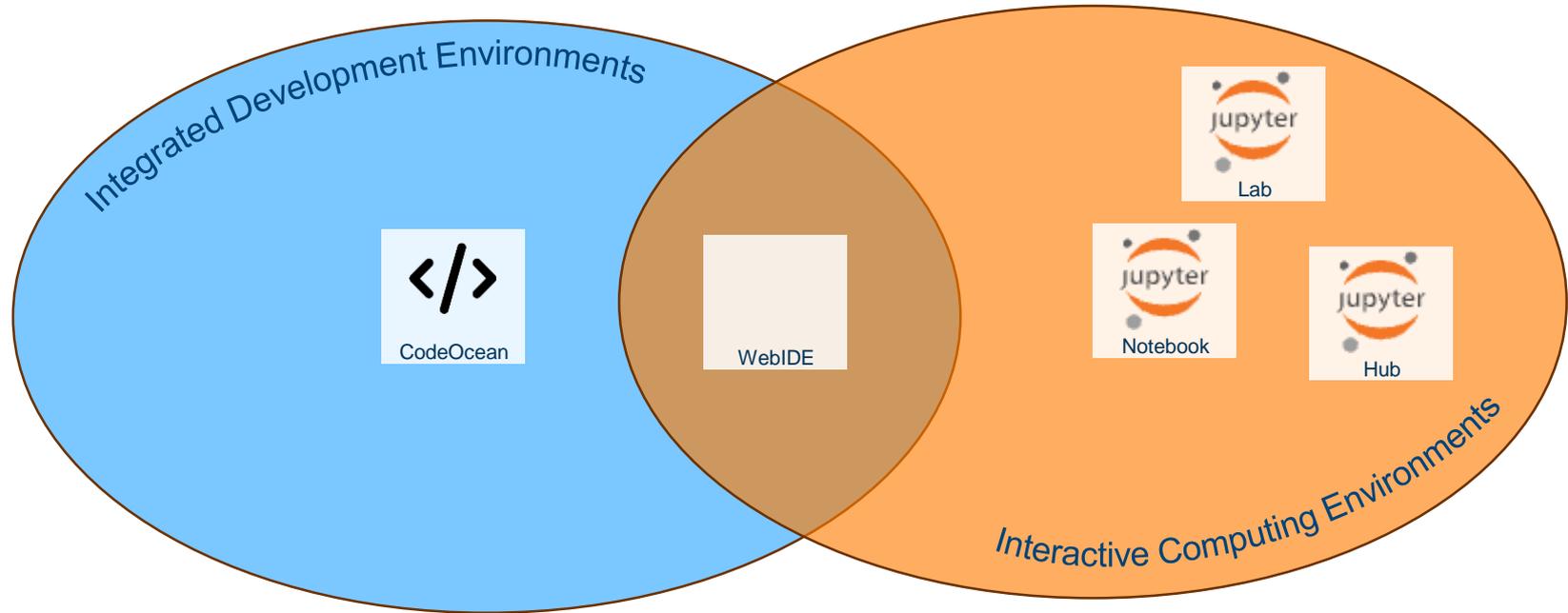


Tag der Lehre 2024

Technologie-Unterstützung in der Programmierausbildung



(Lern-)Werkzeuge

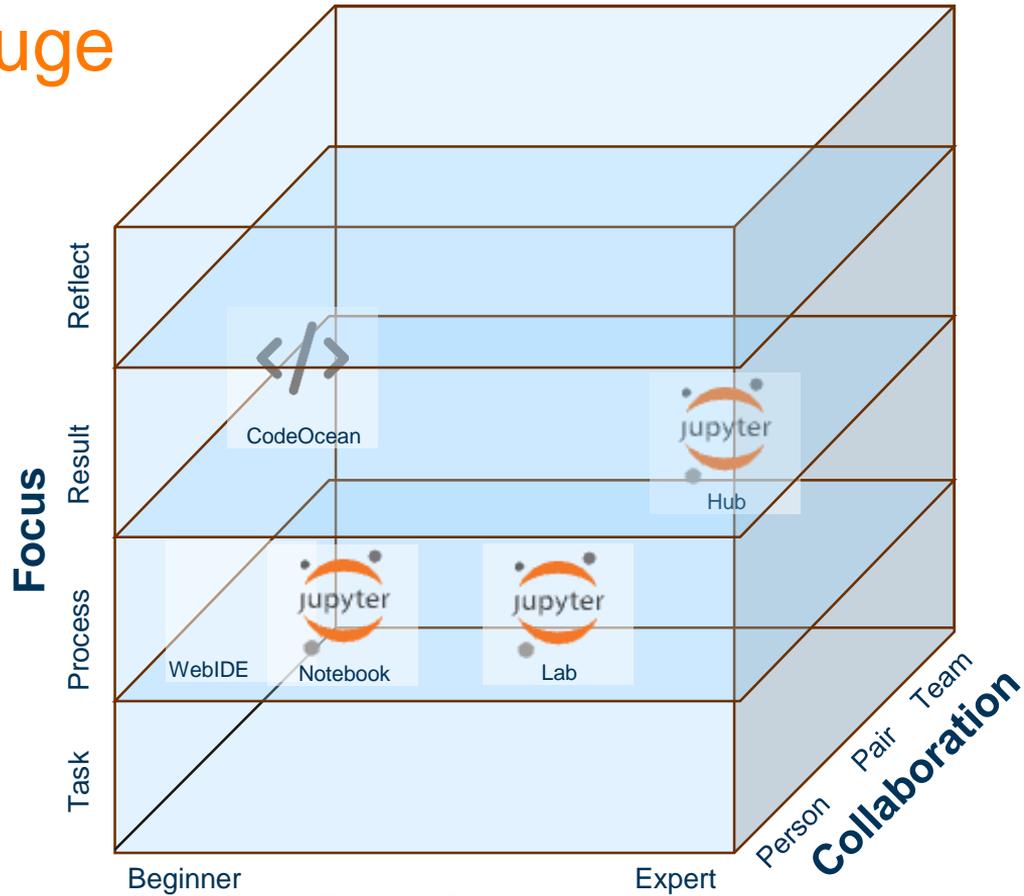


LERN-Werkzeuge

Usability?

Adaptability?

Maintainability?



Tag der Lehre 2024

Aufgabenorientierter Einstieg in die Programmierausbildung mit CodeOcean

examING-Teilprojekt: Autograder in der Programmierausbildung für INGenieure (AutoPING)

The screenshot displays the CodeOcean autograder interface. On the left, a Python code editor shows the implementation of a number-guessing game. The code includes a `guess_number()` function that prompts the user for a guess and a `compare_numbers()` function that checks the guess against a correct number. On the right, the test results panel shows a score of 5 out of 5, indicating that all tests passed. The panel also displays feedback messages and error messages for the tests.

```
1 from random import randint
2
3
4 def guess_number() -> int:
5     try:
6         number = int(input("Ihr Tipp bitte: "))
7         if number < 1 or number > 100:
8             raise ValueError
9         return number
10    except ValueError:
11        print("Fehlerhafte Eingabe")
12        return 0
13
14
15 def compare_numbers(correct_number: int, guessed_number: int) -> bool:
16     if correct_number == guessed_number:
17         print("Richtig - Sehr gut!")
18         return True
19     if guessed_number > correct_number:
20         if guessed_number - correct_number > 20:
21             print("Zahl viel zu groß.")
22         else:
