Einführung in die Programmierung

Prof. Dr. Franziska Matthäus
Prof. Dr. Matthias Kaschube
Dr. Karsten Tolle
Kenan Khauto

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Übungsblatt ÜE-03

Ausgabe: 06.11.2024
Abgabe: 16.11.2024
13:00 Uhr

Module und Funktionen

Hinweis:

- *Es dürfen keine Lösungen aus dem Skript, dem Internet oder anderen Quellen abgeschrieben werden. Diese Quellen dürfen nur mit Quellenangaben verwendet werden und es muss ein hinreichend großer Eigenanteil in den Lösungen deutlich zu erkennen sein.*
- *Digitale Abgaben, die nicht im Format* ***.pdf*** *oder* ***.txt*** *für Texte oder* ***.py*** *für Code erfolgen, werden nicht bewertet. Bei Abgaben mehrerer Dateien müssen diese als* ***.zip*** *zusammengefasst werden.*
- *Achten Sie darauf die Variable* ***__author__*** *in allen Quellcode Dateien (.py) korrekt zu setzen (am Anfang des Quellcodes):* __author__ = "<Matr-Nr>, <Nachname>"
  Beispiel: __author__ = "1234567, Tolle"
  … Leerstellen vor und nach dem „=" und Leerstelle nach dem Komma beachten, sowie keine spitzen Klammern verwenden, die dienen bei der Schreibweise oben zum Definieren von Variablen.
- *Außerdem muss Ihr Name in jeder abgegebenen* ***.pdf*** *und* ***.txt*** *Datei zu finden sein.*
- *Abgaben der Dokumentation, die per Hand geschrieben und eingescannt werden, sind nur in zuvor abgesprochenen Ausnahmefällen erlaubt.*

Aktualisierung der Hinweise:

- *Datei- und Ordnernamen sollen keine Umlaute, diakritische oder Sonderzeichen, mit Ausnahme des Unterstrichs in der Namesmitte, enthalten! (Insb. MAC Nutzer sollten aufpassen, dass nicht zusätzliche Dateien im ZIP-Ordner hinzugefügt wurden! … diese ggf. löschen!)*
- *Beim Programmieren und Kommentieren halten Sie sich an die Regeln im Programmierhandbuch, siehe Moodle-Kurs (Programmierhandbuch WiSe 24/25 (Style Guide)). Im Zweifelsfall gilt PEP 8.*
- *Bitte auch darauf achte, dass Ihre .py Dateien lauffähig sind. Es ist den Tutoren nicht zuzumuten Abgaben zu Debuggen. Probleme, auch was vielleicht nicht geht, kann auskommentiert und mit Kommentaren versehen werden.*

∑ 10 Punkte

Aufgabe ÜE-03-1: Turtle Flower Drawing (4 Punkte)

In dieser Aufgabe soll ein kreatives Muster gezeichnet werden, indem mit der **turtle-Bibliothek** gearbeitet wird (Intro siehe z.B. hier: https://realpython.com/beginners-guide-python-turtle/). Es sollen Funktionen erstellt werden, die eine **spiralförmige Blume** mit Turtle zeichnen, indem Kreise in einem bestimmten Winkel gedreht und ihr Radius iterativ vergrößert wird.

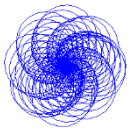
Ziel der Aufgabe
Erstelle ein Programm, das mithilfe von Turtle eine blumenähnliche Form zeichnet, indem Kreise in einer Spirale angeordnet werden. Die Blume besteht aus mehreren "Blütenblättern", die jeweils eine eigene Spirale aus Kreisen darstellen.

Schritte zur Lösung der Aufgabe

Erstelle ein Python-Modul namens spiral_flower.py. In diesem Modul sollen die folgenden Funktionen definiert werden:

- **draw_circle_spiral(radius, angle, step, num_circles)**: Diese Funktion zeichnet eine Spirale aus Kreisen.
    - **Parameter**:
        - radius: Start-Radius für den ersten Kreis in der Spirale.
        - angle: Drehwinkel, um den Turtle sich nach jedem Kreis dreht.
        - step: Die Schrittweite, um den Radius bei jedem Kreis zu vergrößern.
        - num_circles: Anzahl der Kreise in der Spirale.
    - **Hinweis**: Verwende eine Schleife, um die Kreise zu zeichnen und den Radius iterativ zu vergrößern.
- **draw_flower(petals, radius, angle, step)**: Diese Funktion zeichnet eine Blume, die aus mehreren spiralförmigen Blütenblättern besteht.
    - **Parameter**:
        - petals: Anzahl der Blütenblätter (jede Blütenblatt ist eine eigene Spiralform).
        - radius: Start-Radius für die Kreise in jedem Blütenblatt.
        - angle: Winkel, um den Turtle sich innerhalb jedes Blütenblatts dreht.
        - step: Schrittweite für die Spirale innerhalb jedes Blütenblatts.
    - **Hinweis**: Verwende die Funktion draw_circle_spiral() innerhalb dieser Funktion und rotiere Turtle nach jedem Blütenblatt, um sie gleichmäßig um das Zentrum anzuordnen.

