Technische Hochschule Ingolstadt

Fakultät Informatik

Programmierung 1

Praktikumsaufgabe 2: Ufo-Autopilot

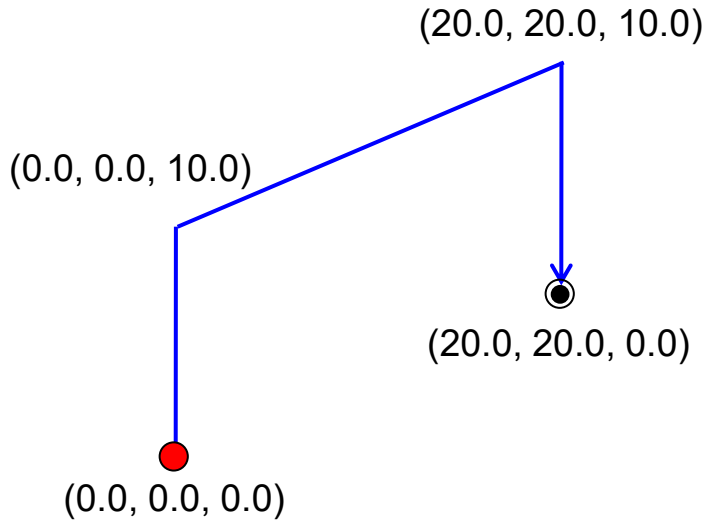
Prof. Dr. Robert Gold | Wintersemester 2025/26

Praktikum

- In den folgenden Praktikumsaufgaben erstellen wir die autonome Steuerung (Autopilot) einer Lieferdrohne, manchmal auch Ufo genannt. Dazu gehören u.a.
  - autonome Steuerung mit Steuerbefehlen
  - Routenplanung: Es sollen mehrere Ziele angesteuert werden
  - Protokollierung der Flugstrecke
  - Einlesen von Konfigurationsdaten
- Die Drohne wird simuliert. Lesen Sie zum Verständnis der Simulation bitte das Ufo-Handbuch.

Praktikum

- In diesem Teil des Praktikums bringen wir das Ufo zum Fliegen.
- Es soll von (0.0, 0.0, 0.0) nach (x, y, 0.0) fliegen. Die Flughöhe ist z.
  - Die Abbildung zeigt x = 20.0, y = 20.0, z = 10.0.
- Das Ufo startet und landet immer auf Höhe 0.0.
- Testen Sie das fertige Programm mit
  - x = 20.0, y = 20.0, z = 10.0
  - x = -100.0, y = 20.5, z = 10.0
  - x = -1.0, y = -1.0, z = 100.0
  - x = 0.0, y = 40.0, z = 8.95



- Gehen Sie bei der Programmierung schrittweise vor, wie auf den folgenden Folien gezeigt.

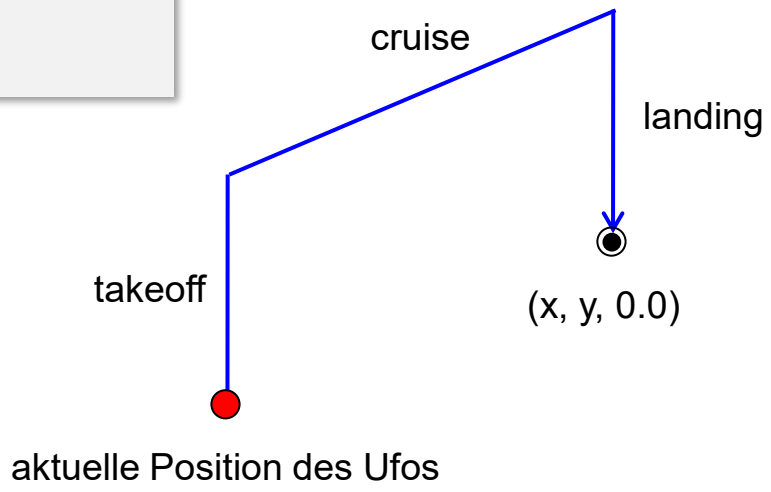
1. Verwenden Sie die vorgegebene Datei ufo_autopilot.py.
2. Darin sind schon die folgenden vier Funktionen enthalten. Sie müssen nicht geändert werden!

```
def fly_to(sim, x: float, y: float, z: float) -> None:
```

Parameter: sim : Referenz auf die Simulation
x, y : Punkt im kartesischen Koordinatensystem
z : Flughöhe

Rückgabewert: Keiner

Diese Funktion fliegt das Ufo von der aktuellen Position des Ufos ins Ziel und ruft lediglich die drei Funktionen takeoff, cruise und landing (siehe folgende Folie) nacheinander auf. Alle drei Funktionen haben keinen Rückgabewert.



def takeoff(sim, z: float) -> None:

Parameter: sim : Referenz auf die Simulation
z : Flughöhe

Rückgabewert: Keiner

def cruise(sim, x: float, y: float) -> None:

Parameter: sim : Referenz auf die Simulation
x, y : Punkt im kartesischen Koordinatensystem

Rückgabewert: Keiner

def landing(sim) -> None:

