Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
class BankAccount:
    def __init__(self, username, user_id, pin):
        self.__username = username
        self.__user_id = user_id
        self.__pin = pin
        self.__balance = 0

    def __deposit(self, amount):
        self.__balance += amount
        print(f"Amount {amount} deposited.")
        self.__display_balance()

    def __withdraw(self, amount):
        if self.__balance >= amount:
            self.__balance -= amount
            print(f"Amount {amount} withdrawn.")
            self.__display_balance()
        else:
            print("Insufficient balance.")

    def __display_balance(self):
        print(f"Current balance: {self.__balance}")

    def transaction(self, pin, amount, transaction_type):
        if pin == self.__pin:
            if transaction_type.lower() == "deposit":
                self.__deposit(amount)
            elif transaction_type.lower() == "withdraw":
                self.__withdraw(amount)
            else:
                print("Invalid transaction type. Please choose 'deposit' or 'withdraw'.")
        else:
            print("Incorrect PIN.")

account = BankAccount("Afifah", "123456", "1234")

account.transaction("1234", 100, "deposit")
account.transaction("1234", 50, "withdraw")
account.transaction("5678", 200, "deposit")  # Incorrect pin
```

input

```
Amount 100 deposited.
Current balance: 100
Amount 50 withdrawn.
Current balance: 50
Incorrect PIN.


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | {} Beautify | Language Python 3

main.py

```python
# Write a program to perform search operation based on filterA dictionary contains house-id
class HouseDatabase:
    def __init__(self):
        self.house_data = {
            "001": {"rent": 5000, "house_type": "1bhk", "furnished": True},
            "002": {"rent": 8000, "house_type": "2bhk", "furnished": False},
            "003": {"rent": 10000, "house_type": "2bhk", "furnished": True},
            "004": {"rent": 12000, "house_type": "3bhk", "furnished": True}
        }

    def search(self, budget, house_type, is_furnished):
        results = []
        for house_id, attributes in self.house_data.items():
            if attributes["house_type"] == house_type and attributes["furnished"] == is_fur
                results.append((house_id, attributes))
        return results

database = HouseDatabase()
budget = 10000
house_type = "2bhk"
is_furnished = True

search_results = database.search(budget, house_type, is_furnished)
print("Search Results:")
if search_results:
    for result in search_results:
        print("House ID:", result[0])
        print("Attributes:", result[1])
else:
    print("No matching houses found.")

```

input

```
Search Results:
House ID: 003
Attributes: {'rent': 10000, 'house_type': '2bhk', 'furnished': True}


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | {} Beautify | Language Python 3

main.py

```python
#Write a program to create list_extenstion class inheriting list class and define a metho
class list_extension(list):
    def indexes(self, element):
        indices = [i for i, x in enumerate(self) if x == element]
        return indices

lst = [121, 115, 89, 54, 121, 110, 33, 92, 44, 67]
obj = list_extension(lst)

element = 121
print(f"Indexes of {element}: {obj.indexes(element)}")

obj.sort()
print("Sorted list:", obj)

obj.reverse()
print("Reversed list:", obj)

obj.append(100)
print("Appended list:", obj)

```

input

```
Indexes of 121: [0, 4]
Sorted list: [33, 44, 54, 67, 89, 92, 110, 115, 121, 121]
Reversed list: [121, 121, 115, 110, 92, 89, 67, 54, 44, 33]
Appended list: [121, 121, 115, 110, 92, 89, 67, 54, 44, 33, 100]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
# Write a program to play "rock-paper-scissor" between computer and player.Hint: random. c
import random

def play_game():
    choices = ["rock", "paper", "scissors"]
    computer_choice = random.choice(choices)
    player_choice = input("Enter your choice (rock/paper/scissors): ").lower()

    print(f"Computer chooses: {computer_choice}")
    print(f"You choose: {player_choice}")

    if player_choice in choices:
        if player_choice == computer_choice:
            print("It's a tie!")
        elif (player_choice == "rock" and computer_choice == "scissors") or \
             (player_choice == "paper" and computer_choice == "rock") or \
             (player_choice == "scissors" and computer_choice == "paper"):
            print("You win!")
        else:
            print("Computer wins!")
    else:
        print("Invalid choice. Please choose rock, paper, or scissors.")

play_game()
```

input

```
Enter your choice (rock/paper/scissors): rock
Computer chooses: paper
You choose: rock
Computer wins!


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
# A dictionary contains strings as values. sort the dictionary by values.
my_dict = {'apple': 'red', 'banana': 'yellow', 'grape': 'purple', 'orange': 'orange'}
sorted_dict = dict(sorted(my_dict.items(), key=lambda item: item[1]))
print("Sorted Dictionary by values:")
for key, value in sorted_dict.items():
    print(key, ":", value)

```

input

```
Sorted Dictionary by values:
orange : orange
grape : purple
apple : red
banana : yellow


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
# write a program to check if sum of any two numbers in list make the number taken from cor
def find_sum_pair(numbers, target):
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if numbers[i] + numbers[j] == target:
                return numbers[i], numbers[j]
    return None
numbers_list = [23, 45, 72, 19, 35]
target_number = int(input("Enter a number: "))
pair = find_sum_pair(numbers_list, target_number)
if pair:
    print(f"Sum {pair} from the list makes {target_number}.")
else:
    print("No pair of numbers in the list makes the entered number.")

```

input

```
Enter a number: 117
Sum (45, 72) from the list makes 117.


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
# A over speed ticket is raised above 70 kmph.A list contains data collected at different c
def calculate_fine(speed_readings):
    total_fine = 0
    num_tickets = 0
    for speed in speed_readings:
        if speed > 70:
            num_tickets += 1
            fine = 100 + (num_tickets - 1) * 50
            total_fine += fine
    return num_tickets, total_fine
speed_readings = [92, 44, 55, 77, 82]
total_tickets, total_fine = calculate_fine(speed_readings)
print("Total tickets raised:", total_tickets)
print("Total fine charged:", total_fine, "rupees")

```

input

```
Total tickets raised: 3
Total fine charged: 450 rupees


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
# write a function to take a list as parameter and return a list by sorting list where eve
def sort_even_odd(input_list):
    def custom_sort_key(num):
        return (num % 2, num)
    sorted_list = sorted(input_list, key=custom_sort_key)
    return sorted_list
actual_list = [9, 2, 3, 7, 1, 4, 5, 8, 0, 6]
sorted_list = sort_even_odd(actual_list)
print("Sorted list:", sorted_list)

```

input

```
Sorted list: [0, 2, 4, 6, 8, 1, 3, 5, 7, 9]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
# write a program to take a number from user and check whether it is part of Fibonacci sequ
def is_fibonacci(number):
    def is_perfect_square(num):
        return int(num**0.5)**2 == num
    return is_perfect_square(5 * number * number + 4) or is_perfect_square(5 * number * num
user_number = int(input("Enter a number: "))
if is_fibonacci(user_number):
    print(f"{user_number} is part of the Fibonacci sequence.")
else:
    print(f"{user_number} is not part of the Fibonacci sequence.")

```

input

```
Enter a number: 7329
7329 is not part of the Fibonacci sequence.


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
# Write a program to perform exception handling and raise exception with message - 'N
def perform_operation(number):
    try:
        result = 10 / number
    except ZeroDivisionError:
        print("Error: Cannot divide by zero!")
    else:
        if number != 0:
            print("Result:", result)
        else:
            raise Exception("No Error Found")
try:
    perform_operation(2)
except Exception as e:
    print(e)

```

input

```
Result: 5.0


...Program finished with exit code 0
Press ENTER to exit console.
```