23             print("Zahl zu groß.")
24         return False
25
26     if correct_number - guessed_number > 20:
27         print("Zahl viel zu niedrig.")
28     else:
29         print("Zahl zu niedrig.")
```

Test Results:

- Score:** 5 out of 5
- Feedback:** Well done. All tests have been passed.
- Error Messages:**

Test File 3 (test_main.py)

- Passed Tests:** 1 out of 1
- Score:** 5 out of 5
- Feedback:** Well done. All tests have been passed.
- Error Messages:**

Lint Feedback (test_style.py)

- Code Rating:** 10 out of 10



Problem

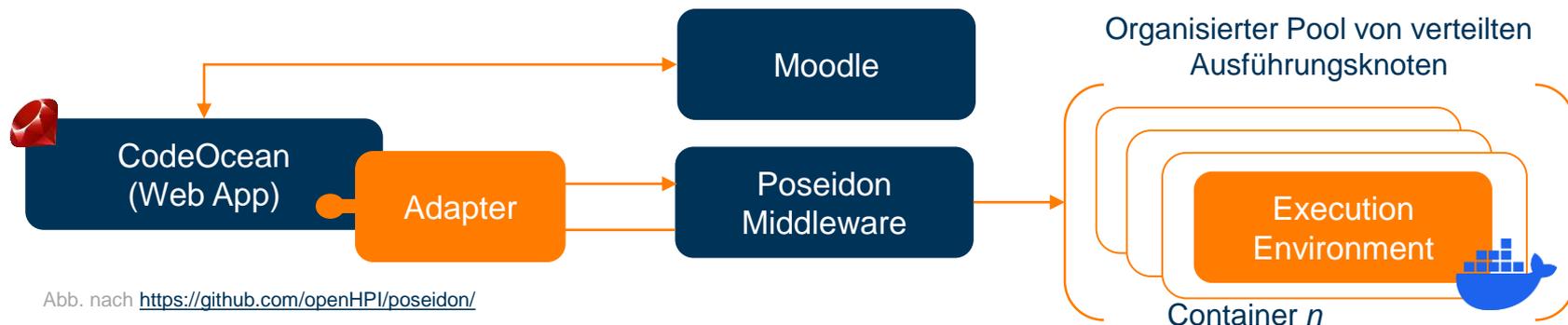
Kompetenzorientiertes Lernen ermöglichen



Lösung

CodeOcean Autograding-System

1. im Produktivbetrieb bewährt
2. webbasierte Lösung mit LTI-Schnittstelle für Moodle-Integration
3. Protokollierungs-, Statistik-, Kollaborationsfunktionen
4. verspricht leichte Adaptierbarkeit
5. aktives Entwickler-Team



CodeOcean für Lernende

Aufgaben-Ansicht

The screenshot shows the CodeOcean interface for a task titled "WIDIG01_Guess-a-Number". The top navigation bar includes the CodeOcean logo, "Administration", and user information for "Jonas Genath (Admin)". The breadcrumb trail is "EXERCISES / WIDIG01_GUESS-A-NUMBER / IMPLEMENT".

WIDIG01_Guess-a-Number 0%

Implementieren Sie das Spiel "Zahlenraten". In dem zu entwickelnden Python-Programm generiert der Computer eine Zufallszahl zwischen 1 und 100 (beide enthalten). Anschließend versucht die Spielerin, die generierte Zahl zu erraten. Hierzu gibt die Spielerin so lange Zahlen ein, bis sie die richtige Zahl gefunden hat.

