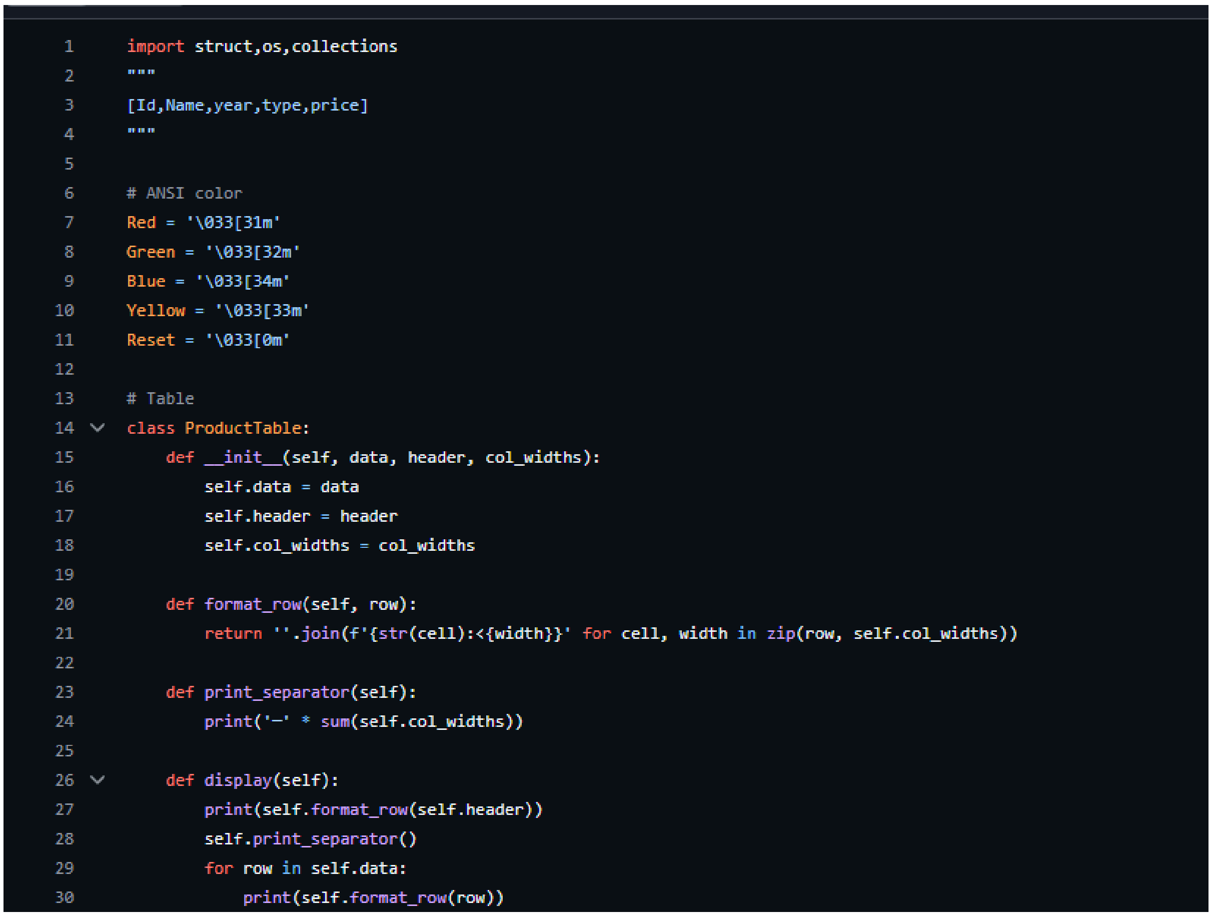
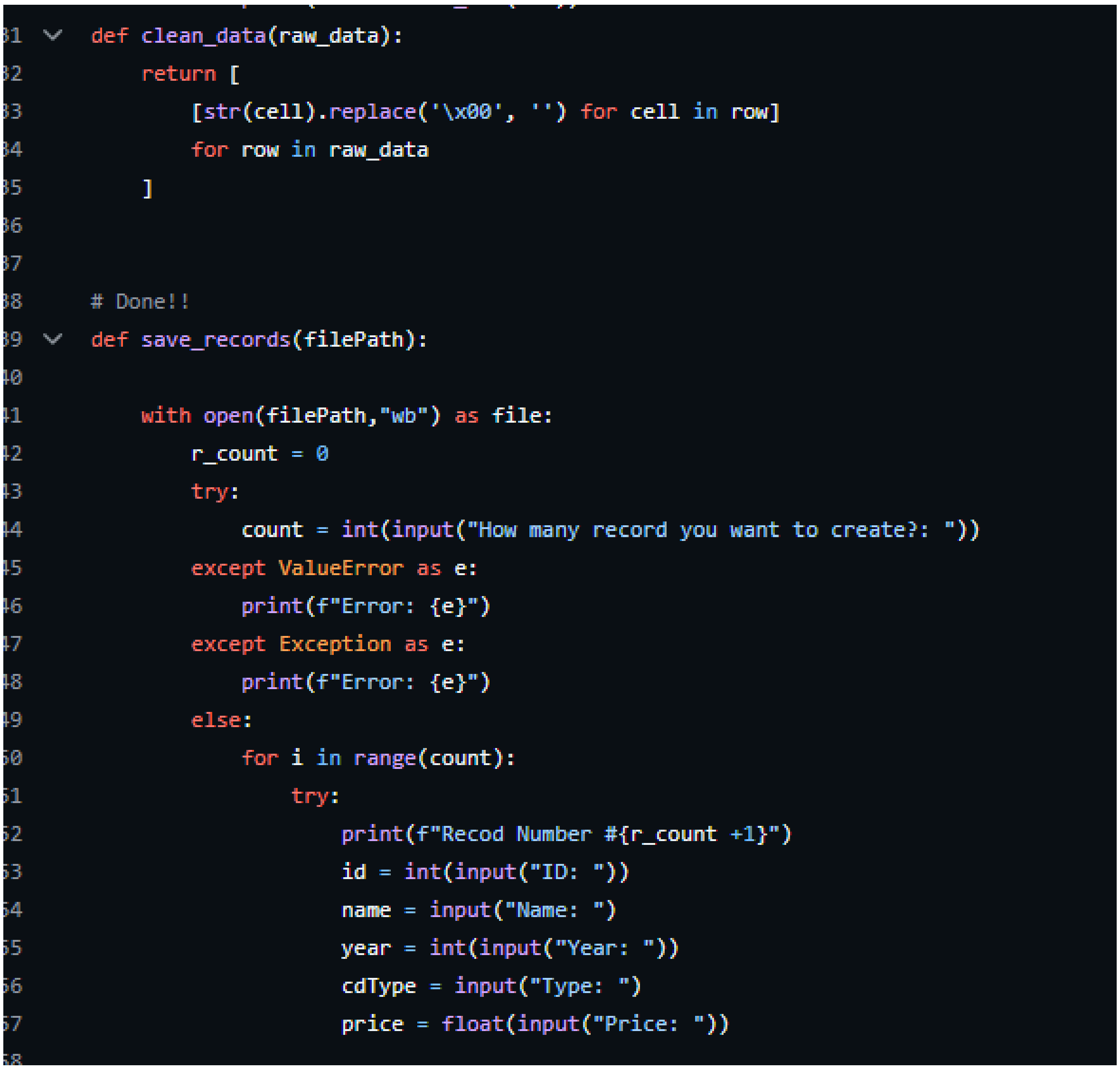
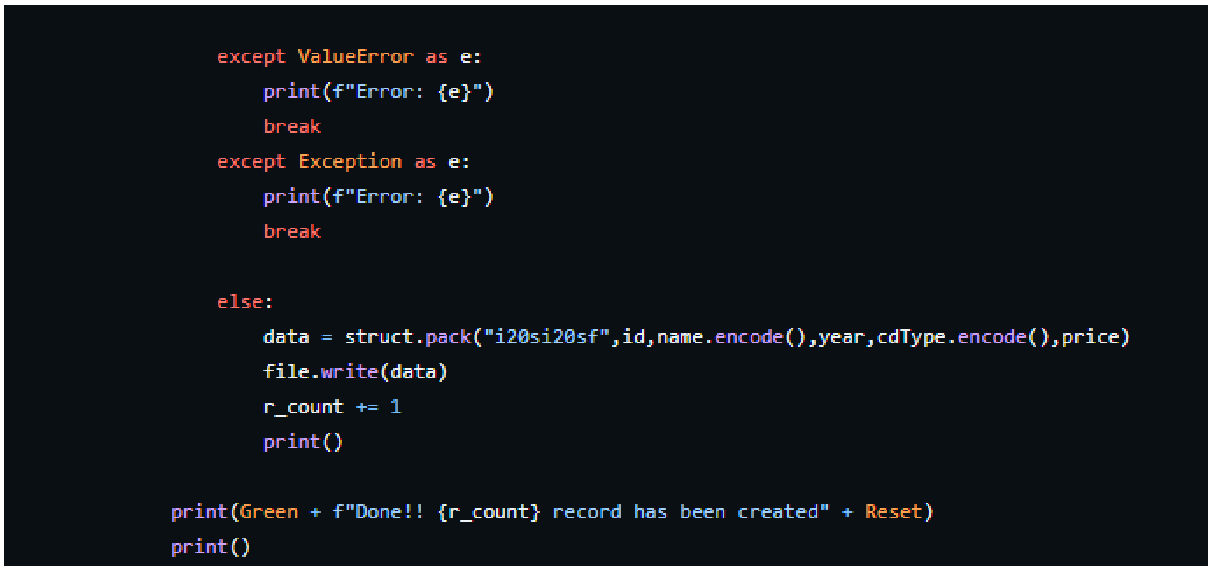


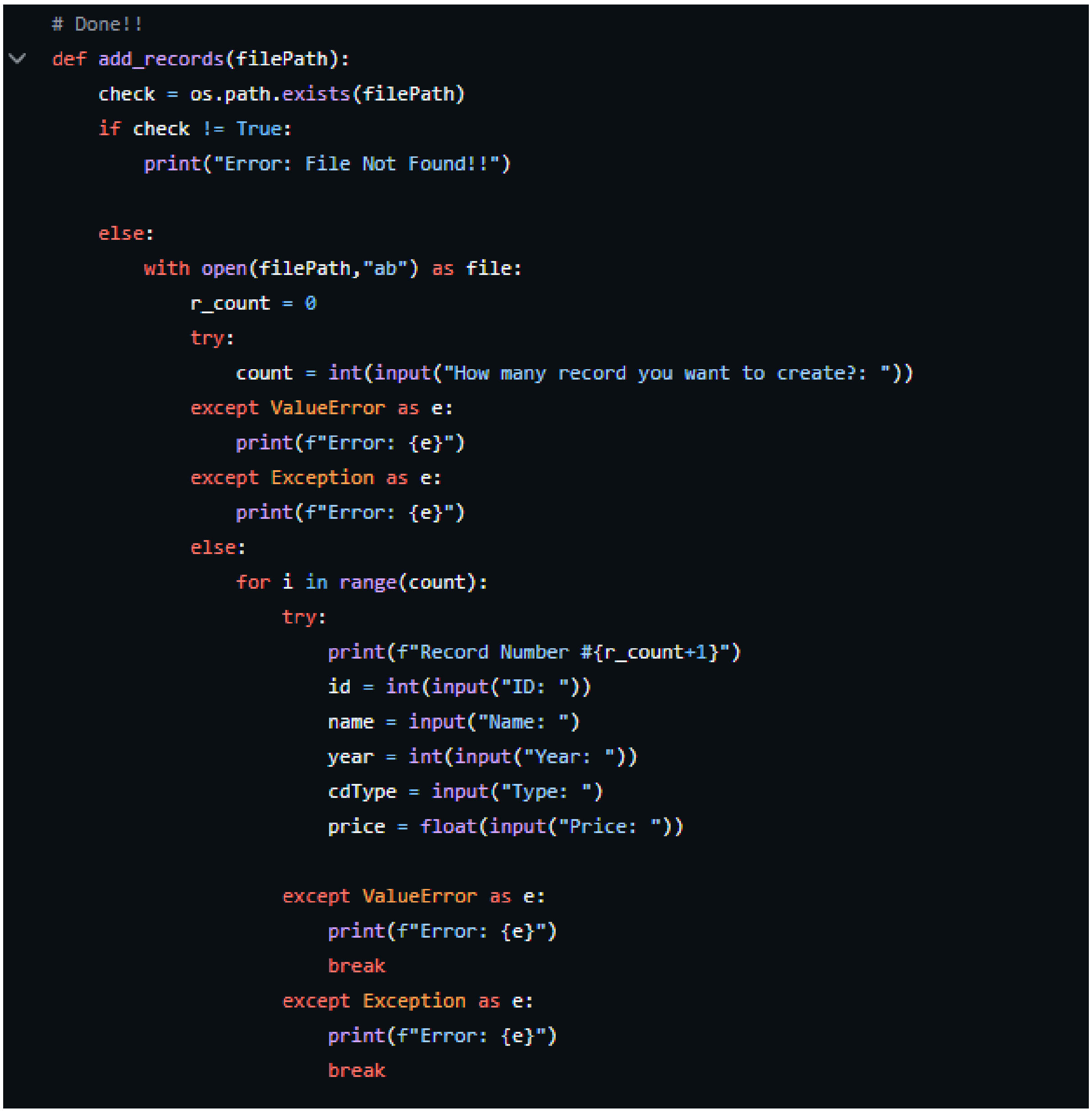
```python
import struct,os,collections
"""
[Id,Name,year,type,price]
"""

# ANSI color
Red = '\033[31m'
Green = '\033[32m'
Blue = '\033[34m'
Yellow = '\033[33m'
Reset = '\033[0m'

# Table
class ProductTable:
    def __init__(self, data, header, col_widths):
        self.data = data
        self.header = header
        self.col_widths = col_widths

    def format_row(self, row):
