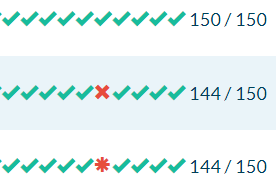
1. Content**.pdf + 09\_Dec\_2023.docx to pdf**
2. **Glasses cleaning**
3. **Make folder as Source root and open Explorer**
4. **Remove unnecessary files and folders**
5. **Submit from correct location**
6. **ZeroDivisionError:**
7. **WATER**
8. **Open excel file**
9. Open tests - Objects and Classes.txt…….
10. **Open exam\_support.py**
11. Pen and paper
12. Google Chrome with quiz tab only + Pen and paper
13. Open
14. **w3schools.com - lists…dict...sets…**

for meal\_name, qty in client.ordered\_meals.items()

return f"Client {client\_phone\_number} successfully canceled his order."



ZeroDivisionError:

return (f"Active Clients: {len(self.clients)}\n"  
 f"Total Income: {total\_clients\_income:.2f}\n"  
 f"Average Client Interest Rate: {avg\_client\_interest\_rate:.2f}\n") error

f"Average Client Interest Rate: {avg\_client\_interest\_rate:.2f} ") error

for attr, value in kwargs.items():  
 setattr(movie, attr, value)

**float problems**

a = 7.55  
b = 240 - 232.45  
print(b) # 7.550000000000011  
print(f"{a:.1f}") # 7.5  
print(f"{b:.1f}") # 7.6  
print(f"{240 - 232.45:.1f}") # 7.6

self.rating: float = 0 vs. at = 0.0

raise (return) Exception(f"No available booth for {number\_of\_people} people!")

if not client:  
 ~~client = Client(client\_phone\_number)  
 self.clients\_list.append(client)~~

self.register\_client(client\_phone\_number) – use the method

"Singer": ["sing high pitch notes", "sing low pitch notes"],

"Singer": ~~"sing high pitch notes and sing low pitch notes",~~

band = self.find\_object(self.bands, "name", band\_name)  
~~musician = self.find\_object(band.members, "name", musician\_name)~~

if not band upper row will raise AttributeError: 'NoneType' object has no attribute 'members'  
if not band: first must be checked that band exists  
 raise Exception(f"{band\_name} isn't a band!")  
musician = self.find\_object(band.members, "name", musician\_name)

return f"{i + 1} astronauts participated in collecting items."

def test\_remove\_from\_cart(self):  
 self.cart.products = {'a': 99, 'b': 12}  
 res = self.cart.remove\_from\_cart('a')  
 expected = "Product a was successfully removed from the cart!"  
  
 self.assertEqual({'b': 12}, self.cart.products) - not empty dictionary check  
 self.assertEqual(expected, res)

self.assertEqual(expected, res) – missing row

04-Exam - 8 April 2023

self.player.wins = ["a", "b"] - error when was submitted with empty list  
expected = ("Tennis Player: asd\n"  
 "Age: 23\n"  
 "Points: 12.5\n"  
 "Tournaments won: a, b")

expected = ("Tennis Player: asd\n"  
 "Age: 23\n"  
 "Points: 12.5\n"  
 "Tournaments won: ")

04-Exam - 8 April 2023

08-Exam - 14 August 2022

self.assertEqual({'a': 0, 'b': 0}, self.store.availability\_in\_store\_by\_book\_titles)

should be checked that if available qty == purchased qty, it’s enough qty to sell

08-Exam - 14 August 2022

05-Exam - 19 December 2022

def test\_drive\_best\_cargo\_with\_cargo(self):  
 self.driver.available\_cargos = {'a': 9, 'b': 10\_000}  
 res = self.driver.drive\_best\_cargo\_offer()  
 expected = "asd is driving 10000 to b."  
  
 self.assertEqual(expected, res)  
 self.assertEqual(10000, self.driver.miles)  
 self.assertEqual(113250, self.driver.earned\_money)

def check\_for\_activities(self, miles):  
 for mile in range(1, miles + 1):  
 self.eat(mile)  
 self.sleep(mile)  
 self.pump\_gas(mile)  
 self.repair\_truck(mile)

05-Exam - 19 December 2022

04\_OOP/Exams/Exams/11-Exam - 11 December 2021/02-Testing/test/test.py

self.assertTrue(self.team.\_\_gt\_\_(other))

self.assertEqual(True, self.team.\_\_gt\_\_(other))

04\_OOP/Exams/Exams/11-Exam - 11 December 2021/02-Testing/test/test.py