Nutzen Sie bitte die vorgegebenen Programmteile und überarbeiten Sie diese. Weitere Hinweise finden Sie an den durch '# ToDo' im Quellcode marklierten Stellen. Bitte ändern Sie keine bereits festgelegten Funktionsnamen. Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.

Folgende Anforderungen soll das Programm erfüllen:

- 1.1 Bitte verwenden Sie für jede ausgegebene Zeile, die Text enthält, einen separaten Print-Befehl. Print-Befehle, die nur einen oder mehrere Zeilenumbrüche (wie z. B. `print('\n')` oder `print(, '\n')` enthalten, sind untersagt.
- 1.2 Das Programm wählt eine Zufallszahl zwischen 1 und 100 (beide enthalten) aus.
- 1.3 Das Programm lässt die Spielerin über die Standardeingabe so lange ihre Tipps eingeben, bis diese die richtige Zahl errät.
- 1.4 Das Programm prüft nach jeder Eingabe, ob die richtige Zahl erraten wurde.
- 1.5 Wenn die eingegebene Zahl zu niedrig ist, soll "Zahl zu niedrig" und bei mehr als 20 zu niedrig "Zahl viel zu niedrig" über die Standardausgabe ausgegeben werden.
- 1.6 Ist die eingegebene Zahl zu hoch, soll "Zahl zu groß" und bei mehr als 20 zu hoch "Zahl viel zu groß" über die Standardausgabe ausgegeben werden.
- 1.7 Wenn die Zahl richtig ist, soll "Richtig - Sehr gut!" über die Standardausgabe ausgegeben und das Spiel beendet werden.
- 1.8 Bei fehlerhaften Eingaben (keine Zahl oder keine Zahl von 1 bis 100) soll "Fehlerhafte Eingabe" über die Standardausgabe ausgegeben und das Raten einfach fortgesetzt werden.

[\(Hide\)](#)

Collapse Action Sidebar

Files

- solution01.py
- task01.py**
- test_util.py

Run Score Request Comments

```
1 # ToDo:
2 # Überprüfen Sie den gesamten Code, auch die bereits vorgegebenen Programmteile.
3 # Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.
4
5 from random import randint
6
7
8 def main():
9     correct_number = randint(1, 100)
10    print(
11        "Willkommen! Ich habe mir eine Zahl zwischen 1 und 100 ausgedacht, die Sie nun erraten können."
12    )
13
14    # ToDo:
15    # Implementieren Sie hier Code, der den Spieler solange um die Eingabe einer Zahl bittet,
16    # bis dieser die Zahl richtig erraten hat. Nutzen Sie für die Eingabe die Methode "guess_number" und
```

Collapse Output Sidebar

Results

4 test files have been executed.

Test File 1 (test_compare_numbers.py)

Passed Tests	0 out of 1
Score	0 out of 5



CodeOcean für Lernende

Aufgaben-Ansicht

CodeOcean Administration | English | Help | Jonas Genath (Admin)

EXERCISES / WIDIG01_GUESS-A-NUMBER / IMPLEMENT

WIDIG01_Guess-a-Number 0%

Implementieren Sie das Spiel "Zahlenraten". In dem zu entwickelnden Python-Programm generiert der Computer eine Zufallszahl zwischen 1 und 100 (beide enthalten). Anschließend versucht die Spielerin, die generierte Zahl zu erraten. Hierzu gibt die Spielerin so lange Zahlen ein, bis sie die richtige Zahl gefunden hat.

Nutzen Sie bitte die vorgegebenen Programmteile und überarbeiten Sie diese. Weitere Hin-weise finden Sie an den durch '# ToDo' im Quellcode marklierten Stellen. Bitte ändern Sie keine bereits festgelegten Funktionsnamen. Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.

Folgende Anforderungen soll das Programm erfüllen:

- 1.1 Bitte verwenden Sie für jede ausgegebene Zeile, die Text enthält, einen separaten Print-Befehl. Print-Befehle, die nur einen oder mehrere Zeilenumbrüche (wie z. B. `print('\n')` oder `print(, '\n')` enthalten, sind untersagt.
- 1.2 Das Programm wählt eine Zufallszahl zwischen 1 und 100 (beide enthalten) aus.
- 1.3 Das Programm lässt die Spielerin über die Standardeingabe so lange Ihre Tipps eingeben, bis diese die richtige Zahl errät.
- 1.4 Das Programm prüft nach jeder Eingabe, ob die richtige Zahl erraten wurde.
- 1.5 Wenn die eingegebene Zahl zu niedrig ist, soll "Zahl zu niedrig" und bei mehr als 20 zu niedrig "Zahl viel zu niedrig" über die Standardausgabe ausgegeben werden.
- 1.6 Ist die eingegebene Zahl zu hoch, soll "Zahl zu groß" und bei mehr als 20 zu hoch "Zahl viel zu groß" über die Standardausgabe ausgegeben werden.
- 1.7 Wenn die Zahl richtig ist, soll "Richtig - Sehr gut!" über die Standardausgabe ausgegeben und das Spiel beendet werden.
- 1.8 Bei fehlerhaften Eingaben (keine Zahl oder keine Zahl von 1 bis 100) soll "Fehlerhafte Eingabe" über die Standardausgabe ausgegeben und das Raten einfach fortgesetzt werden.

