Getting started with lists

A list is an ordered and changeable sequence of elements. It can hold integers, characters, floats, strings, and even objects.

Creating lists

A list can contain different types of elements:

```python
my_list = [1, 'hello', 3.14, True]
```

List functions and methods

- `len(list)` - Returns the length of the list.
- `max(list)` - Returns the largest item in the list.
- `min(list)` - Returns the smallest item in the list.
- `list.append(x)` - Adds `x` to the list.
- `list.remove(x)` - Removes the first occurrence of `x` from the list.
- `list.pop()` - Removes and returns the last item in the list.
- `list.pop(index)` - Removes and returns the item at the specified index.
- `list.insert(index, x)` - Inserts `x` into the list at the specified index.
- `list.extend(list2)` - Extends the list by appending all the items from `list2`.

Selecting list elements

- `list[0]` - Returns the first item in the list.
- `list[-1]` - Returns the last item in the list.
- `list[0:3]` - Returns a slice of the list from index 0 to 3 (inclusive).
- `list[:3]` - Returns a slice of the list from the beginning to index 3 (inclusive).
- `list[3:]` - Returns a slice of the list from index 3 to the end.
- `list[::2]` - Returns every second item in the list.
- `list.index(x)` - Returns the index of the first occurrence of `x` in the list.
- `list.count(x)` - Returns the number of occurrences of `x` in the list.

Getting started with dictionaries

A dictionary stores data using a key-value pair. Each key is a unique identifier, and the value is any data type.

Creating dictionaries

A dictionary is created using curly braces `{}`

```python
my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}
```

Dictionary functions and methods

- `dict.keys()` - Returns a list of all keys in the dictionary.
- `dict.values()` - Returns a list of all values in the dictionary.
- `dict.items()` - Returns a list of tuples, where each tuple is a key-value pair.
- `dict.get(key)` - Returns the value associated with the key. If the key is not found, returns `None`.
- `dict.pop(key)` - Removes the key-value pair associated with the key and returns the value.
- `dict.popitem()` - Removes and returns a random key-value pair.
- `dict.update(dict2)` - Updates the dictionary with the key-value pairs from `dict2`.
- `dict.setdefault(key, default)` - Returns the value associated with the key. If the key is not found, assigns the default value.

Selecting dictionary elements

- `my_dict['name']` - Returns the value associated with the key 'name'.
- `my_dict.get('name', 'default')` - Returns the value associated with the key 'name'. If the key is not found, returns the default value.
- `len(my_dict)` - Returns the number of key-value pairs in the dictionary.

Getting started with characters and strings

A string is a sequence of characters enclosed in single or double quotes.

Creating strings

A string can be created using single or double quotes:

```python
my_string = 'Hello, World!
```

Combining and splitting strings

- `str.join(iterable)` - Combines elements of iterable into one string.
- `str.split(delimiter)` - Splits the string into substrings using the delimiter.
- `str.upper()` - Converts all characters to uppercase.
- `str.lower()` - Converts all characters to lowercase.
- `str.lstrip()` - Removes leading whitespace from the string.
- `str.rstrip()` - Removes trailing whitespace from the string.
- `str.center(width)` - Returns a centered copy of the string.
- `str.replace(old, new)` - Replaces all occurrences of `old` with `new`.
- `str.index()` - Returns the index of the first occurrence of `sub` in the string.

Mutate strings

- `str[0]` - Assigns a new character to the string.
- `str[0] = 'x'` - Changes the first character of the string.

Getting started with DataFrames

A DataFrame is a 2-dimensional data structure with columns of potentially different types.

Creating DataFrames

- `pd.DataFrame(data)`
- `pd.DataFrame(data, index, columns)`

Selecting DataFrames

- `df['column']` - Select a column.
- `df['column'][0:5]` - Select the first 5 rows of the column.
- `df.loc[0:5, 'column']` - Select the first 5 rows and the column.
- `df.iloc[0:5, 0:2]` - Select a range of rows and columns.

Manipulating DataFrames

- `df.rename(columns={'old': 'new'})` - Renames a column.
- `df.drop(['column'])` - Drops a column.
- `df.sort_values('column')` - Sorts the DataFrame by a column.
- `df.groupby('column').mean()` - Groups the DataFrame by a column and calculates the mean.
- `df.apply(func)` - Applies a function to each element in the DataFrame.
- `df.to_csv('filename.csv')` - Saves the DataFrame to a CSV file.

Operators

Arithmetic operators

- `+` - Addition
- `-` - Subtraction
- `*` - Multiplication
- `/` - Division
- `%` - Modulus
- `**` - Exponentiation

Assignment operators

- `=` - Assignment
- `+=` - Addition assignment
- `-=` - Subtraction assignment
- `*=` - Multiplication assignment
- `/=` - Division assignment
- `**=` - Exponentiation assignment

Logical operators

- `and` - Logical AND
- `or` - Logical OR
- `not` - Logical NOT

Comparisons

- `==` - Equal
- `!=` - Not equal
- `>` - Greater than
- `<` - Less than
- `>=` - Greater than or equal to
- `<=` - Less than or equal to

Math functions

- `abs(x)` - Absolute value
- `max(x)` - Maximum
- `min(x)` - Minimum
- `round(x)` - Round to the nearest integer
- `pow(x, y)` - Power
- `sqrt(x)` - Square root
- `log(x)` - Natural logarithm
- `exp(x)` - Exponential

Python packages

- `pip install pandas` - Installs the pandas package
- `import pandas` - Imports the pandas package

Data wrangling in pandas

- `df['column'].unique()` - Returns the unique values in a column
- `df['column'].value_counts()` - Counts the frequency of each value in a column
- `df['column'].isnull().sum()` - Counts the number of missing values in a column
- `df.fillna(value)` - Replaces missing values with a specified value

Calculations with NumPy

- `np.mean(arr)` - Calculates the mean of an array
- `np.std(arr)` - Calculates the standard deviation of an array
- `np.max(arr)` - Calculates the maximum of an array
- `np.min(arr)` - Calculates the minimum of an array
- `np.sum(arr)` - Calculates the sum of an array
- `np.prod(arr)` - Calculates the product of an array
- `np.sqrt(arr)` - Calculates the square root of an array
- `np.log(arr)` - Calculates the natural logarithm of an array
- `np.exp(arr)` - Calculates the exponential of an array
- `np.sin(arr)` - Calculates the sine of an array
- `np.cos(arr)` - Calculates the cosine of an array
- `np.tan(arr)` - Calculates the tangent of an array
- `np.arcsin(arr)` - Calculates the arcsine of an array
- `np.arccos(arr)` - Calculates the arccosine of an array
- `np.arctan(arr)` - Calculates the arctangent of an array
- `np.floor(arr)` - Rounds down to the nearest integer
- `np.ceil(arr)` - Rounds up to the nearest integer
- `np.rint(arr)` - Rounds to the nearest integer
- `np.trunc(arr)` - Truncates to the nearest integer