        return ''.join(f'{str(cell):<{width}}' for cell, width in zip(row, self.col_widths))

    def print_separator(self):
        print('-' * sum(self.col_widths))

    def display(self):
        print(self.format_row(self.header))
        self.print_separator()
        for row in self.data:
            print(self.format_row(row))
```



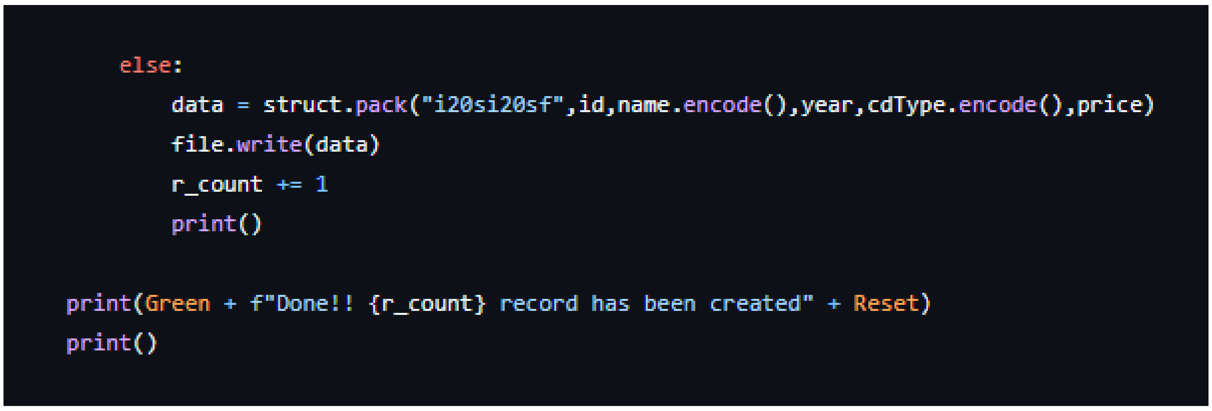
```python
def clean_data(raw_data):
    return [
        [str(cell).replace('\x00', '') for cell in row]
        for row in raw_data
    ]