(Hide)

Collapse Action Sidebar

Run | **Score** | **Request Comments**

Collapse Output Sidebar

Files

- solution01.py
- task01.py**
- test_util.py

```
1 # ToDo:
2 # Überprüfen Sie den gesamten Code, auch die bereits vorgegebenen Programmteile.
3 # Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.
4
5 from random import randint
6
7
8 def main():
9     correct_number = randint(1, 100)
10    print(
11        "Willkommen! Ich habe mir eine Zahl zwischen 1 und 100 ausgedacht, die Sie nun erraten können."
12    )
13
14    # ToDo:
15    # Implementieren Sie hier Code, der den Spieler solange um die Eingabe einer Zahl bittet,
16    # bis dieser die Zahl richtig erraten hat. Nutzen Sie für die Eingabe die Methode "guess_number" und
```

Results

4 test files have been executed.

Test File 1 (test_compare_numbers.py)

Passed Tests	0 out of 1
Score	0 out of 5



CodeOcean für Lernende

Aufgaben-Ansicht

The screenshot displays the CodeOcean interface for a task titled "WIDIG01_Guess-a-Number". The interface includes a top navigation bar with "CodeOcean", "Administration", "English", "Help", and a user profile "Jonas Genath (Admin)". Below the navigation bar is a breadcrumb trail: "EXERCISES / WIDIG01_GUESS-A-NUMBER / IMPLEMENT".

The task description is as follows:

▼ WIDIG01_Guess-a-Number 0%

Implementieren Sie das Spiel "Zahlenraten". In dem zu entwickelnden Python-Programm generiert der Computer eine Zufallszahl zwischen 1 und 100 (beide enthalten). Anschließend versucht die Spielerin, die generierte Zahl zu erraten. Hierzu gibt die Spielerin so lange Zahlen ein, bis sie die richtige Zahl gefunden hat.

Nutzen Sie bitte die vorgegebenen Programmteile und überarbeiten Sie diese. Weitere Hin-weise finden Sie an den durch '# ToDo' im Quellcode marklierten Stellen. Bitte ändern Sie keine bereits festgelegten Funktionsnamen. Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.

Folgende Anforderungen soll das Programm erfüllen:

- 1.1 Bitte verwenden Sie für jede ausgegebene Zeile, die Text enthält, einen separaten Print-Befehl. Print-Befehle, die nur einen oder mehrere Zeilenumbrüche (wie z. B. `print('\n')` oder `print(, '\n')` enthalten, sind untersagt.
- 1.2 Das Programm wählt eine Zufallszahl zwischen 1 und 100 (beide enthalten) aus.
- 1.3 Das Programm lässt die Spielerin über die Standardeingabe so lange Ihre Tipps eingeben, bis diese die richtige Zahl errät.
- 1.4 Das Programm prüft nach jeder Eingabe, ob die richtige Zahl erraten wurde.
- 1.5 Wenn die eingegebene Zahl zu niedrig ist, soll "Zahl zu niedrig" und bei mehr als 20 zu niedrig "Zahl viel zu niedrig" über die Standardausgabe ausgegeben werden.
- 1.6 Ist die eingegebene Zahl zu hoch, soll "Zahl zu groß" und bei mehr als 20 zu hoch "Zahl viel zu groß" über die Standardausgabe ausgegeben werden.
- 1.7 Wenn die Zahl richtig ist, soll "Richtig - Sehr gut!" über die Standardausgabe ausgegeben und das Spiel beendet werden.
- 1.8 Bei fehlerhaften Eingaben (keine Zahl oder keine Zahl von 1 bis 100) soll "Fehlerhafte Eingabe" über die Standardausgabe ausgegeben und das Raten einfach fortgesetzt werden.

The code editor shows the following Python code:

```
1 # ToDo:
2 # Überprüfen Sie den gesamten Code, auch die bereits vorgegebenen Programmteile.
3 # Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.
4
5 from random import randint
6
7
8 def main():
9     correct_number = randint(1, 100)
10    print(
11        "Willkommen! Ich habe mir eine Zahl zwischen 1 und 100 ausgedacht, die Sie nun erraten können."
12    )
13
14 # ToDo:
15 # Implementieren Sie hier Code, der den Spieler solange um die Eingabe einer Zahl bittet,
16 # bis dieser die Zahl richtig erraten hat. Nutzen Sie für die Eingabe die Methode "guess_number" und
```

The results panel shows:

Results

4 test files have been executed.

Test File 1 (test_compare_numbers.py)

Passed Tests	0 out of 1
Score	0 out of 5



CodeOcean für Lernende

Aufgaben-Ansicht

The screenshot displays the CodeOcean interface for a task titled "WIDIG01_Guess-a-Number". The task description includes instructions to implement a Python program that generates a random number between 1 and 100 and allows a user to guess it. The interface is divided into several sections:

- Task Description:** Explains the goal of the task and lists eight requirements (1.1 to 1.8) for the program's behavior.
- Files:** A sidebar showing the file structure with files named "solution01.py", "task01.py", and "test_util.py".
- Code Editor:** A central area with a Python code snippet for a "guess" function. The code includes comments in German and uses the "random" module to generate a number.
- Results:** A section on the right indicating that 4 test files have been executed, with a specific result for "Test File 1 (test_compare_numbers.py)" showing 0 out of 1 passed tests and 0 out of 5 score.

