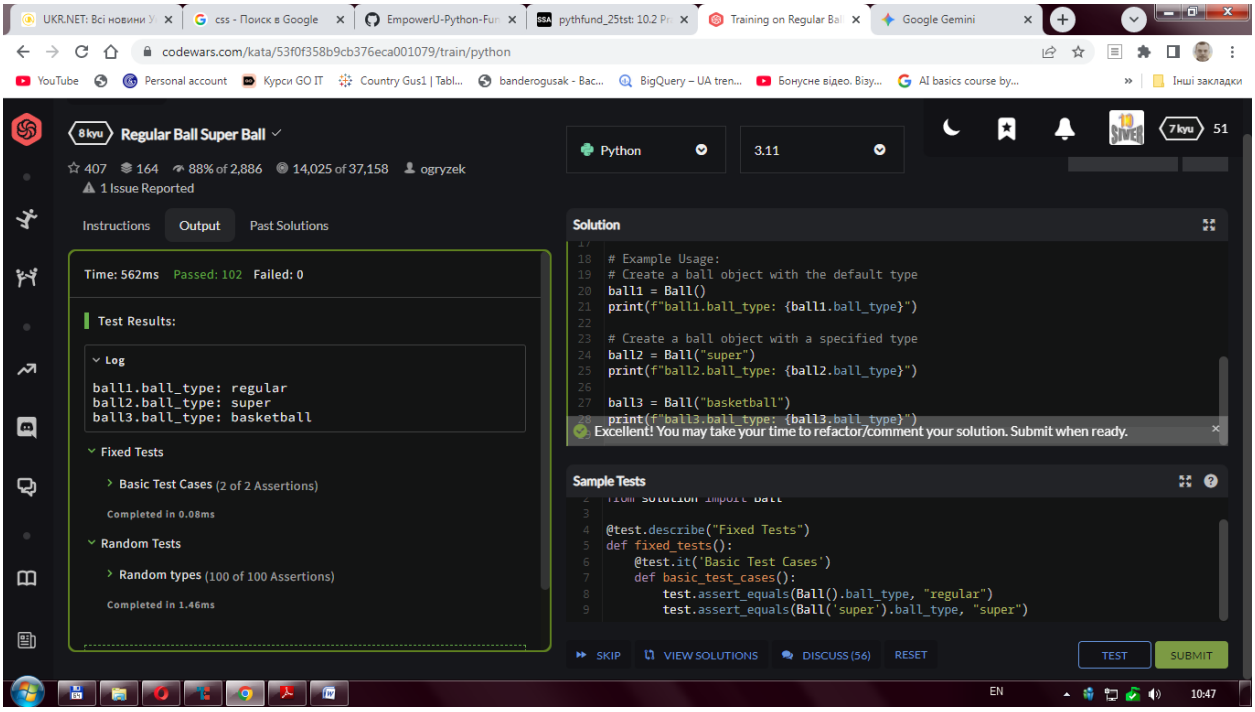
HW_10.2

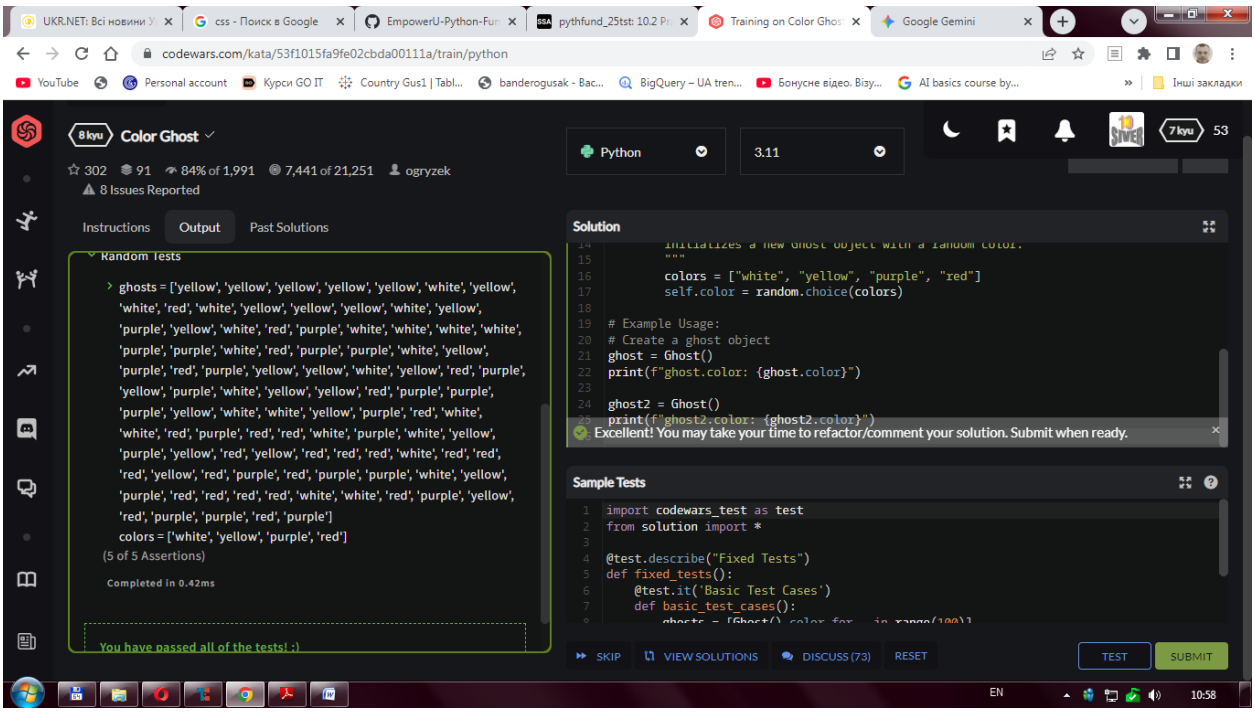
Regular Ball Super Ball

```
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