

HW_10.2

Regular Ball Super Ball

```
class Ball(object):

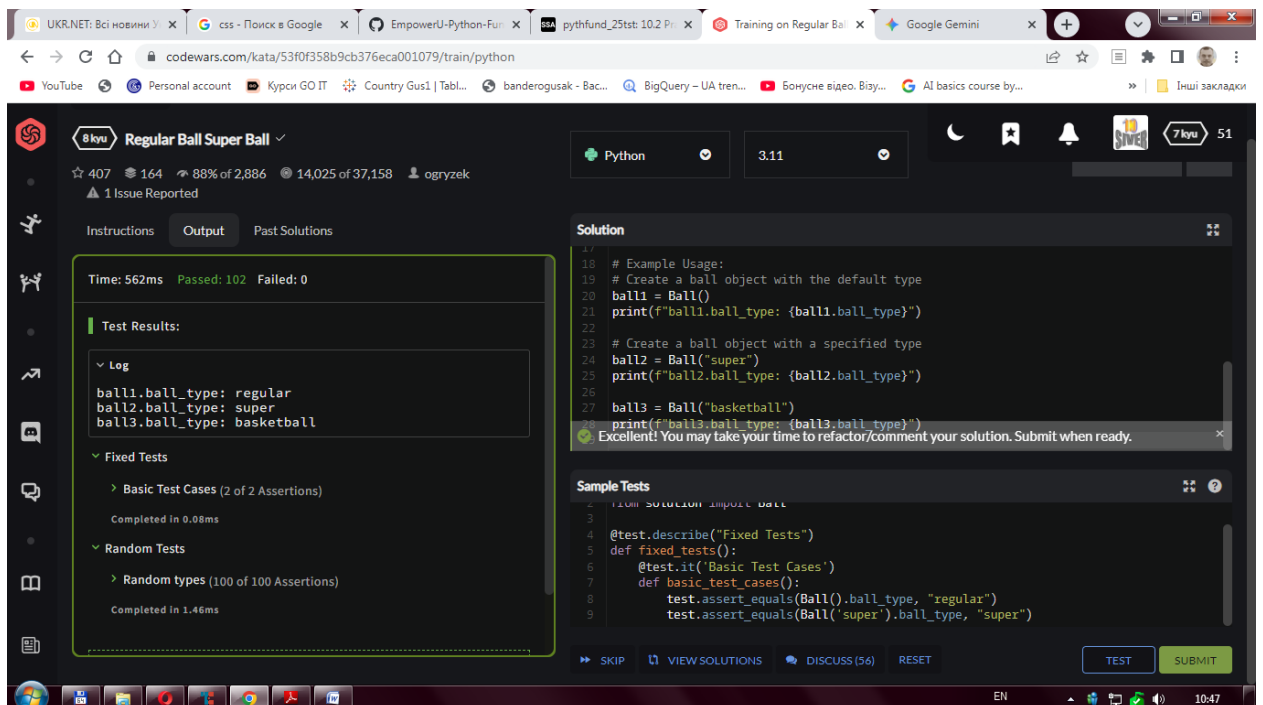
    def __init__(self, ball_type="regular"):

        self.ball_type = ball_type

ball1 = Ball()
print(f"ball1.ball_type: {ball1.ball_type}")

ball2 = Ball("super")
print(f"ball2.ball_type: {ball2.ball_type}")

ball3 = Ball("basketball")
print(f"ball3.ball_type: {ball3.ball_type}")
```