CodeOcean für Lernende

Aufgaben-Ansicht

The screenshot displays the CodeOcean interface for a task titled "WIDIG01_Guess-a-Number". The task description includes instructions to implement a Python program that generates a random number between 1 and 100 and allows a user to guess it. The requirements are listed as follows:

- 1.1 Bitte verwenden Sie für jede ausgegebene Zeile, die Text enthält, einen separaten Print-Befehl. Print-Befehle, die nur einen oder mehrere Zeilenumbrüche (wie z. B. `print('\n')` oder `print(, '\n')` enthalten, sind untersagt.
- 1.2 Das Programm wählt eine Zufallszahl zwischen 1 und 100 (beide enthalten) aus.
- 1.3 Das Programm lässt die Spielerin über die Standardeingabe so lange ihre Tipps eingeben, bis diese die richtige Zahl errät.
- 1.4 Das Programm prüft nach jeder Eingabe, ob die richtige Zahl erraten wurde.
- 1.5 Wenn die eingegebene Zahl zu niedrig ist, soll "Zahl zu niedrig" und bei mehr als 20 zu niedrig "Zahl viel zu niedrig" über die Standardausgabe ausgegeben werden.
- 1.6 Ist die eingegebene Zahl zu hoch, soll "Zahl zu groß" und bei mehr als 20 zu hoch "Zahl viel zu groß" über die Standardausgabe ausgegeben werden.
- 1.7 Wenn die Zahl richtig ist, soll "Richtig - Sehr gut!" über die Standardausgabe ausgegeben und das Spiel beendet werden.
- 1.8 Bei fehlerhaften Eingaben (keine Zahl oder keine Zahl von 1 bis 100) soll "Fehlerhafte Eingabe" über die Standardausgabe ausgegeben und das Raten einfach fortgesetzt werden.

The code editor shows the following Python code:

```
1 # TODO:
2 # Überprüfen Sie den gesamten Code, auch die bereits vorgegebenen Programmteile.
3 # Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.
4
5 from random import randint
6
7
8 def main():
9     correct_number = randint(1, 100)
10    print(
11        "Willkommen! Ich habe mir eine Zahl zwischen 1 und 100 ausgedacht, die Sie nun erraten können."
12    )
13
14 # TODO:
15 # Implementieren Sie hier Code, der den Spieler solange um die Eingabe einer Zahl bittet,
16 # bis dieser die Zahl richtig erraten hat. Nutzen Sie für die Eingabe die Methode "guess_number" und
```



CodeOcean für Lernende

Feedback für nicht bestandene Tests

▶ Run 🏆 Score 💬 Request Comments

```
1 # ToDo:
2 # Überprüfen Sie den gesamten Code, auch die bereits vorgegebenen Programmteile.
3 # Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.
4
5 from random import randint
6
7
8 def main():
9     correct_number = randint(1, 100)
10    print(
11        "Willkommen! Ich habe mir eine Zahl zwischen 1 und 100 ausgedacht, die Sie nun erraten können."
12    )
13
14    # ToDo:
15    # Implementieren Sie hier Code, der den Spieler solange um die Eingabe einer Zahl bittet,
16    # bis dieser die Zahl richtig erraten hat. Nutzen Sie für die Eingabe die Methode 'guess_number' und
17    # zur Überprüfung der Eingabe die Methode 'compare_numbers'.
18
19    guessed_number = guess_number()
20
21
22 def compare_numbers(guessed_number: int, correct_number: int) -> bool:
23
24    # ToDo:
25    # Implementieren Sie hier Code, der feststellt, ob die 'guessed_number' mit der
26    # 'correct_number' übereinstimmt. Stellen Sie sicher, dass eine Meldung ausgegeben wird,
27    # ob die geschätzte Zahl zu klein, viel zu klein, zu groß, viel zu groß oder richtig ist.
28    # Beachten Sie, dass die in der Aufgabenstellung spezifizierten Ausgaben korrekt ausgegeben werden.
29
30    return False
31
32
33 def guess_number() -> int:
34
35    # ToDo:
36    # Mithilfe dieser Funktion, eine Zahl von der Standardeingabe eingelesen und dann in eine
37    # Ganzzahl konvertiert werden. Dafür ist bereits eine entsprechende Codezeile vorhanden.
38    # Implementieren Sie hier Code, der die eingegebene Zahl überprüft.
39    # Wenn eine ungültige Zahl ('ValueError') oder eine Zahl,
40    # die nicht im Bereich von 1 bis 100 liegt, eingegeben wurde, sollte die Funktion die in der
41    # Aufgabenstellung spezifizizierte Fehlermeldung ausgeben und als Rückgabewert 0 zurückgeben.
42    # Wenn die Zahl gültig ist, soll die Funktion diese Zahl zurückgeben.
43
44    number = int(input("Ihr Tipp bitte: "))
45    return 0
46
```

▣ Collapse Output Sidebar

Results

4 test files have been executed.