# Done!!
def save_records(filePath):

    with open(filePath,"wb") as file:
        r_count = 0
        try:
            count = int(input("How many record you want to create?: "))
        except ValueError as e:
            print(f"Error: {e}")
        except Exception as e:
            print(f"Error: {e}")
        else:
            for i in range(count):
                try:
                    print(f"Recod Number #{r_count +1}")
                    id = int(input("ID: "))
                    name = input("Name: ")
                    year = int(input("Year: "))
                    cdType = input("Type: ")
                    price = float(input("Price: "))
```



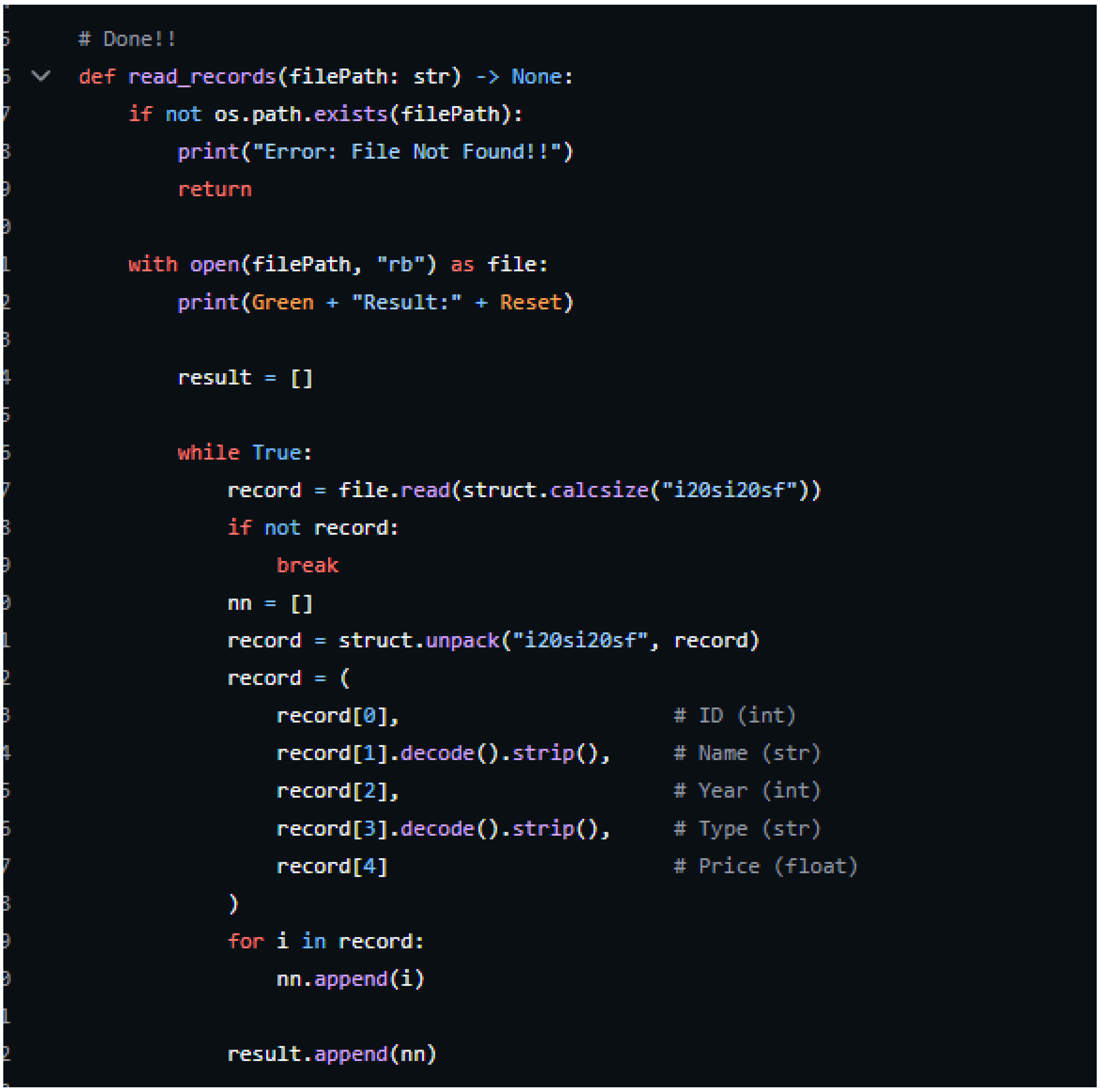
```python
                except ValueError as e:
                    print(f"Error: {e}")
                    break
                except Exception as e:
                    print(f"Error: {e}")
                    break

                else:
                    data = struct.pack("i20si20sf",id,name.encode(),year,cdType.encode(),price)
                    file.write(data)
                    r_count += 1
                    print()

            print(Green + f"Done!! {r_count} record has been created" + Reset)
            print()
```