Color Ghost

```
import random

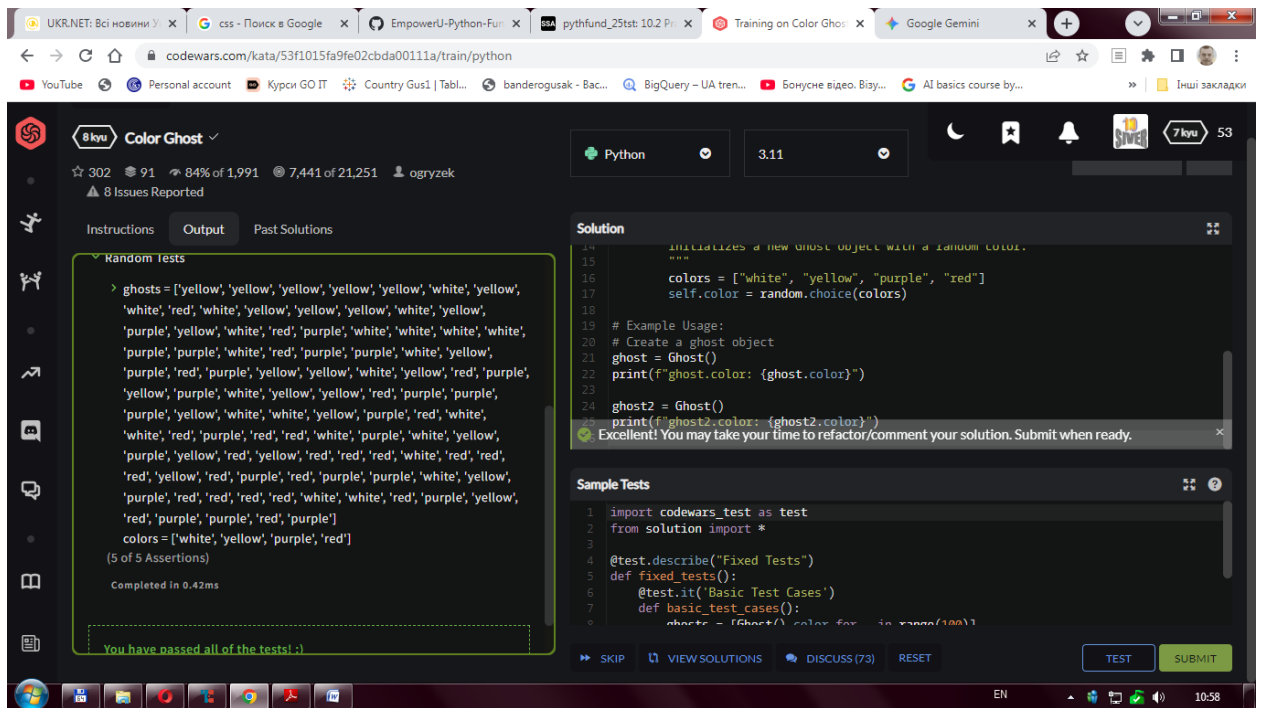
class Ghost(object):

    def __init__(self):

        colors = ["white", "yellow", "purple", "red"]
        self.color = random.choice(colors)

ghost = Ghost()
print(f"ghost.color: {ghost.color}")

ghost2 = Ghost()
print(f"ghost2.color: {ghost2.color}")
```



Basic subclasses - Adam and Eve

```
class Human(object):

    def __init__(self, name="Human"):
        self.name = name
        print(f"A new Human named {self.name} has been created.")

class Man(Human):

    def __init__(self, name="Adam"):
        super(Man, self).__init__(name) # Call the parent class (Human) constructor
        print(f"Specifically, {self.name} is a Man.")

class Woman(Human):

    def __init__(self, name="Eve"):

        super(Woman, self).__init__(name) # Call the parent class (Human) constructor
        print(f"Specifically, {self.name} is a Woman.")

def God():

    print("--- God is beginning creation ---")
    adam = Man() # Instantiate Adam
    eve = Woman() # Instantiate Eve
    print("--- Creation complete ---")
    return [adam, eve]

# Example Usage:
first_humans = God()

print("\nChecking types and names:")
if isinstance(first_humans[0], Man) and isinstance(first_humans[0], Human):
    print(f"First human is a Man (and a Human): {first_humans[0].name}")
if isinstance(first_humans[1], Woman) and isinstance(first_humans[1], Human):
    print(f"Second human is a Woman (and a Human): {first_humans[1].name}")

print("\nAccessing their names directly:")
print(f"Adam's name: {first_humans[0].name}")
print(f"Eve's name: {first_humans[1].name}")
```

UKR.NET: Всі новини

css - Поиск в Google

EmpowerU-Python-Fun

pythfund_25tst: 10.2 Pr

Training on Basic subcl

Google Gemini

codewars.com/kata/547274e24481cfc469000416/train/python

YouTube

Personal account

Курси GO IT

Country Gusl | Tabl...

banderogusak - Bac...