Test File 1 (test_compare_numbers.py)

Passed Tests	0 out of 1
Score	0 out of 5
Feedback	So far, not all test cases for the method compare_numbers are fulfilled (more details below the scoring).
Error Messages	test_compare_numbers_string_outputs: False != True : ERROR/HINT: Ausgabe entspricht nicht den Anforderungen. (siehe Aufgabe 1.7) Soll: True Ist:False

Test File 2 (test_guess_number.py)

Passed Tests	0 out of 2
Score	0 out of 5
Feedback	So far, not all test cases for the method guess_number are fulfilled (more details below the scoring).
Error Messages	• test_guess_number_correct_inputs: 0 != 1 : ERROR/HINT:



CodeOcean für Lernende

Feedback für bestandene Tests

No Action Score Request Comments

```
1 from random import randint
2
3
4 def guess_number() -> int:
5     try:
6         number = int(input("Ihr Tipp bitte: "))
7         if number < 1 or number > 100:
8             raise ValueError
9         return number
10    except ValueError:
11        print("Fehlerhafte Eingabe")
12        return 0
13
14
15 def compare_numbers(correct_number: int, guessed_number: int) -> bool:
16     if correct_number == guessed_number:
17         print("Richtig - Sehr gut!")
18         return True
19     if guessed_number > correct_number:
20         if guessed_number - correct_number > 20:
21             print("Zahl viel zu groß.")
22         else:
23             print("Zahl zu groß.")
24         return False
25
26     if correct_number - guessed_number > 20:
27         print("Zahl viel zu niedrig.")
28     else:
29         print("Zahl zu niedrig.")
30     return False
31
32
33 def main():
34     print(
35         "Willkommen! Ich habe mir eine Zahl zwischen 1 und 100 ausgedacht, die Sie nun erraten können."
36     )
37     correct_number = randint(1, 100)
38     correct_guess = False
39     while not correct_guess:
40         guessed_number = guess_number()
41         if guessed_number == 0:
42             continue
```

Collapse Output Sidebar

Score	100 out of 100
Feedback	Well done. All tests have been passed.
Error Messages	

Test File 3 (test_main.py)

Passed Tests	1 out of 1
Score	5 out of 5
Feedback	Well done. All tests have been passed.
Error Messages	

Linter Feedback (test_style.py)

Code Rating	10 out of 10
Score	3 out of 3
Feedback	Well done. The linter is completely satisfied.
Messages	

Score: 100%

CodeOcean für Lernende

Kommentar-Anfragen in die Lerngemeinschaft

The screenshot shows the CodeOcean interface with a modal dialog box titled "Kommentaranfrage stellen". The dialog contains the following text:

Bitte beschreiben Sie kurz Ihre Probleme oder nennen Sie den Programmteil, zu dem Sie Feedback wünschen. Ihr Programmcode und eventuelle Fehlermeldungen werden automatisch zur Anfrage hinzugefügt.

Momentan habe ich 0 Fehler aber eine Warnung beim Style. Ändere ich nun Zeile 50 zu "except TypeError:" habe ich 100% Style aber ein Test schlägt fehl. Könnten Sie mir sagen woran dies liegt?

Buttons in the dialog: "Kommentaranfrage stellen" (blue) and "Fenster schließen" (orange).

The background interface shows a code editor with Python code for a number-guessing game. The code includes comments and a function `main()` that generates a random number and prompts the user to guess it. The interface also features a sidebar with "Aktions-Leiste Einklappen", "Dateien" (showing `task01.py`), and "Tipps".

CodeOcean für Lehrende

Aufgaben erstellen

1. Aufgabenbeschreibung

WIDIG01_Guess_a_Number_100

Bearbeiten ▾

Titel WIDIG01_Guess_a_Number_100

Alternativer Titel —

Autor Daniel Fischer

Beschreibung Implementieren Sie das Spiel "Zahlenraten". In dem zu entwickelnden Python-Programm generiert der Computer eine Zufallszahl. Anschließend versucht die Spielerin, die generierte Zahl zu erraten. Hierzu gibt die Spielerin so lange Tipps ein, bis sie die richtige Zahl gefunden hat.

Bitte beachten Sie die folgenden weiteren Anforderungen:

- Bitte ändern Sie keine bereits festgelegten Funktionsnamen.
- Achten Sie auf den Programmierstil und stellen Sie den Linter zufrieden.
- Bitte nutzen Sie für jede Textzeile, die über die Standardausgabe ausgegeben werden soll, einzelne Print-Befehle und vermeiden Sie bitte Print-Befehle, die nur Zeilenumbrüche enthalten, wie z. B. `print("\n")`.
- Zusätzlicher Hinweis: Überlegen Sie, welche der Anforderungen Sie effizienter mithilfe von Funktionen implementieren können. Insbesondere bei sich wiederholenden bzw. mehrfach im Programm vorkommenden Abläufen eignet sich der Einsatz von Funktionen.