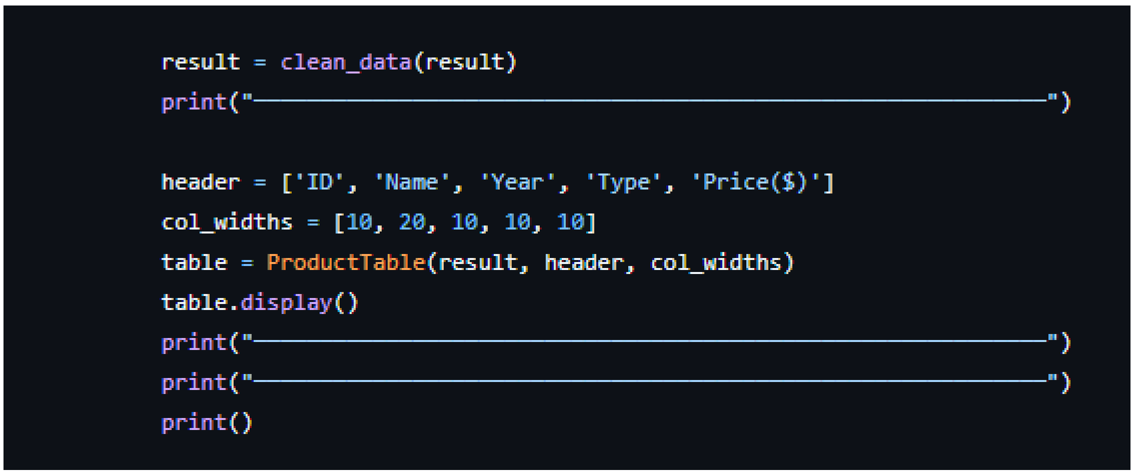
```python
# Done!!
def add_records(filePath):
    check = os.path.exists(filePath)
    if check != True:
        print("Error: File Not Found!!")

    else:
        with open(filePath,"ab") as file:
            r_count = 0
            try:
                count = int(input("How many record you want to create?: "))
            except ValueError as e:
                print(f"Error: {e}")
            except Exception as e:
                print(f"Error: {e}")
            else:
                for i in range(count):
                    try:
                        print(f"Record Number #{r_count+1}")
                        id = int(input("ID: "))
                        name = input("Name: ")
                        year = int(input("Year: "))
                        cdType = input("Type: ")
                        price = float(input("Price: "))

                    except ValueError as e:
                        print(f"Error: {e}")
                        break
                    except Exception as e:
                        print(f"Error: {e}")
                        break
```



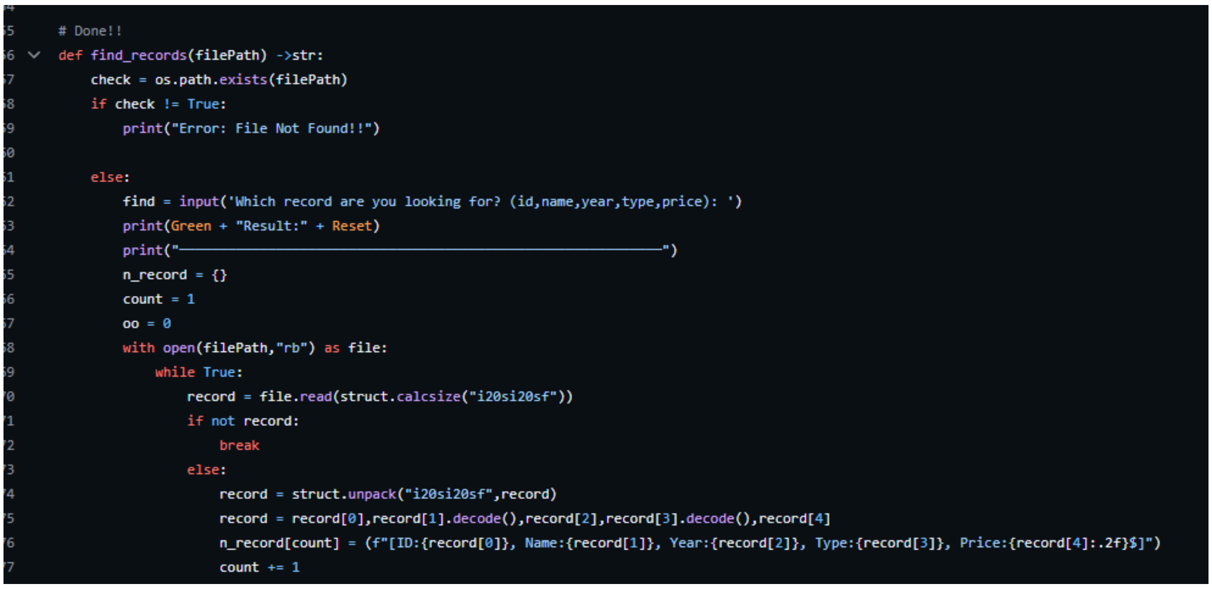
```python
    else:
        data = struct.pack("i20si20sf",id,name.encode(),year,cdType.encode(),price)
        file.write(data)
        r_count += 1
        print()

print(Green + f"Done!! {r_count} record has been created" + Reset)
print()
```



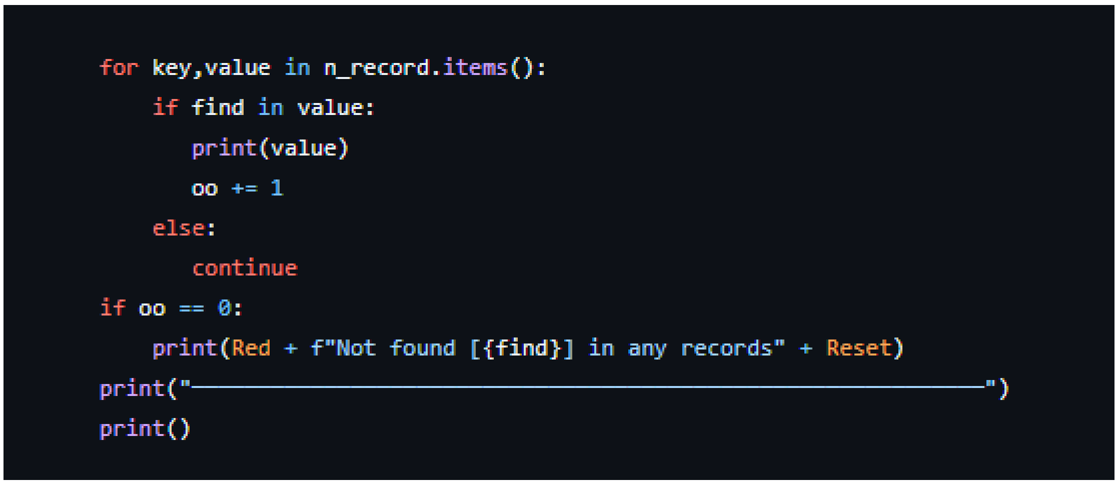
```python
# Done!!
def read_records(filePath: str) -> None:
    if not os.path.exists(filePath):
        print("Error: File Not Found!!")