BigQuery - UA tren...

Бонусне відео. Візу...

AI basics course by...

Інші закладки

Kata Training

8 kyu

Basic subclasses - Adam and Eve

☆ 351

👤 121

👍 78% of 2,489

👤 6,613 of 19,379

hansrusten

Python

3.11

VIM

EMACS

7 kyu

55

Instructions

Output

Adam's name: Adam
Eve's name: Eve
--- God is beginning creation ---
A new Human named Adam has been created.
Specifically, Adam is a Man.
A new Human named Eve has been created.
Specifically, Eve is a Woman.
--- Creation complete ---

Human test

✔ Test Passed

✔ Test Passed

✔ Test Passed

✔ Test Passed

✔ Test Passed

You have passed all of the tests! :)

Solution

39 # example usage:
60 first_humans = God()
61
62 print("\nChecking types and names:")
63 if isinstance(first_humans[0], Man) and isinstance(first_humans[0], Human):
64 print(f"First human is a Man (and a Human): {first_humans[0].name}")
65 if isinstance(first_humans[1], Woman) and isinstance(first_humans[1], Human):
66 print(f"Second human is a Woman (and a Human): {first_humans[1].name}")
67
68 print("\nAccessing their names directly:")
69 print(f"Adam's name: {first_humans[0].name}")
70 print(f"Eve's name: {first_humans[1].name}")

✔ Correct! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

1 paradise = God()
2 test.assertEqual(isinstance(paradise[0], Man) , True, "First object are a man")

SKIP

UNLOCK SOLUTIONS

DISCUSS (55)

RESET

TEST

SUBMIT

EN

11:02

Classy Classes

```
class Person(object):

    def __init__(self, name, age):

        self.name = name
        self.age = age

    @property
    def info(self):
        return f"{self.name}'s age is {self.age}"

person1 = Person("john", 34)
print(f"Person 1 Info: {person1.info}")

person2 = Person("alice", 28)
print(f"Person 2 Info: {person2.info}")
```

The screenshot shows a web browser with multiple tabs open, including 'codewars.com/kata/55a144eff5124e546400005a/train/python'. The main content area displays the 'Classy Classes' challenge on Codewars. The challenge is rated 8 kyu and has 164 stars. The user has completed the challenge successfully, as indicated by the message 'You have passed all of the tests! :)' and the 'Correctamundo!' message. The solution code is displayed in the 'Solution' tab, and the 'Sample Tests' tab shows the test cases and the expected output.

Classy Classes

8 kyu

164 stars

86% of 1,997

9,580 of 18,438

matt c

Instructions

Output

Testing for 'blop' and 43

Testing for 'matt' and 23

Testing for 'leroy' and 34

Testing for 'jeff' and 89

Testing for 'horse' and 25

Testing for 'santa' and 68

Testing for 'morgan' and 28

Testing for 'alex' and 27

Completed in 4.30ms

You have passed all of the tests! :)

Solution

```
25 Returns:
26 str: A string in the format "Name's age is Age".
27
28 """
29 return f"{self.name}'s age is {self.age}"
30
31 # Example Usage:
32 person1 = Person("john", 34)
33 print(f"Person 1 Info: {person1.info}")
34
35 person2 = Person("alice", 28)
36 print(f"Person 2 Info: {person2.info}")
37
38 Correctamundo! You may take your time to refactor/comment your solution. Submit when ready.
```

Sample Tests

```
1 names=["john", "matt", "alex", "cam"]
2 ages=[16,25,57,39]
3 for i in range(4):
4     name,age=names[i],ages[i]
5     person=Person(name,age)
6     @test.it("Testing for %s and %s" % (repr(name),age))
7     def basic_test_cases():
8         test.assertEqual(person.info, name+"'s age is "+str(age))
```

