): Returns a new set with all items from both sets.
    - .intersection(other_set): Returns a new set with items common to both sets.

Data Science-Specific Data Types

NumPy ndarray 📊

NumPy arrays are multi-dimensional, fixed-size arrays used for numerical operations. They are significantly faster and more memory-efficient than Python lists for numerical tasks.

- **Key Attributes**:
    - .shape: A tuple representing the dimensions of the array.
    - .dtype: The data type of the elements in the array.
    - .size: The total number of elements in the array.
    - .ndim: The number of dimensions.

- **Key Methods**:
  - .reshape(new_shape): Returns a new array with a different shape without changing the data.
  - .T: Returns the transpose of the array.
  - .sum(), .mean(), .std(): Aggregation functions that return the sum, mean, and standard deviation of the array elements.
  - np.sqrt(), np.exp(): Mathematical functions that operate element-wise.

Pandas Series and DataFrame 📈

Series is a one-dimensional labeled array, similar to a column in a spreadsheet.

- **Key Attributes**:
  - .index: The index labels of the Series.
  - .values: The data as a NumPy array.
  - .dtype: The data type of the elements.
  - .name: The name of the Series.
- **Key Methods**:
  - .head(n): Returns the first n rows.
  - .tail(n): Returns the last n rows.
  - .describe(): Generates descriptive statistics.
  - .unique(): Returns the unique values in the Series.
  - .value_counts(): Returns a Series containing counts of unique values.

DataFrame is a two-dimensional labeled data structure with columns of potentially different types, similar to a spreadsheet.

- **Key Attributes**:
  - .index: The index labels of the rows.
  - .columns: The column labels.
  - .shape: A tuple representing the dimensions (rows, columns).
  - .dtypes: A Series containing the data type of each column.
- **Key Methods**:
  - .head(n), .tail(n): Returns the first or last n rows.
  - .info(): Prints a concise summary of the DataFrame.
  - .describe(): Generates descriptive statistics for numerical columns.
  - .dropna(): Removes rows or columns with missing values.
  - .fillna(value): Fills missing values with a specified value.
  - .groupby(column): Groups the DataFrame by a column for aggregation.
  - .merge(), .join(): Combines DataFrames.

Common Programming Constructs

Looping 🔁

Loops are used to iterate over a sequence (like a list or tuple) or perform a block of code repeatedly.

- **for loop**: Iterates over the items of a sequence.

  Python

  ```
  my_list = [1, 2, 3]
  for item in my_list:
      print(item)
  ```

- **while loop**: Continues as long as a condition is true.

  Python

  ```
  count = 0
  while count < 3:
      print(count)
      count += 1
  ```

Functions 🛠️

Functions are reusable blocks of code that perform a specific task. They are defined using the def keyword.

Python
```
def greet(name):
    """This function prints a greeting."""
    print(f"Hello, {name}!")

greet("Alice")
```

Conditional Statements ⚖️

if, elif, and else statements are used to execute different code blocks based on conditions.

Python
```
x = 10
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is 5")
else:
    print("x is less than 5")
```

Comprehensions ✍️

Comprehensions provide a concise way to create lists, dictionaries, or sets.

- **List comprehension**:

  Python

  ```
  squares = [x**2 for x in range(5)]
  # Output: [0, 1, 4, 9, 16]
  ```

- **Dictionary comprehension**:

  Python

  ```
  my_dict = {key: key*2 for key in ["a", "b", "c"]}
  # Output: {'a': 'aa', 'b': 'bb', 'c': 'cc'}
  ```