        return

    with open(filePath, "rb") as file:
        print(Green + "Result:" + Reset)

        result = []

        while True:
            record = file.read(struct.calcsize("i20si20sf"))
            if not record:
                break
            nn = []
            record = struct.unpack("i20si20sf", record)
            record = (
                record[0],                      # ID (int)
                record[1].decode().strip(),     # Name (str)
                record[2],                      # Year (int)
                record[3].decode().strip(),     # Type (str)
                record[4]                       # Price (float)
            )
            for i in record:
                nn.append(i)

            result.append(nn)
```



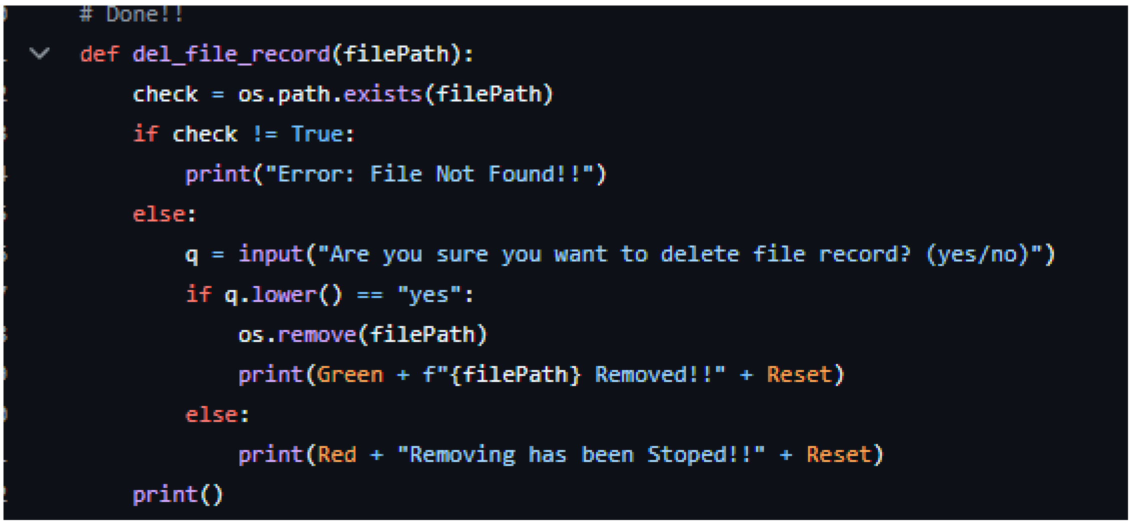
```python
        result = clean_data(result)
        print("────────────────────────────────────────────────────────────")

        header = ['ID', 'Name', 'Year', 'Type', 'Price($)']
        col_widths = [10, 20, 10, 10, 10]
        table = ProductTable(result, header, col_widths)
        table.display()
        print("────────────────────────────────────────────────────────────")
        print("────────────────────────────────────────────────────────────")
        print()
```



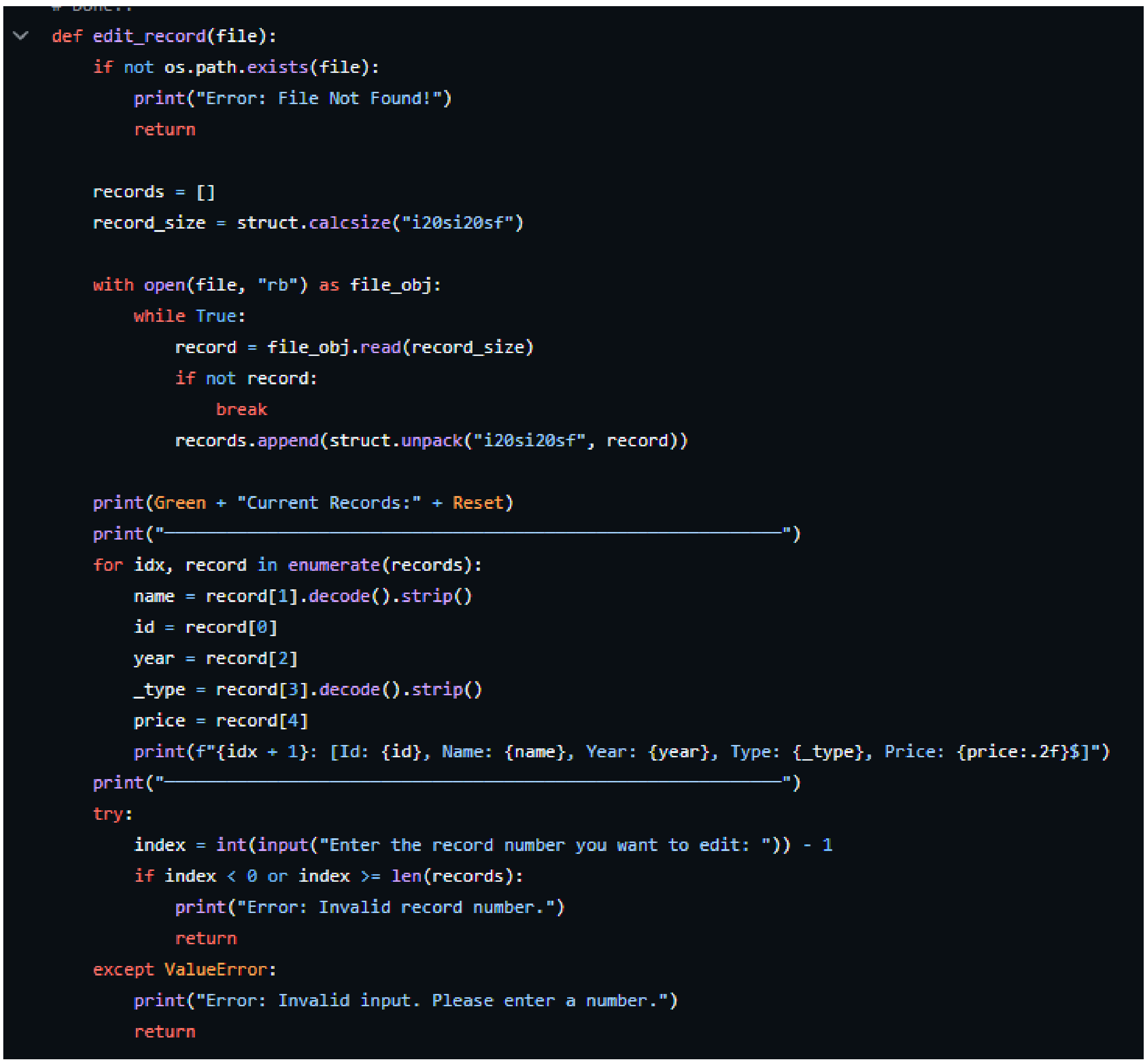
```python
# Done!!
def find_records(filePath) ->str:
    check = os.path.exists(filePath)
    if check != True:
        print("Error: File Not Found!!")