2. Parameter

führungsumgebung Python 3.8

ichbare 30.0

ktzahl
abefrist 2023-11-15 23:59:00 +0100

ipätete —
abefrist

ntlich ×

ktiviert ×

ibaum ×

tecken

ierstellung ×

uben

vervollständigung ✓

vieren

vierigkeitsgrad 1

D —

i —

imeter für LTI- locale=de&token=01c1bc8b

ettung

3. Dateien (Hauptdatei, Testdatei, ...)

Tipps

Tipps

Tipps

Dateien

default.pylintrc

programm_test_runner.py

solution01.py

task01.py

test_compare_numbers.py

test_compare_numbers_data.py

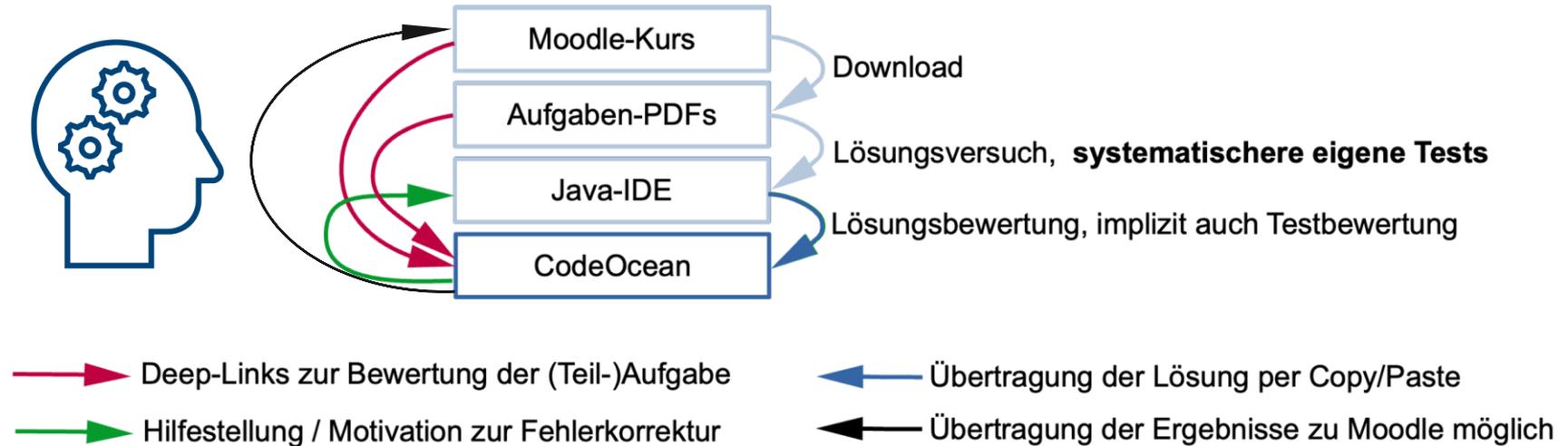
test_guess_number.py

test_guess_number_data.py

Didaktische Integration



Beispiel: Erste Programmierfähigkeiten in Java



Didaktische Integration



Welche Schwerpunktprobleme gibt es?

- neben funktionalen Tests auch Tests auf formale Anforderungen der Aufgabenstellung (Rückgabetypen, ...)
- Fehlerbeschreibungen müssen an Anfängerniveau angepasst sein
→ Hilfe zur Fehlersuche oder Behebung
- Testwerkzeuge (JUnit, PyUnit) sind häufig auf Softwareentwickler zugeschnitten
→ Fehlermeldungen für Anfänger häufig nicht verständlich
→ Studierende verlieren ihre Motivation
- Voneinander unabhängige Tests dürfen sich nicht beeinflussen
→ wichtig für Erfolgserlebnisse / Motivation

Didaktische Integration



Wie sehen die verfolgten Lösungsansätze aus?

1. Hilfestellung für Fehlerlokalisierung (Anzeige von Parametern/Werten, die zu Fehlern führten) → Nachvollziehbarkeit sicherstellen
2. Entwicklung eigener Testframeworks auf Basis bestehender Testwerkzeuge
3. was die Aufgabenstellung nicht explizit fordert, wird auch nicht getestet



Erfolgreiche Tests	0 von 2
Punktzahl	0 von 5
Feedback	So far, not all test cases for the open_cars_with_most_reservations function are fulfilled (more details below the scoring).
Fehlermeldungen	<ul style="list-style-type: none">• test_1_open_cars_with_most_reservations: 10 != 0 : Aufgabe 13: 10 Fehler für den User-Input 0 aufgetreten.<ul style="list-style-type: none">• Die 3. Ausgabe ist fehlerhaft. Soll: 96 Tage - Opel Corsa. Ist: 86 Tage - Skoda Octavia.

Didaktische Integration



Welche Erfahrungen konnten gesammelt werden?

1. Befragungen und Beobachtung ergaben:

- Einsatz von CodeOcean von den Studierenden insgesamt als positiv bewertet
- Studierende lobten vor allem die Möglichkeit, jederzeit automatisches Feedback zu ihren Lösungsversuchen abrufen zu können (auch bei den Abschlusstests)

2. aber immer noch Kritikpunkte:

- teilweise ungenaues, verwirrendes oder nicht gut nachvollziehbares Feedback
- unklare und missverständliche Aufgabenstellungen

