Student Challenge

Class Dice

sides   __init__( )

roll_dice( )

1 to 6

1 to 6

random

← randint

6 sides

12 sides

```python
from random import *

class Dice:

    def __init__(self, sides):
        self.sides = sides

    def roll_dice(self):
        return randint(1,self.sides)


d1 = Dice(6)

print(d1.roll_dice())
print(d1.roll_dice())
```

Run: SCOops1

```
/Users/abdulbari/PycharmProjects/Oop/venv/bin/python /Users/abdulbari/PycharmProjects/Oop/SCOops1.py
6
4


Process finished with exit code 0
```

▶ Run | ☰ TODO | ⊘ Problems | ▣ Terminal | ⊕ Python Packages | ⊕ Python Console
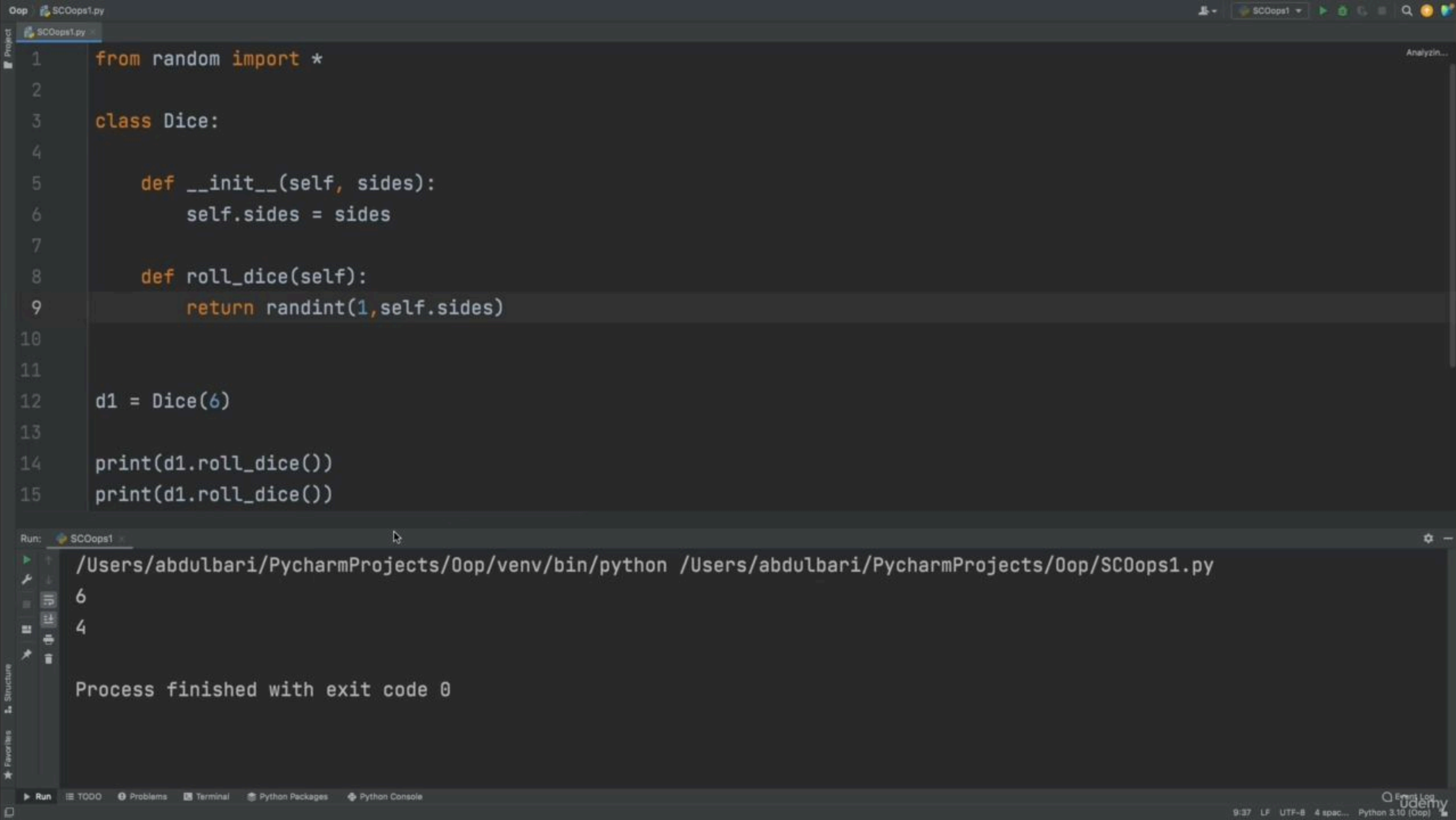
Student Challenge

Write a Class for Circle

✓ radius

✓ area()

✓ perimeter()