    else:
        find = input('Which record are you looking for? (id,name,year,type,price): ')
        print(Green + "Result:" + Reset)
        print("────────────────────────────────────────────────────────────")
        n_record = {}
        count = 1
        oo = 0
        with open(filePath,"rb") as file:
            while True:
                record = file.read(struct.calcsize("i20si20sf"))
                if not record:
                    break
                else:
                    record = struct.unpack("i20si20sf",record)
                    record = record[0],record[1].decode(),record[2],record[3].decode(),record[4]
                    n_record[count] = (f"[ID:{record[0]}, Name:{record[1]}, Year:{record[2]}, Type:{record[3]}, Price:{record[4]:.2f}$]")
                    count += 1
```



```python
for key,value in n_record.items():
    if find in value:
        print(value)
        oo += 1
    else:
        continue
if oo == 0:
    print(Red + f"Not found [{find}] in any records" + Reset)
print("────────────────────────────────────────────────────────────")
print()
```

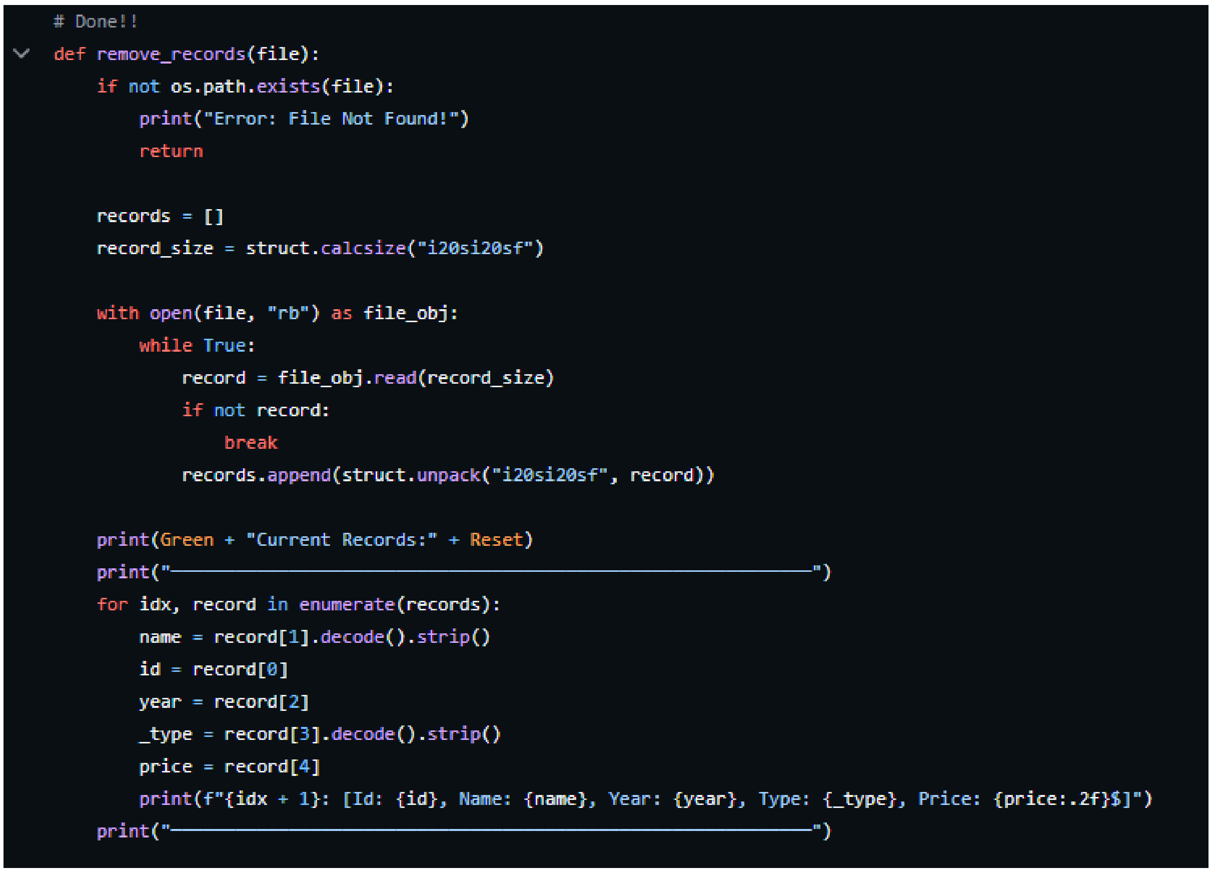


```python
    # Done!!
    def del_file_record(filePath):
        check = os.path.exists(filePath)
        if check != True:
            print("Error: File Not Found!!")
        else:
            q = input("Are you sure you want to delete file record? (yes/no)")
            if q.lower() == "yes":
                os.remove(filePath)
                print(Green + f"{filePath} Removed!!" + Reset)
            else:
                print(Red + "Removing has been Stoped!!" + Reset)
        print()
```

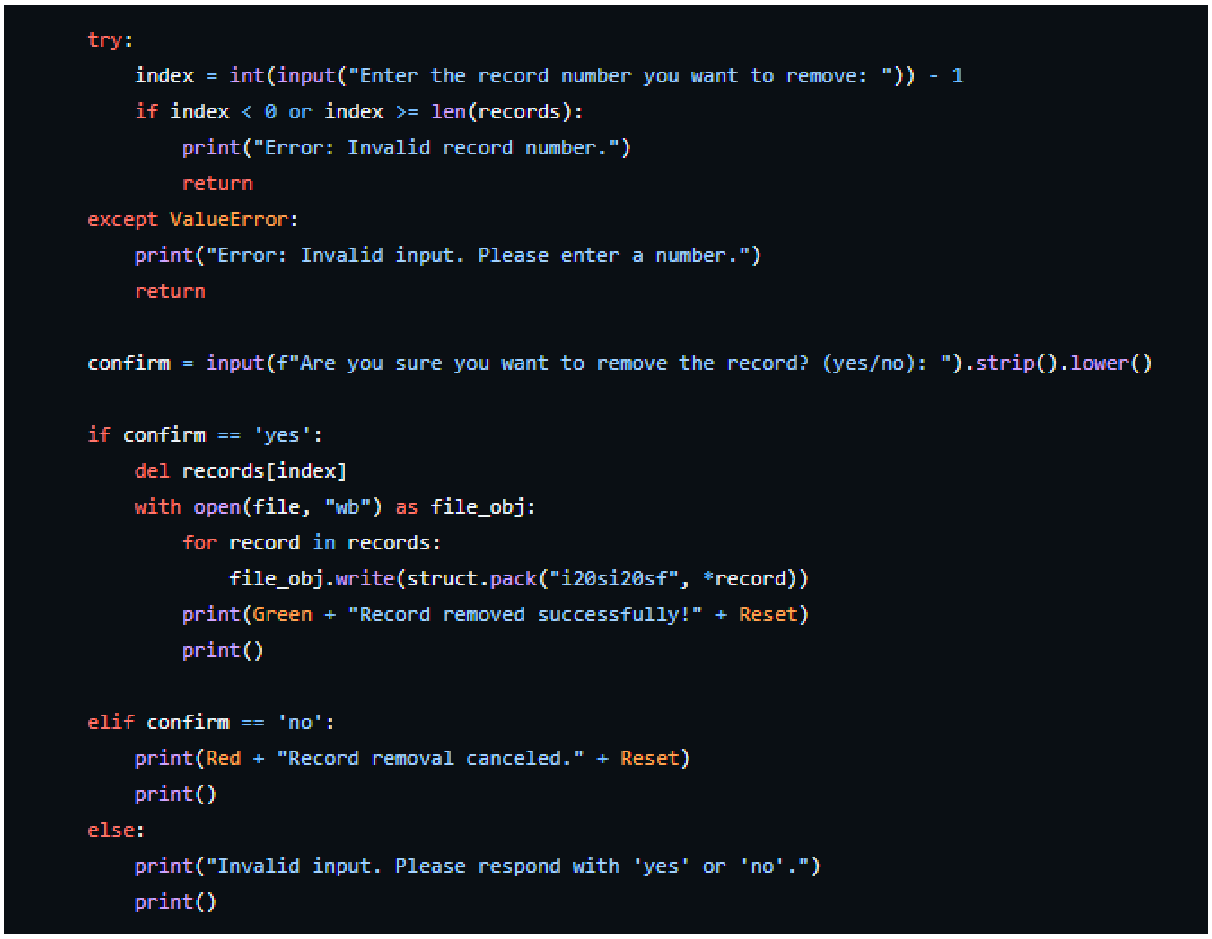


```python
def edit_record(file):
    if not os.path.exists(file):
        print("Error: File Not Found!")