}")
```



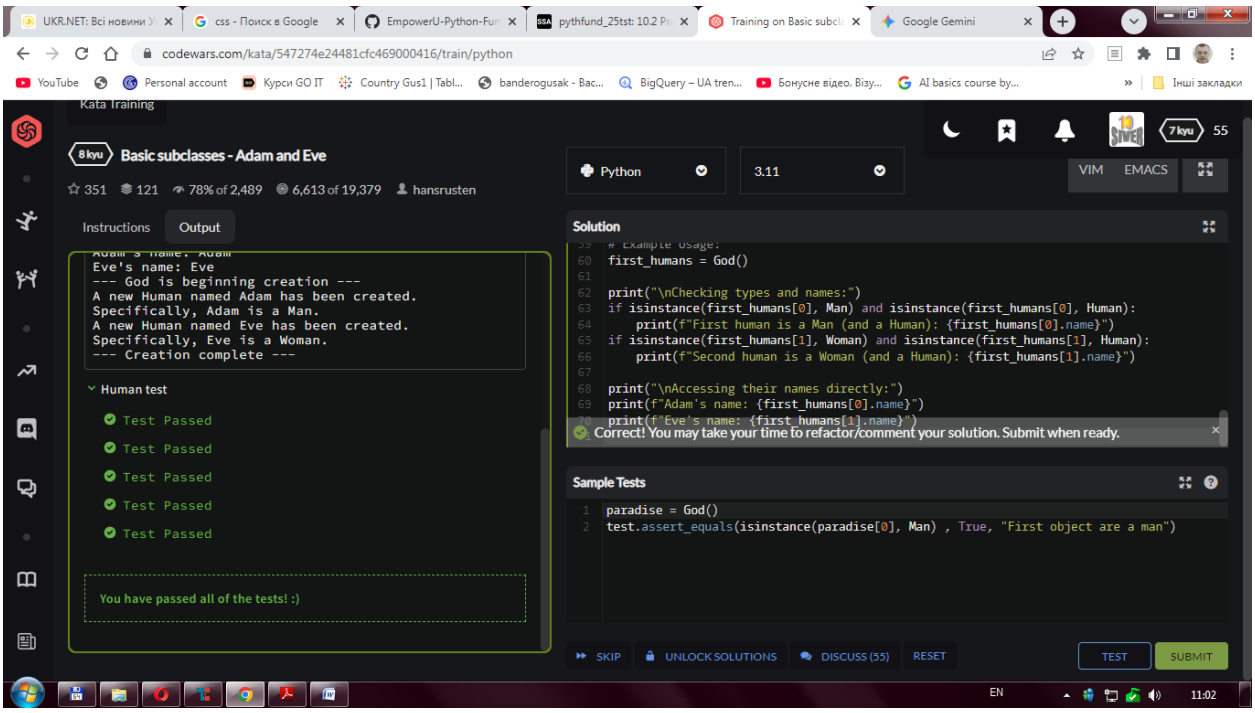
Color Ghost

```
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}")
```