$\cdot$ PI

$area()$ $\pi * radius^2$

$preimetu()$ $2 * \pi * radius$

```python
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius * self.radius

    def perimeter(self):
        return 2 * math.pi * self.radius


c1 = Circle(7)
print('Area:', c1.area())
print('Perimeter:', c1.perimeter())
```

```python
class Book:

    def __init__(self, title, author, price):
        self.title = title
        self.author = author
        self.price = price

    def show_details(self):
        print('Title:', self.title)
        print('Author:', self.author)
        print('Price:', self.price)


b1 = Book('Python Crash Course', 'Eric Matthews', 1000)

b2 = Book('Learn Python', 'Mark Lutz', 500)

b1.show_details()
print('')
b2.show_details()
```

Student Challenge

Static Method

Class Calculator

add(a , b) $a+b$
sub(a , b) $a-b$
mul(a , b) $a \ast b$
div(a , b) a / b

class calculator

@static method
def add(a, b)
    return a+b

$x = 10$
$y = 5$
print (calcutor. add(x,y))

```python
class Calculator:

    @staticmethod
    def add(a, b):
        return a + b

    @staticmethod
    def sub(a, b):
        return a - b

    @staticmethod
    def mul(a, b):
        return a * b

    @staticmethod
    def div(a, b):
        return a / b


x = 10
y = 3

print(Calculator.add(x,y))
print(Calculator.div(x,y))
```

Student Challenge

Instance and Class Variable

class Employee

- name
- salary
- emp_id
- designation

e101   e102

employee_count = 101

show-details( )

total-emp ( ) employee-count=101

Student Challenge

Instance and Class Variable

class Employee

name

salary

→ emp_id

designation

e101 e102

employee_count = 101

show - details ( )

talal - emp ( ) employee - count → 101

_ _init_ _ ( )

show - details ( )

@staticmethod @classmethod

total - emp ( )

```python
class Employee:


    employee_count = 101


    def __init__(self, name, desig, sal):
        self.name = name
        self.designation = desig
        self.salary = sal
        self.eid = 'e' + str(Employee.employee_count)
        Employee.employee_count += 1

    def show_details(self):
        print('Name:', self.name)
        print('Eid:', self.eid)
        print('Designation:', self.designation)
        print('Salary:', self.salary)

    @classmethod
    def total_emp(cls):
        return cls.employee_count - 101
```

Run   TODO   Problems   Debug   Terminal   Python Packages   Python Console

Python Debugger Extension Available: Cython extension speeds up Python debugging // Install   How does it work (yesterday 7:45 PM)   28:10   LF   UTF-8   4 spac...   Python 3.10 (Oop)

```
19        @classmethod
20        def total_emp(cls):
21            return cls.employee_count - 101
22
23
24   e1 = Employee('John', 'Manager', 10000)
25   e2 = Employee('Mark', 'Team Leader', 8000)
26
27   e1.show_details()
28   print('')
29   e2.show_details()
30   print('Total Employees:', e1.total_emp())
```

Run: SCOops7

```
Eid: e102
Designation: Team Leader
Salary: 8000
Total Employees: 2


Process finished with exit code 0
```

Run   ≡ TODO   ⊘ Problems   ⊕ Debug   ▦ Terminal   ⬢ Python Packages   ⬢ Python Console

Python Debugger Extension Available: Cython extension speeds up Python debugging // Install   How does it work (yesterday 7:45 PM)

Student Challenge

Accessors Mutators

class Customer

name

phoneno

→ get_name( )

→ get_phoneno( )

set_phoneno( )