        return

    records = []
    record_size = struct.calcsize("i20si20sf")

    with open(file, "rb") as file_obj:
        while True:
            record = file_obj.read(record_size)
            if not record:
                break
            records.append(struct.unpack("i20si20sf", record))

    print(Green + "Current Records:" + Reset)
    print("────────────────────────────────────────────────────────────")
    for idx, record in enumerate(records):
        name = record[1].decode().strip()
        id = record[0]
        year = record[2]
        _type = record[3].decode().strip()
        price = record[4]
        print(f"{idx + 1}: [Id: {id}, Name: {name}, Year: {year}, Type: {_type}, Price: {price:.2f}$]")
    print("────────────────────────────────────────────────────────────")
    try:
        index = int(input("Enter the record number you want to edit: ")) - 1
        if index < 0 or index >= len(records):
            print("Error: Invalid record number.")
            return
    except ValueError:
        print("Error: Invalid input. Please enter a number.")
        return
```

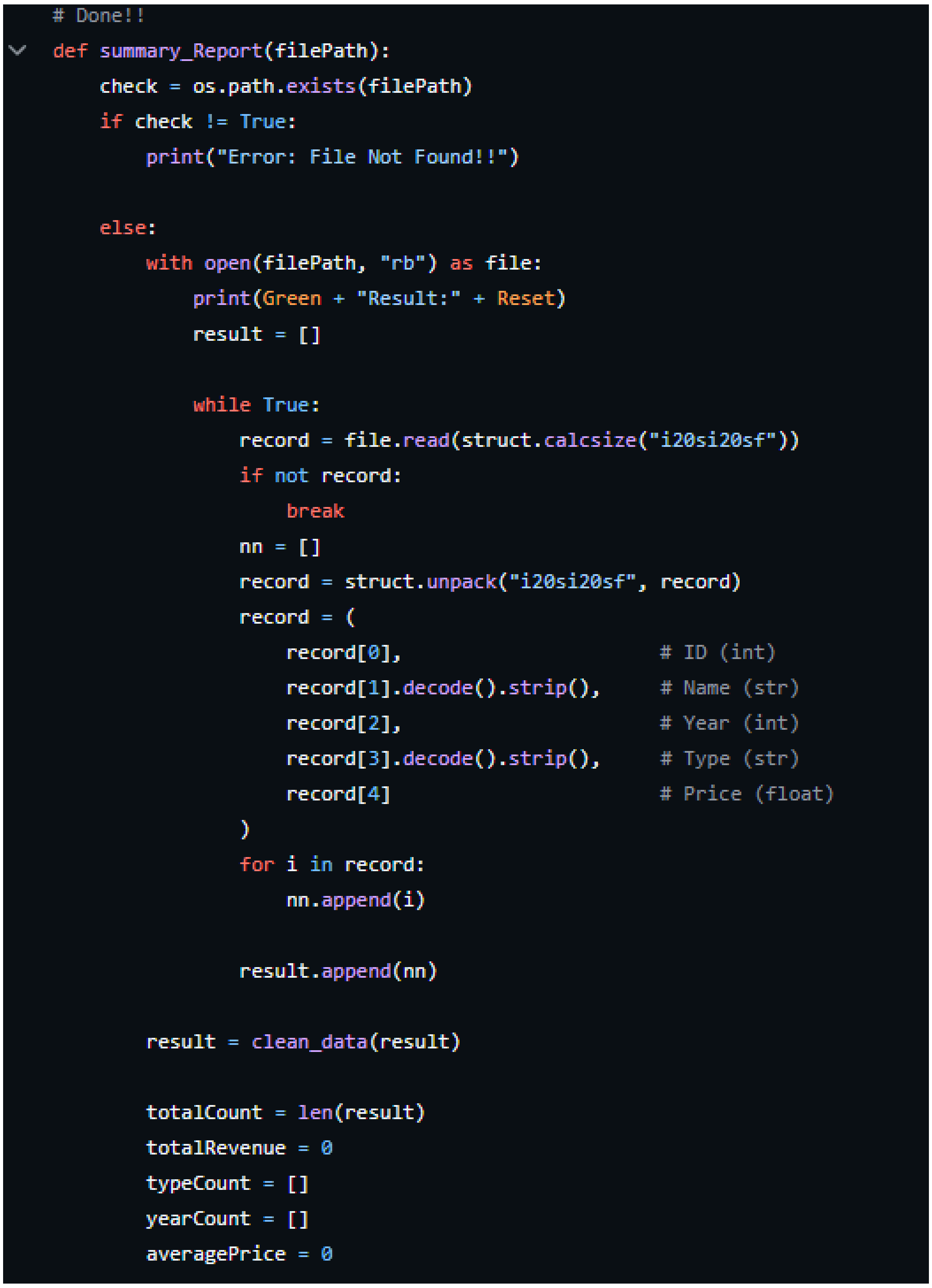
```python
record_to_edit = records[index]
id = record_to_edit[0]
name = record_to_edit[1].decode().strip()
year = record_to_edit[2]
_type = record_to_edit[3].decode().strip()
price = record_to_edit[4]

print(f"\nEditing record {index + 1}: [Id: {id}, Name: {name}, Year: {year}, Type: {_type}, Price: {price:.2f}$]")

new_id = input(f"New ID (leave blank to not change): ")
new_name = input(f"New Name (leave blank to not change): ")
new_year = input(f"New Year (leave blank to not change): ")
new_type = input(f"New Type (leave blank to not change): ")
new_price = input(f"New Price (leave blank to not change): ")

new_id = int(new_id) if new_id else id
new_name = new_name.encode() if new_name.strip() else record_to_edit[1]
new_year = int(new_year) if new_year else year
new_type = new_type.encode() if new_type.strip() else record_to_edit[3]
new_price = float(new_price) if new_price else price


records[index] = (new_id, new_name, new_year, new_type, new_price)