Parameter: sim : Referenz auf die Simulation

Rückgabewert: Keiner

3. Implementieren Sie in der Datei ufo_autopilot.py die folgenden Funktionen:

a)

```
def distance(x1: float, y1: float, x2: float, y2: float) -> float:
```

Parameter: x1, y1, x2, y2 : zwei Punkte im kartesischen Koordinatensystem

Rückgabewert: Abstand zwischen (x1, y1) und (x2, y2)

Diese Funktion wird benötigt, um rechtzeitig vor dem Erreichen des Ziels abzubremsen.

b)

```
def angle_q1(x1: float, y1: float, x2: float, y2: float) -> float:
```

Parameter: x1, y1, x2, y2 : zwei Punkte im kartesischen Koordinatensystem mit $x2 \geq x1$ und $y2 \geq y1$

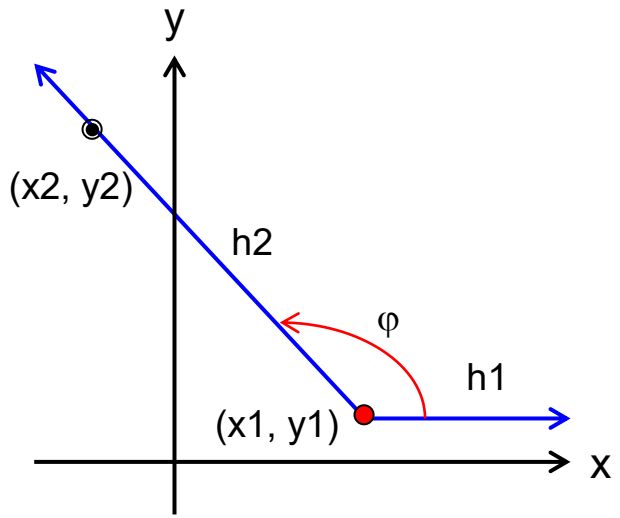
Rückgabewert: Winkel φ in Grad, $0° \leq \varphi < 90°$ (siehe Praktikumsaufgabe 1)

Das Skript aus Praktikumsaufgabe 1 wird kopiert und als Funktion geschrieben. Die input-Anweisungen entfallen. Aus print-Anweisungen werden return-Anweisungen.

c) def angle(x1: float, y1: float, x2: float, y2: float) -> float:

Parameter: x1, y1, x2, y2 : zwei Punkte im kartesischen Koordinatensystem

Rückgabewert: Winkel φ in Grad, $0° \leq \varphi < 360°$

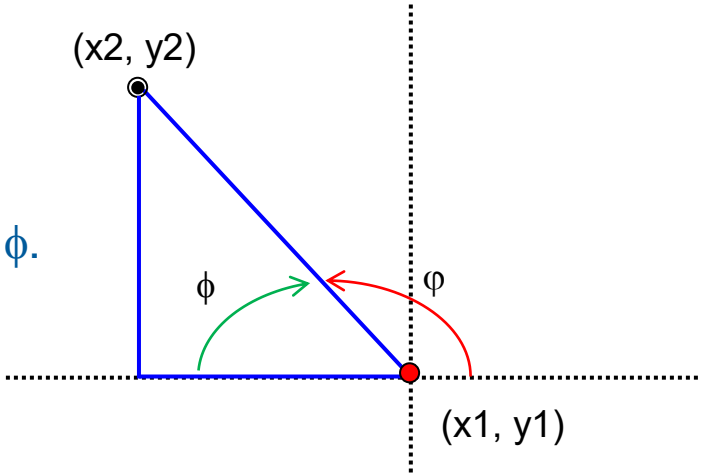


Mit dieser Funktion wird der Drehwinkel des Ufos bestimmt.
x1, y1, x2, y2 können jetzt aber beliebig sein!

Je nachdem in welchem Quadranten (von (x1, y1) aus gesehen) (x2, y2) liegt, gibt es vier Fälle. Die Abbildungen zeigen den Fall, dass (x2, y2) im 2. Quadranten liegt. Die Funktion angle_q1(–x1, y1, –x2, y2) berechnet den Winkel ϕ. Dann ist $\varphi = 180 - \phi$.

In der Funktion angle soll die Funktion angle_q1 aufgerufen werden.

Analog für die anderen Quadranten.



d) def flight_distance(x1: float, y1: float, x2: float, y2: float, z: float) -> float:

Parameter: x1, y1, x2, y2 : zwei Punkte im kartesischen Koordinatensystem
z : Flughöhe

Rückgabewert: zu fliegende Strecke (Summe von drei Teilstrecken takeoff, cruise, landing)

Diese Funktion braucht man, um die tatsächlich geflogene Strecke mit der berechneten Strecke zu vergleichen. Verwenden Sie die Funktion distance.

e) def format_flight_data(sim) -> str:

Parameter: sim : Referenz auf die Simulation

Rückgabewert: formatierte Zeichenkette mit Flugzeit, x-, y-, z-Koordinate des Ufos