$\_\_init\_\_(\ )$

```python
class Customer:

    def __init__(self, name, phoneno):
        self.name = name
        self.phneno = phoneno

    def get_name(self):
        return self.name

    def get_phoneno(self):
        return self.phneno

    def set_phoneno(self, ph):
        self.phneno = ph


c1 = Customer('John', 7650984321)

print('Name:', c1.get_name())
print('Phone No:', c1.get_phoneno())

c1.set_phoneno(6789012345)
```
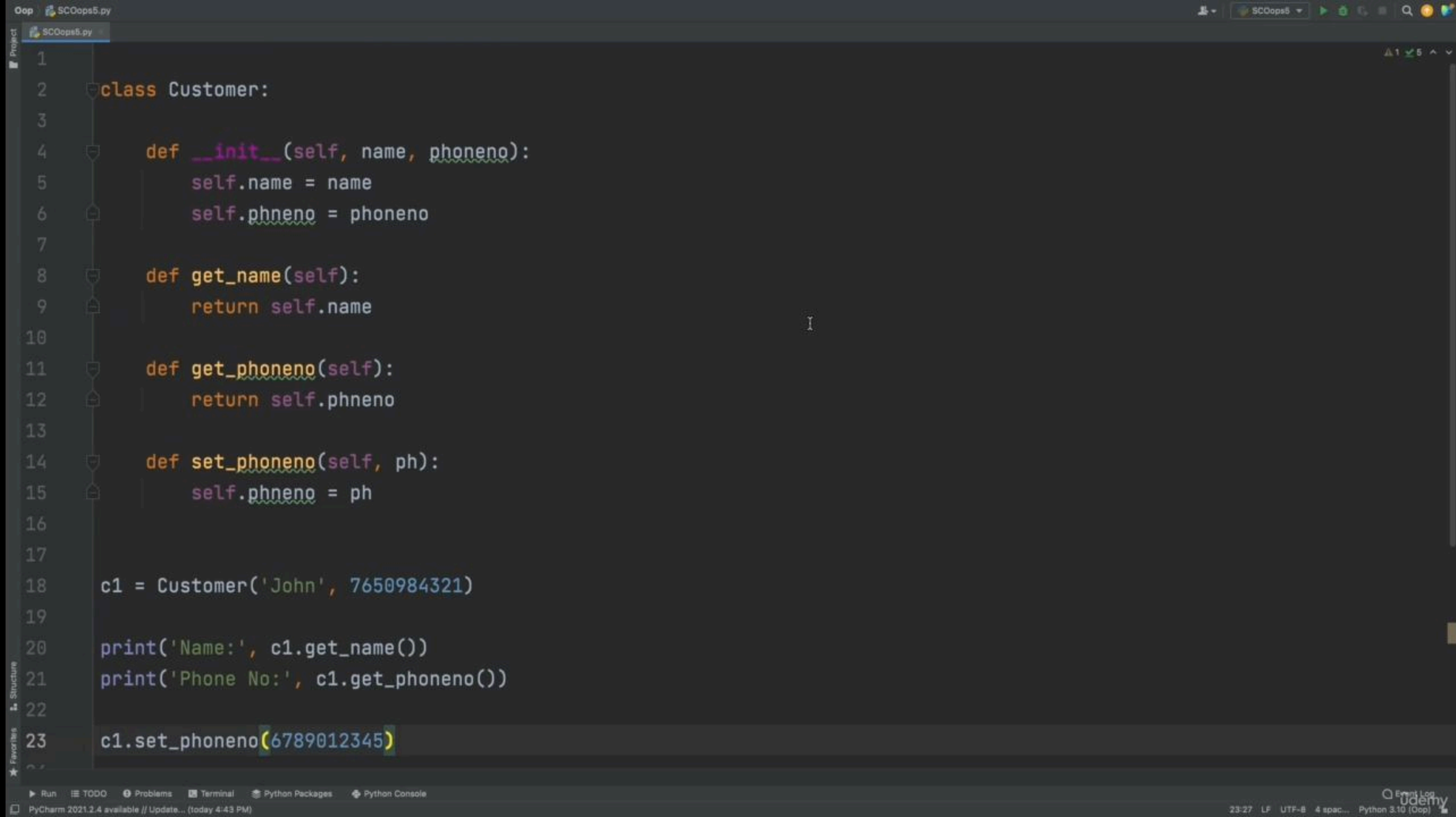
Student Challenge

Accessor and Mutator

class CurrencyConverter

- currency ✓
- rate ✓
- set_currency() ✓
- get_currency() ✓
- set_rate() ✓
- get_rate() ✓

convert(amount)

$amt = 100 \xrightarrow{\text{USD}} \text{INR} \quad 7000$

$rate = 70$

$--init-- (\quad n \underset{\circ}{\overset{\circ}{\cdot}}r \mid$

$=$

$cc = con ----('USD', 70)$

$cc - set\_currency('AUD')$
$cc \cdot set\_rate(50)$

```python
    def __init__(self, name, rate):
        self.currency = name
        self.rate = rate


    def get_currency(self):
        return self.currency


    def get_rate(self):
        return self.rate


    def set_currency(self, name):
        self.currency = name


    def set_rate(self, rate):
        self.rate = rate


    def convert(self, amount):
        return self.currency + 'conversion is ' + str(self.rate * amount)



cc = CurrencyConverter('USD', 70)
print(cc.convert(1))
```

Student Challenge

$\star$ Bank

class Account

- name ✓
- account_number ✓
- balance ✓ — (1000)

- deposit() ✓
- withdraw() ——— 500 ↑
- show_details() ——

→ AccNumber

`MinimumBalanceError` ✓

class MinimumBalanceError ———

pass

class Account:

AccNumber = 1001

def --init-- (self, name, balance)

def deposit (self, amt)

—— withdraw (self, amt)

show_details( )

```python
class MinimumBalanceError(Exception):
    pass


class Account:
    AccNumber = 1001

    def __init__(self, name, balance=1000):

        if balance < 1000:
            raise MinimumBalanceError('Account Cannot be Created')
        self.name = name
        self.balance = balance
        self.account_number = self.AccNumber
        self.AccNumber += 1

    def deposit(self):
        
```

```python
        self.name = name
        self.balance = balance
        self.account_number = self.AccNumber
        self.AccNumber += 1


    def deposit(self, amt):
        self.balance += amt


    def withdraw(self, amt):
        if self.balance - amt < 1000:
            raise MinimumBalanceError('Amount cannot be withdrawn')
        self.balance -= amt


    def show_details(self):
        print('Account Number', self.account_number)
        print('Name', self.name)
        print('Balance',self)
```

*values: object, sep: str | None = ..., end: str | None = ...,
file: SupportsWrite[str] | None = ..., flush: bool = ...

Account · show_details()

Run   TODO   Problems   Terminal   Python Packages   Python Console