Python Temperature Conversion Functions

Robotik AG

March 6, 2024

Funktionen

Erklärung

Die Struktur einer Python-Funktion mit Docstring und Doctest sieht folgendermaßen aus:

```
def function_name(parameter: parameter_type) -> return_type:
    """
    Description of the function.

    Args:
        parameter (parameter_type): Description of the parameter.

    Returns:
        return_type: Description of the return value.

    Raises:
        ExceptionType: Description of the exception (if any).

    Examples:
        >>> function_name(example_value)
        expected_result
    """
    # Implementation of the function
    # ...
    pass
```

Hier sind die Erklärungen zu den verschiedenen Teilen:

- **function_name:** Der Name der Funktion.
- **parameter:** Der Eingabeparameter der Funktion.
- **parameter_type:** Der Datentyp des Eingabeparameters.
- **return_type:** Der Datentyp des Rückgabewerts.

- **Description of the function:** Eine kurze Beschreibung der Funktion und ihrer Aufgabe.
- **Description of the parameter:** Eine kurze Beschreibung des Eingabeparameters.
- **Description of the return value:** Eine kurze Beschreibung des Rückgabewerts.
- **Raises:** Eine optionale Sektion, die Ausnahmen dokumentiert, die die Funktion auslösen kann.
- **ExceptionType:** Der Typ der Ausnahme, wenn eine auftritt.
- **Examples:** Beispiele, wie die Funktion verwendet werden kann, zusammen mit den erwarteten Ergebnissen.

Doctests: Verwendung und Bedeutung

- **Doctests** sind spezielle Kommentare in Python-Docstrings, die dazu dienen, Beispiele für die Verwendung der Funktion zu dokumentieren.
- **Vorteile:** Doctests ermöglichen es, die Funktionen automatisch zu testen und sicherzustellen, dass sie wie erwartet funktionieren.
- **Beispiel:** In den `Examples`-Sektionen sind Beispiele gegeben, wie die Funktion aufgerufen wird und welches Ergebnis erwartet wird.
- **Ausführen:** Doctests können durch Ausführen des Python-Moduls mit dem Parameter `python -m doctest pythonfile.py` gestartet werden.

Umrechnungsformeln

Dies kann dir sicher helfen.
Celsius = 5/9 * (Fahrenheit - 32).
Celsius = Kelvin - 273.15.
Die tiefste mögliche Temperatur ist den absoluten Nullpunkt 0K.

Aufgabe 1

Bitte vervollständige die Funktionen `celsius_to_fahrenheit`, `fahrenheit_to_celsius`, `celsius_to_kelvin`, `kelvin_to_celsius`, `fahrenheit_to_kelvin` und `kelvin_to_fahrenheit`. Du kannst sie in der Python-Datei `Temperaturen.py` speichern.

Aufgabe 2

Schreibe ein Funktion `main`, die den Benutzer nach eingabe Temperatur fragt und sie dann in die gewünschte Temperatur umrechnet.

1 Schleifen

Aufgabe 1

Vervollständige die Funktion `weihnachtsbaum` in `Schleifen.py` so, dass sie einen Weihnachtsbaum mit der gewünschten Anzahl von Ebenen und Sternen zeichnet.
Beispiel:

```
>>weihnachtsbaum(3):
   *
  ***
*****
```

Aufgabe 2

Schreibe jeweils eine Funktion, die folgende Aufgaben erfüllt:

a)

Berechne die Gesamtarbeitsstunden für die Woche.

b)

Überprüfe, an welchem Tag die meisten Stunden gearbeitet wurden.

c)

Überprüfe, ob der Schüler an jedem Tag mindestens 6 Stunden gearbeitet hat. Wenn ja, gib eine Nachricht aus, dass die Woche erfolgreich war, sonst eine Nachricht des Bedauerns.

d)

Erstelle ein neues Dictionary, das die Arbeitsstunden pro Tag in Minuten enthält, wobei angenommen wird, dass 1 Stunde gleich 60 Minuten ist.

e)

Füge einen neuen Arbeitstag hinzu und weise ihm 8 Stunden zu.