Füge in spiral_flower.py unterhalb der Funktionen im Block ***if __name__ == "__main__"*** Testfälle hinzu, um die Funktionen ***draw_circle_spiral()*** und ***draw_flower()*** mit unterschiedlichen Parametern zu testen.
Verwende **turtle.clear()** nach jedem Testfall, um das Zeichenfenster zu leeren und das nächste Muster anzuzeigen.
Beispiel mit ***draw_flower(petals=10, radius=15, angle=10, step=4)***



Tipps und Hinweise

1. **Schrittweise Entwicklung**: Erstellen und teste Sie zuerst ***draw_circle_spiral***.
2. **Parameter anpassen**: Experimentieren Sie mit verschiedenen Werten für **angle, radius, step,** und **petals,** um verschiedene Muster zu erzeugen.
3. **Turtle-Attribute**: Sie können die Turtle-Geschwindigkeit mit ***turtle.speed(0)*** auf das Maximum setzen und Farben mit ***turtle.color()*** anpassen.
4. **Rotation in draw_flower**: Damit die Blütenblätter gleichmäßig angeordnet werden, drehen Sie die Turtle nach jedem Blütenblatt um *360 / petals Grad*.
5. **Dokumentation und Docstrings nicht vergessen!**

Aufgabe ÜE-03-2: Einfache Ver- und Entschlüsselung (6 Punkte)

In dieser Aufgabe soll eine einfache Verschlüsselungs- und Entschlüsselungsanwendung in Python erstellt werden. Das Ziel ist es, den Benutzer eine Nachricht eingeben zu lassen und die Nachricht dann so zu verschlüsseln, dass die Buchstaben verschoben werden (eine einfache Caesar-Verschlüsselung). Eine weitere Funktion soll eine verschlüsselte Nachricht wieder entschlüsseln.

In der **Caesar-Verschlüsselung** wird jeder Buchstabe des Klartexts um eine feste Anzahl von Positionen im Alphabet verschoben. Die Formel zur Berechnung des verschlüsselten Buchstabens basiert auf den ASCII-Werten der Buchstaben.
Für einen Buchstaben char mit einem Shift-Wert *shift* lautet die Formel:

encrypted_char = chr((ord(char) − shift_base + shift) % 26 + shift_base)

Erklärung der Formel

- ***ord(char):*** Diese Funktion gibt den ASCII-Wert des Zeichens *char* zurück.
- ***shift_base***: Der Startwert für Groß- oder Kleinbuchstaben im ASCII-Code:
    - Für Großbuchstaben ist shift_base = ord('A'), also 65.
    - Für Kleinbuchstaben ist shift_base = ord('a'), also 97. Dadurch wird die Verschiebung innerhalb des Alphabets (26 Buchstaben) korrekt berechnet.
- ***shift***: Die Anzahl der Positionen, um die jeder Buchstabe verschoben wird.
- ***% 26***: Das Modulo 26 sorgt dafür, dass das Ergebnis innerhalb des Alphabets bleibt. Wenn ***ord(char) - shift_base + shift*** größer als 25 ist (z. B. Z + 3), beginnt es wieder von vorne bei A.
- ***chr(...):*** Wandelt den berechneten ASCII-Wert wieder in einen Buchstaben um.

Aufgabenbeschreibung

1. Erstellen Sie das Modul **message_cipher.py**: In diesem Modul sollen die Funktionen **encrypt_message** und **decrypt_message** definiert werden. **(4 Punkte)**
    - ***encrypt_message(message, shift):*** Diese Funktion verschlüsselt eine Nachricht, indem sie jeden Buchstaben im Text um eine bestimmte Anzahl von Positionen (Shift) im Alphabet verschiebt und gibt sie zurück.
    - ***decrypt_message(encrypted_message, shift):*** Diese Funktion entschlüsselt die verschlüsselte Nachricht, indem sie die Verschiebung umkehrt und gibt sie zurück.

2. Erstellen Sie ein Hauptprogramm in einer anderen Datei namens ***main.py***: In diesem Programm soll das Modul **message_cipher.py** genutzt werden. ***main.py*** sollte: **(2 Punkte)**
    - Den Benutzer fragen, ob er eine Nachricht verschlüsseln oder entschlüsseln möchte.

Wichtige Hinweise:

I. Module richtig verwenden: Speichere die Funktionen ***encrypt_message*** und ***decrypt_message*** im Modul ***message_cipher.py*** und importiere sie in ***main.py***.
II. Groß- und Kleinbuchstaben: Achte darauf, dass Groß- und Kleinbuchstaben korrekt behandelt werden. Nicht-Buchstaben-Zeichen wie Leerzeichen oder Satzzeichen sollten unverändert bleiben.
III. ASCII-Werte: Verwende die Funktionen ***ord()*** und ***chr()***, um ASCII-Werte zu manipulieren.
IV. Docstrings für die Funktionen nicht vergessen!
V. Testfälle für jede Funktion unter ***if __name__ == "__main__"*** in **message_cipher.py**.

Für diese Aufgabe ist keine Dokumentation nötig.