SKIP UNLOCK SOLUTIONS DISCUSS (27) RESET TEST SUBMIT

Building Spheres

```
import math

class Sphere(object):

    def __init__(self, radius, mass):

        self.radius = radius
        self.mass = mass

    def get_radius(self):

        return self.radius

    def get_mass(self):

        return self.mass

    def get_volume(self):
        volume = (4/3) * math.pi * (self.radius ** 3)
        return round(volume, 5)

    def get_surface_area(self):
        surface_area = 4 * math.pi * (self.radius ** 2)
        return round(surface_area, 5)

    def get_density(self):

        raw_volume = (4/3) * math.pi * (self.radius ** 3)
        if raw_volume == 0:
            return 0.0 # Avoid division by zero if radius is 0
        density = self.mass / raw_volume
        return round(density, 5)

ball = Sphere(2, 50)
print(f"Radius: {ball.get_radius()}")
print(f"Mass: {ball.get_mass()}")
print(f"Volume: {ball.get_volume()}")
print(f"Surface Area: {ball.get_surface_area()}")
print(f"Density: {ball.get_density()}")

print("\n--- Another Example ---")
small_ball = Sphere(0.5, 10)
print(f"Radius: {small_ball.get_radius()}")
print(f"Mass: {small_ball.get_mass()}")
print(f"Volume: {small_ball.get_volume()}")
print(f"Surface Area: {small_ball.get_surface_area()}")
print(f"Density: {small_ball.get_density()}")
```

UKR.NET: Всі нові...css - Пошук в GoogleEmpowerU-Pythonpythfund_25tstb10Training on BuildingGoogle GeminiGoogle Переклад...

codewars.com/kata/55c1d030da313ed05100005d/train/python

YouTubePersonal accountКурси GO ITCountry Gus1 | Tabl...banderogusak - Bac...BigQuery - UA tren...Бонусне відео. Візу...AI basics course by...Інші закладки

7 kyuBuilding Spheres

☆ 61👤 27👍 91% of 458👥 2,421 of 3,179👤 NaMe613

🚩 1 Issue Reported

InstructionsOutput

> Sphere with radius of 25.86 and mass of 198.68 (5 of 5 Assertions)

> Sphere with radius of 20.39 and mass of 193.36 (5 of 5 Assertions)

> Sphere with radius of 4.99 and mass of 128.06 (5 of 5 Assertions)

> Sphere with radius of 6.57 and mass of 72.99 (5 of 5 Assertions)

> Sphere with radius of 26.58 and mass of 10.42 (5 of 5 Assertions)

> Sphere with radius of 0.99 and mass of 234.31 (5 of 5 Assertions)

> Sphere with radius of 1.81 and mass of 27.09 (5 of 5 Assertions)

> Sphere with radius of 5.11 and mass of 149.04 (5 of 5 Assertions)

> Sphere with radius of 23.67 and mass of 32.42 (5 of 5 Assertions)

Completed in 4.49ms

You have passed all of the tests! :)

Python3.11

Solution

```
82 print(f'Mass: {ball.get_mass()}')
86 print(f'Volume: {ball.get_volume()}')
87 print(f'Surface Area: {ball.get_surface_area()}')
88 print(f'Density: {ball.get_density()}')
89
90 print("\n--- Another Example ---")
91 small_ball = Sphere(0.5, 10)
92 print(f'Radius: {small_ball.get_radius()}')
93 print(f'Mass: {small_ball.get_mass()}')
94 print(f'Volume: {small_ball.get_volume()}')
95 print(f'Surface Area: {small_ball.get_surface_area()}')
96 print(f'Density: {small_ball.get_density()}')
```

Correct! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

```
1 import codewars_test as test
2 from solution import *
3
4 @test.describe("Fixed Tests")
5 def fixed_tests():
6     @test.it("Basic Test Cases")
7     def basic_test_cases():
8
9
```

