API Documentation

built with pdoc

# undocumented

▶ View Source

**students**: Dict[str, List[float]] = {'Max': [85, 92, 78], 'Marina': [90, 88], 'Ashley': [65, 74, 50], 'Bonnie': [99, 87]}

**def display_menu**() -> **None**:

▼ View Source

```python
def display_menu() -> None:
    """
    Display the main menu options.
    """
    print("-" * 30)
    print(
        """1. Add student
2. Add grade
3. Show average for student
4. Show all students
5. Save to file
6. Load from file
7. Exit"""
    )
    print("-" * 30)
```

Display the main menu options.

**def add_student**(students: Dict[str, List[float]], name: str) -> **None**:

▼ View Source

```python
def add_student(students: Dict[str, List[float]], name: str) -> None:
    """
    Add a new student to the students dictionary.
```

built with pdoc

```

    Parameters:
        students (dict): The dictionary of students.
        name (str): The name of the student to add.

    Returns:
        None
    """
    if name not in students:
        students[name] = []
    else:
        print(f"Student {name} already exists.")
```

Add a new student to the students dictionary.

Parameters: students (dict): The dictionary of students. name (str): The name of the student to add.

Returns: None

**def add_grade**(students: Dict[str, List[float]], name: str, grade: float) -> **None**: ▼ View Source

```
def add_grade(students: Dict[str, List[float]], name: str, grade: float) -> None:
    """
    Add a grade to a student's list of grades.

    Parameters:
        students (dict): The dictionary of students.
        name (str): The name of the student.
        grade (float): The grade to add (0-100).

    Returns:
        None
    """
    if name in students:
        if 0 <= grade <= 100:
            students[name].append(grade)
        else:
```

# API Documentation

built with pdoc

```python
            print("Grade must be between 0 and 100.")
    else:
        print(f"Student {name} does not exist.")
```

Add a grade to a student's list of grades.

Parameters: students (dict): The dictionary of students. name (str): The name of the student. grade (float): The grade to add (0-100).

Returns: None

**def average**(students: Dict[str, List[float]], name: str) -> float | **None**:

▼ View Source

```python
def average(students: Dict[str, List[float]], name: str) -> Optional[float]:
    """
    Calculate the average grade for a student.

    Parameters:
        students (dict): The dictionary of students.
        name (str): The name of the student.

    Returns:
        float or None: The average grade, or None if no grades exist.
    """
    if name in students and students[name]:
        return sum(students[name]) / len(students[name])
    else:
        print(f"{name} has no grades.")
        return None
```

Calculate the average grade for a student.

Parameters: students (dict): The dictionary of students. name (str): The name of the student.

Returns: float or None: The average grade, or None if no grades exist.

# API Documentation

built with pdoc

### def save_to_file(students: Dict[str, List[float]], filename: str) -> None:

▼ View Source

```python
def save_to_file(students: Dict[str, List[float]], filename: str) -> None:
    """
    Save the students dictionary to a file in JSON format.

    Parameters:
        students (dict): The dictionary of students.
        filename (str): The name of the file to save to.

    Returns:
        None
    """
    with open(filename, "w") as file:
        json.dump(students, file)
```

Save the students dictionary to a file in JSON format.

Parameters: students (dict): The dictionary of students. filename (str): The name of the file to save to.

Returns: None

### def load_from_file(filename: str) -> Dict[str, List[float]]:

▼ View Source

```python
def load_from_file(filename: str) -> Dict[str, List[float]]:
    """
    Load the students dictionary from a file in JSON format.

    Parameters:
        filename (str): The name of the file to load from.

    Returns:
        dict: The dictionary of students loaded from the file, or an empty dictio
    """
    if not os.path.exists(filename):
```

# API Documentation

```
            print("File does not exist.")
            return {}

        with open(filename, "r") as file:
            return json.load(file)
```

Load the students dictionary from a file in JSON format.

Parameters: filename (str): The name of the file to load from.

Returns: dict: The dictionary of students loaded from the file, or an empty dictionary if the file does not exist.

**def delete_student**(students: Dict[str, List[float]], name: str) -> **None**:

▼ View Source

```
def delete_student(students: Dict[str, List[float]], name: str) -> None:
    """
    Delete a student from the students dictionary.

    Parameters:
        students (dict): The dictionary of students.
        name (str): The name of the student to delete.

    Returns:
        None
    """
    if name in students:
        del students[name]
        print(f"Deleted student {name}.")
    else:
        print(f"Student {name} does not exist.")
```

Delete a student from the students dictionary.

Parameters: students (dict): The dictionary of students. name (str): The name of the student to delete.

# API Documentation

Returns: None

**def delete_grade**(students: Dict[str, List[float]], name: str, grade: float) -> **None**:

▼ View Source

```
def delete_grade(students: Dict[str, List[float]], name: str, grade: float) -> No
    """
    Delete a grade from a student's list of grades.

    Parameters:
        students (dict): The dictionary of students.
        name (str): The name of the student.
        grade (float): The grade to delete.

    Returns:
        None
    """
    if name not in students:
        print(f"Student {name} does not exist.")
        return
    if grade not in students[name]:
        print(f"{name} does not have grade {grade}.")
        return
    students[name].remove(grade)
    print(f"Removed grade {grade} from {name}.")
```

Delete a grade from a student's list of grades.

Parameters: students (dict): The dictionary of students. name (str): The name of the student. grade (float): The grade to delete.

Returns: None