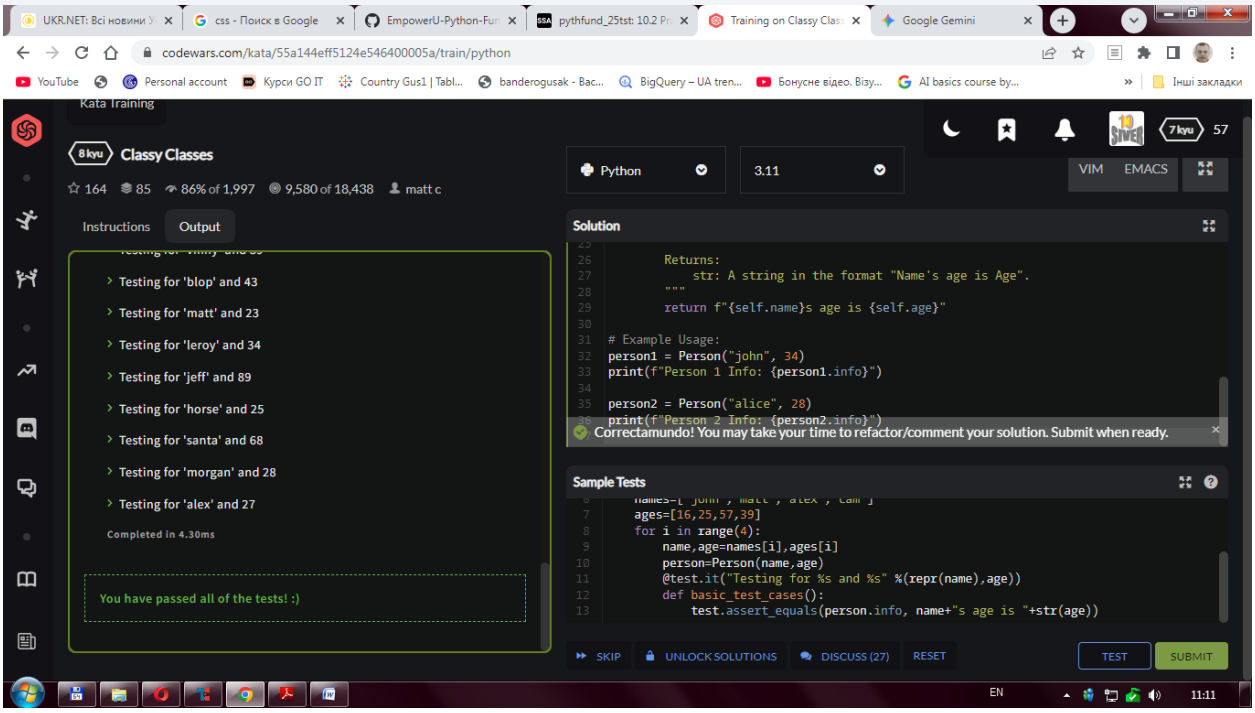
Basic subclasses - Adam and Eve

```
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}")
```


codewars.com/kata/547274e24481cfc469000416/train/python
Kata Training
8 kyu
Basic subclasses - Adam and Eve
351
121
78% of 2,489
6,613 of 19,379
hansrusten
Instructions
Output
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! :)
Python
3.11
VIM
EMACS
Solution
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}")
Correct! You may take your time to refactor/comment your solution. Submit when ready.
Sample Tests
paradise = God()
test.assert_equals(isinstance(paradise[0], Man) , True, "First object are a man")
SKIP
UNLOCK SOLUTIONS
DISCUSS (55)
RESET
TEST
SUBMIT
EN
11:02