Auf die Daten kann mit sim.get_ftime(), sim.get_x(), sim.get_y(), sim.get_z() zugegriffen werden.
Beispiel eines Rückgabewerts: `"  8.5 s:   10.2   -3.1  10.0 "`
Beachten Sie die Formatierung. Alle Zahlen sollen eine Nachkommastelle haben. Die Flugzeit und die z-Koordinate sollen 4 Zeichen, die x- und y-Koordinaten 5 Zeichen haben. Auf die Typannotation von sim kann verzichtet werden.

f)

```
def fac(m: int =1, n: int =1) -> int:
```

Parameter: n, m mit n, m > 0

Rückgabewert: Produkt n · (n+1) · (n+2) · … · m

Berechnen Sie das Produkt mit der Rekursionsformel fac(m , n) = n * fac(m - 1, n + 1) * m, d.h. in jedem Rekursionsschritt werden zwei Multiplikationen ausgeführt. Die Funktion muss rekursiv sein und darf insbesondere keine Schleife enthalten. Die Funktion factorial darf nicht verwendet werden. Wenn m < n, soll das Ergebnis 1 sein.

Beispiele: fac(5, 3) = 3 · 4 · 5 = 60, fac(4, 3) = 12, fac(3, 3) = 3, fac(2, 3) = 1, fac(4) = 24.

Diese Funktion wird in dieser Praktikumsaufgabe erstellt, aber erst in einer späteren Praktikumsaufgabe verwendet.

4. Das Hauptprogramm ufo_main.py ist vorgegeben (siehe Vorlage für ufo_main.py auf Moodle). Ergänzen Sie das Hauptprogramm an den gekennzeichneten Stellen

   i. um die Konsoleingabe des Ziels x, y und der Flughöhe z
   (Die Flughöhe sollte > 0 sein. Braucht aber nicht überprüft zu werden.)

   ii. um eine Konsolausgabe der zu fliegenden Distanz (mit zwei Nachkommastellen)

   ```
   flight_distance(0.0, 0.0, x, y, z)
   ```

   iii. um eine Konsolausgabe der tatsächlich geflogenen Distanz (mit zwei Nachkommastellen)

   ```
   sim.get_dist()
   ```

   Wenn Sie nach (–100.0, 20.0) fliegen, werden Sie feststellen, dass das Ufo ein Stück neben dem Zielpunkt landet. Das ist nicht gut. Falls das Ufo z.B. auf einer Straße landet, wird es überfahren und die zu liefernde Pizza schmeckt nicht mehr.

   Warum ist das so? Wie kann man das ändern?

Praktikum

Tipp zur Bearbeitung

- Sie können bei der Bearbeitung folgendermaßen vorgehen:
  - Herunterladen von ufo_main.py, ufo_autopilot.py, pa2_utest.py von Moodle
  - Kopieren der Kopfzeilen der Funktionen aus diesem Dokument in ufo_autopilot.py
  - Ergänzen der Funktionen durch einen Dummy-Rumpf return 0.0 oder return 0 oder return ""
  - Ausführen von pa2_utest.py (am Anfang schlagen die meisten Testfälle fehl)
  - Schrittweises Ausprogrammieren der Funktionen und Ausführen von pa2_utest.py bis alle Testfälle erfolgreich sind
  - Ergänzen von ufo_main.py
  - Testen von ufo_main.py mit einigen selbst gewählten Eingaben

Praktikum

Hinweise

- Verpacken Sie die beiden py-Dateien ufo_autopilot.py und ufo_main.py in eine zip-Datei.
  - Andere Komprimierungsformate als zip (z.B. 7z, rar, …) sind nicht erlaubt.
  - Achten Sie darauf, dass die beiden Dateien genauso heißen.
- Laden Sie die zip-Datei anschließend rechtzeitig in Moodle hoch.
- Die Lösungen müssen nur bei offenen Fragen im Praktikum vorgeführt werden.
- Vor der Abgabe stellen Sie bitte folgendes sicher:
  - Die Funktionen haben genau die in der Aufgabestellung spezifizierten Parameter und Rückgabewerte, keine anderen und keine zusätzlichen.
  - Die py-Datei ufo_autopilot.py enthält die Funktionen distance, angle_q1, angle, flight_distance, format_flight_data, fac, fly_to, takeoff, cruise, landing. Außer diesen Funktionen und benötigten Import-Anweisungen ist kein weiterer Code enthalten.

(weiter auf der nächsten Folie)

Praktikum

Hinweise

- Die py-Datei ufo_main.py enthält das Hauptprogramm. Außer dem Hauptprogramm und benötigten Import-Anweisungen ist kein weiterer Code enthalten.
- Die Parameter (außer dem Parameter sim) und die Rückgabewerte aller Funktionen des Programms haben Typannotationen. Die Überprüfung mit mypy liefert keine Fehler.
- In allen Skripten sind die Regeln für Variablennamen eingehalten.
- Alle Skripte haben keine Syntaxfehler.
- Das Skript pa2_utest.py enthält einen groben Funktionstest. Das Testskript muss sich im selben Verzeichnis wie ufo_autopilot.py und ufo_main.py befinden. Das Testskript muss fehlerfrei gelaufen sein.
- Die Praktikumsaufgaben sind von Ihnen selbstständig gelöst worden.