3. Workflow muss für die Studierenden einfach nachvollziehbar und durchführbar sein

4. Hauptarbeitspunkt: Verständlichkeit des Feedbacks und der Aufgabenstellungen

Technische Integration

Adaptierbarkeit von CodeOcean

1. Angepasste Execution Environments für weitere Use Cases
2. Adapter für weitere Programmiersprachen / Testumgebungen

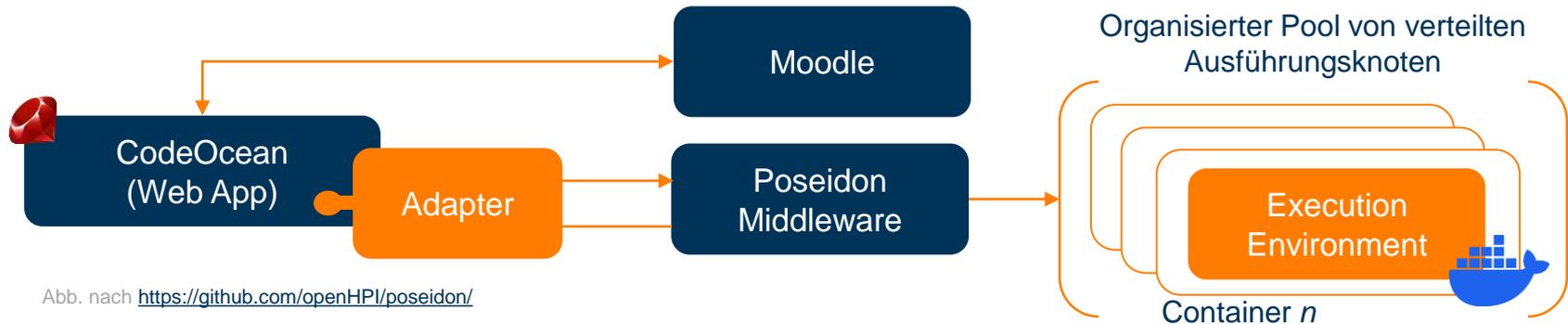


Abb. nach <https://github.com/openHPI/poseidon/>

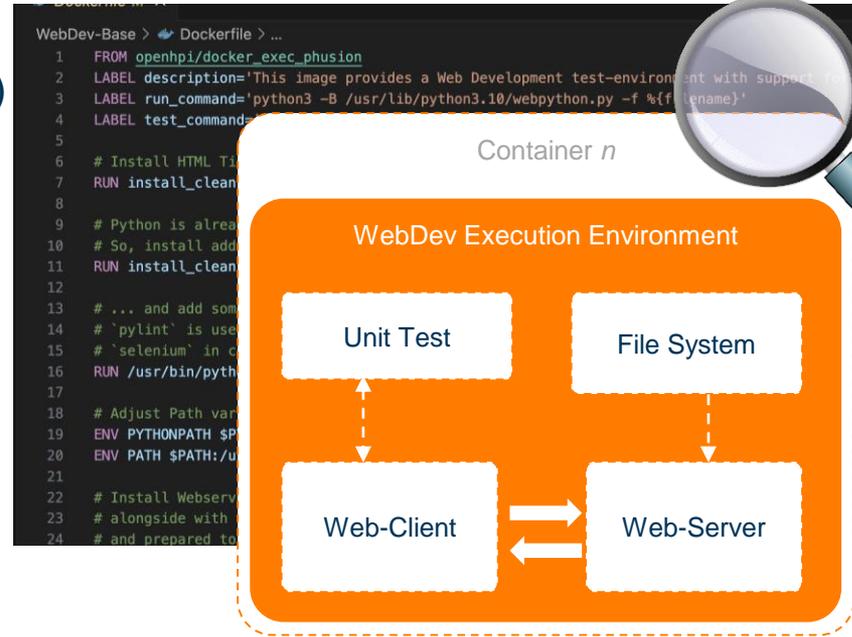
Beispiel: Angepasstes Execution Environment

Wie kann kompetenzorientiert mit E2E-Tests geprüft werden?

Szenario: Aufgaben (HTML, CSS, JS) über lokalen Web-Server & Web-Client testen (E2E)

Ansatz: Single-Container-Anwendung mit lokaler Client-Server-Architektur und Client-Schnittstelle für Unittests

- angepasstes Single-Container-Images
- Start mehrerer Prozesse in einem einzelnen Container
- Steuerung des Web-Client mittels Selenium-Webdriver



Beispiel: Adapter für Rust



Bessere Wissensvermittlung im Bereich Systemsoftware, aber wie?

Szenario: Programmiersprache Rust

- robuste und effiziente Implementierungen
- hardwarenahe Speicherverwaltung
- nichtfunktionale Tests

Ansatz: Entwicklung eines Rust-spezifischen Adapters für Integration in CodeOcean

- Schnittstelle zum Web-Frontend: Anzahl ausgeführte Tests, Anzahl fehlgeschlagene Tests, Fehlermeldungen
- Ausgaben des Build-Tools cargo extrahierbar
- strukturiertes JSON-Format genutzt, z.B.:

```
1 main.rs 2 thread.rs
1 /// Module for the [`Thread`] trait, which should be implemented
2 /// type [`MyThread`].
3
4 use crate::Dtq;
5 use std::fmt::Display;
6
7 /// Trait for thread behavior.
8 // `Display` is called a _supertrait_ of `Thread`, which means
9 // implementing `Thread` has to implement `Display` as well
10
11 >> pub trait Thread: Display {
12     /// Simulates a thread working for a maximal interval of `d
13     /// quants.
14     ///
15     /// Returns the duration of effective computation activity
16     fn do_work(&mut self, dur: Dtq) -> Dtq;
17 }
```



```
{"type": "suite", "event": "ok", "passed": 4, "failed": 0, "ignored": 1,
  "measured": 0, "filtered_out": 0, "exec_time": 0.000473204}
```

Vielen Dank für ihre Aufmerksamkeit!



Möchten Sie mehr wissen?

- Jonas.Genath@tu-ilmenau.de
- Daniel.Fischer@tu-ilmenau.de
- Ulf.Doering@tu-ilmenau.de
- Gunther.Kreuzberger@tu-ilmenau.de
- Peter.Amthor@tu-ilmenau.de