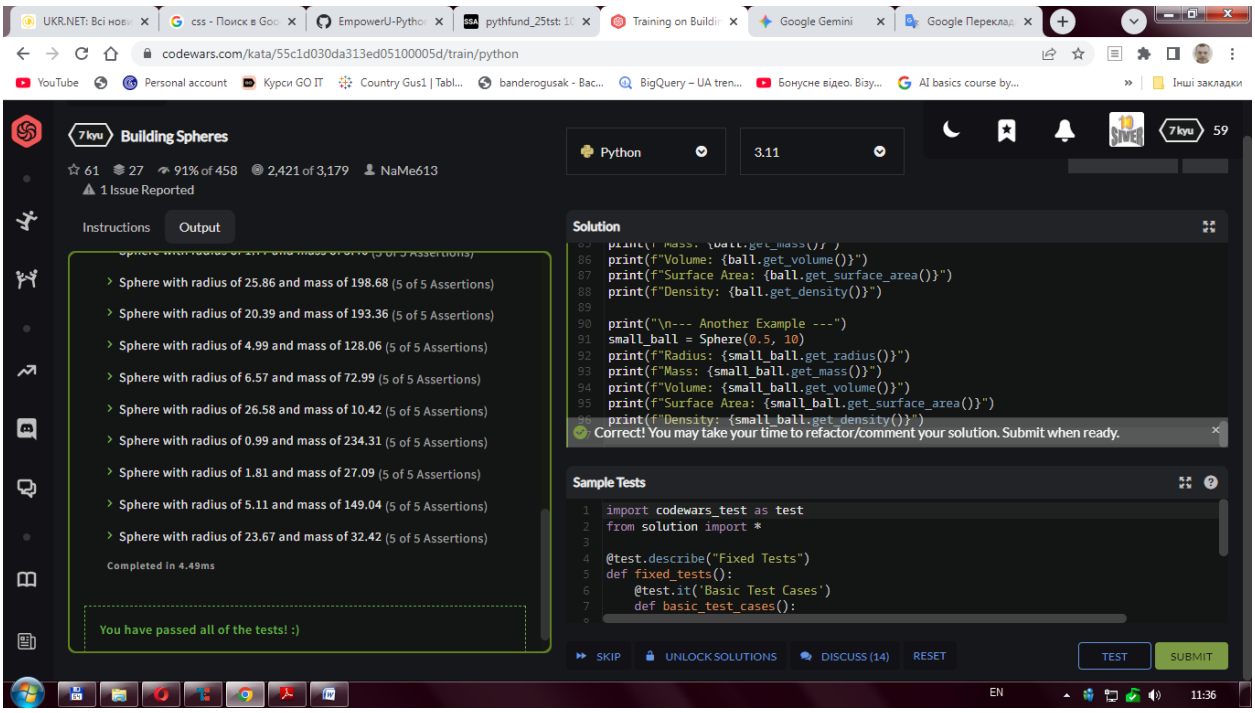
Classy Classes

```python
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}")
```



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()}")
```


codewars.com/kata/55c1d030da313ed05100005d/train/python
7 kyu
Building Spheres
61
27
91% of 458
2,421 of 3,179
NaMe613
1 Issue Reported
Instructions
Output
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! :)
Python
3.11
Solution
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()}")
Correct! You may take your time to refactor/comment your solution. Submit when ready.
Sample Tests
import codewars_test as test
from solution import *
@test.describe("Fixed Tests")
def fixed_tests():
@test.it('Basic Test Cases')
def basic_test_cases():
SKIP
UNLOCK SOLUTIONS
DISCUSS (14)
RESET
TEST
SUBMIT
EN
11:36

Dynamic Classes #1

```python
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}")
```