SKIPUNLOCK SOLUTIONSDISCUSS (14)RESET

TESTSUBMIT

EN

11:36

```

import math

class Sphere(object):

    def __init__(self, radius, mass):

        self.radius = radius
        self.mass = mass

    def get_radius(self):

        return self.radius

    def get_mass(self):

        return self.mass

    def get_volume(self):

        volume = (4/3) * math.pi * (self.radius ** 3)
        return round(volume, 5)

    def get_surface_area(self):

        surface_area = 4 * math.pi * (self.radius ** 2)
        return round(surface_area, 5)

    def get_density(self):

        raw_volume = (4/3) * math.pi * (self.radius ** 3)
        if raw_volume == 0:
            return 0.0 # Avoid division by zero if radius is 0
        density = self.mass / raw_volume
        return round(density, 5)

def class_name_changer(cls, new_name):

    if not isinstance(new_name, str):
        raise ValueError("New name must be a string.")
    if not new_name.isalnum():
        raise ValueError("New name must contain only alphanumeric characters.")
    if not new_name[0].isupper():
        raise ValueError("New name must start with an uppercase letter.")

    cls.__name__ = new_name
    print(f"Class name changed to: {cls.__name__}")

ball = Sphere(2, 50)
print(f"Original Sphere class name: {Sphere.__name__}")

try:
    class_name_changer(Sphere, "UsefulClass")
    print(f"New Sphere class name after first change: {Sphere.__name__}")
except ValueError as e:
    print(f"Error changing class name: {e}")

try:
    class_name_changer(Sphere, "SecondUsefulClass")
    print(f"New Sphere class name after second change: {Sphere.__name__}")
except ValueError as e:
    print(f"Error changing class name: {e}")

print("\n--- Testing invalid names ---")
try:
    class_name_changer(Sphere, "invalid name") # Contains space

```

```

except ValueError as e:
    print(f"Error changing class name: {e}")

try:
    class_name_changer(Sphere, "startslow") # Starts with lowercase
except ValueError as e:
    print(f"Error changing class name: {e}")

try:
    class_name_changer(Sphere, "With!Symbol") # Contains symbol
except ValueError as e:
    print(f"Error changing class name: {e}")

# Example of a dummy class to demonstrate
class MyClass:
    pass

print(f"\nOriginal MyClass name: {MyClass.__name__}")
try:
    class_name_changer(MyClass, "RenamedClass")
    print(f"New MyClass name: {MyClass.__name__}")
except ValueError as e:
    print(f"Error changing MyClass name: {e}")

```

The screenshot shows a web browser with multiple tabs open, including 'UKR.NET: Всі нові...', 'css - Пошук в Google', 'EmpowerU-Python', 'pythfund_25tst: 10', 'Training on Python', 'Google Gemini', and 'Google Переклад'. The main content is a coding challenge on Codewars titled 'Python's Dynamic Classes #1' with a rating of 7kyu. The challenge has 47 stars, 31 comments, and 87% of 285 users have solved it. The user 'adam-tokarski' has solved it, and there is 1 issue reported. The challenge is in the 'Python' language and has a score of 3.11. The user is using 'VIM' and 'EMACS' editors. The challenge description is 'Class name changer to, RenameClass. New MyClass name: RenamedClass'. The tests passed are: 'Test Passed', 'First try (3 of 3 Assertions)', 'Check for empty name', 'Check for name starting with lower case (2 of 2 Assertions)', 'Check for name with illegal chars (4 of 4 Assertions)', 'Check for name starts with a number (2 of 2 Assertions)', and 'Check for random names (20 of 20 Assertions)'. The solution code is:


```

130 print(f"Error changing class name: {e}")
131
132 try:
133     class_name_changer(Sphere, "With!Symbol") # Contains symbol
134 except ValueError as e:
135     print(f"Error changing class name: {e}")
136
137 # Example of a dummy class to demonstrate
138 class MyClass:
139     pass
140
    
```

 The sample tests are:


```

4 myobj = MyClass()
5 test.assertEqual(MyClass.__name__, "MyClass")
6
7 class_name_changer(MyClass, "UsefulClass");
8 test.assertEqual(MyClass.__name__, "UsefulClass")
9
10 class_name_changer(MyClass, "SecondUsefulClass");
11 test.assertEqual(MyClass.__name__, "SecondUsefulClass")
    
```

 The user has passed all the tests, and the challenge is now 'Solved'. The bottom of the screen shows the Windows taskbar with the time 11:44.