with open(file, "wb") as file_obj:
    for record in records:
        file_obj.write(struct.pack("i20si20sf", *record))

print(Green + "Record updated successfully!" + Reset)
print()
```



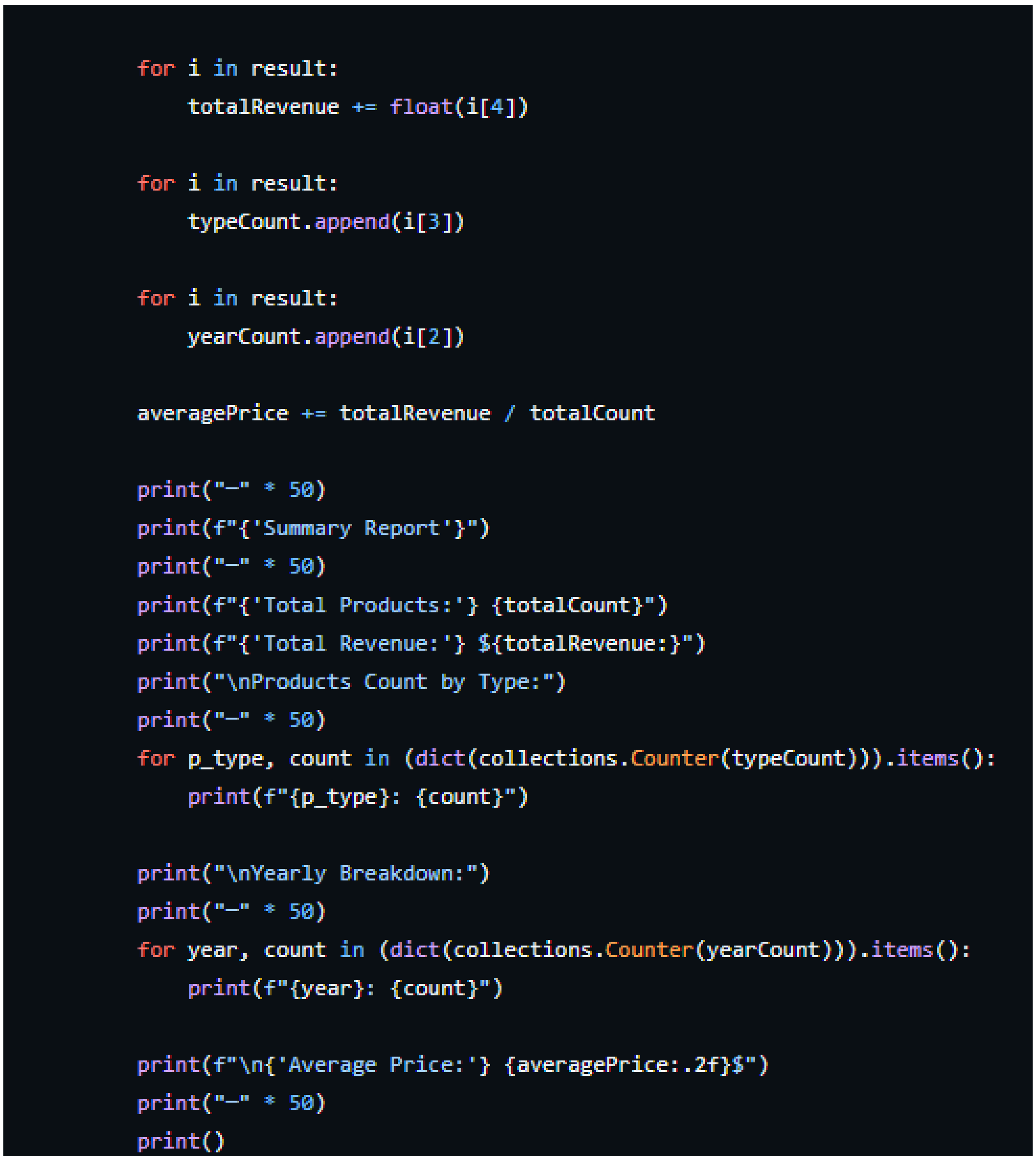
```python
# Done!!
def remove_records(file):
    if not os.path.exists(file):
        print("Error: File Not Found!")
        return

    records = []
    record_size = struct.calcsize("i20si20sf")

    with open(file, "rb") as file_obj:
        while True:
            record = file_obj.read(record_size)
            if not record:
                break
            records.append(struct.unpack("i20si20sf", record))

    print(Green + "Current Records:" + Reset)
    print("————————————————————————————————————————————————————————————")
    for idx, record in enumerate(records):
        name = record[1].decode().strip()
        id = record[0]
        year = record[2]
        _type = record[3].decode().strip()
        price = record[4]
        print(f"{idx + 1}: [Id: {id}, Name: {name}, Year: {year}, Type: {_type}, Price: {price:.2f}$]")
    print("————————————————————————————————————————————————————————————")
```



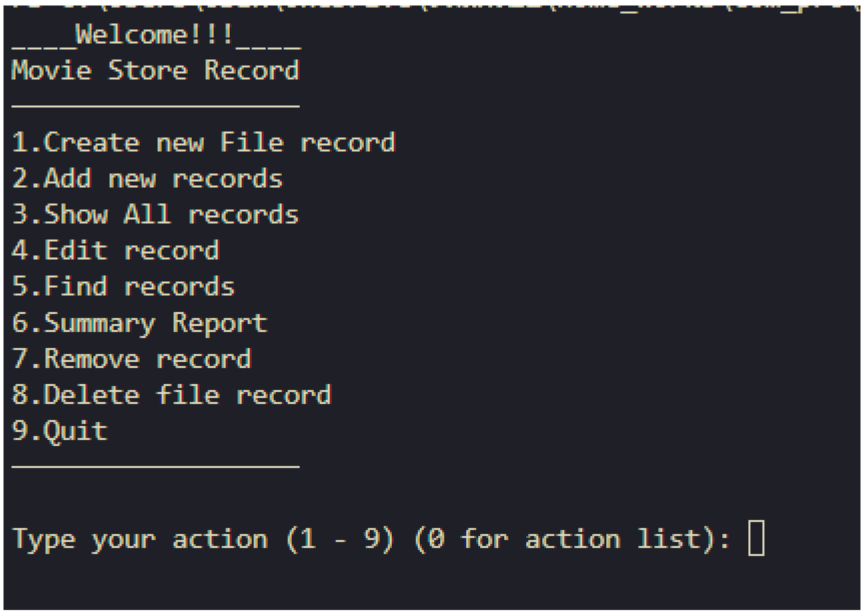
```python
    try:
        index = int(input("Enter the record number you want to remove: ")) - 1
        if index < 0 or index >= len(records):
            print("Error: Invalid record number.")
            return
    except ValueError:
        print("Error: Invalid input. Please enter a number.")
        return

    confirm = input(f"Are you sure you want to remove the record? (yes/no): ").strip().lower()

    if confirm == 'yes':
        del records[index]
        with open(file, "wb") as file_obj:
            for record in records:
                file_obj.write(struct.pack("i20si20sf", *record))
            print(Green + "Record removed successfully!" + Reset)
            print()

    elif confirm == 'no':
        print(Red + "Record removal canceled." + Reset)
        print()
    else:
        print("Invalid input. Please respond with 'yes' or 'no'.")
        print()
```



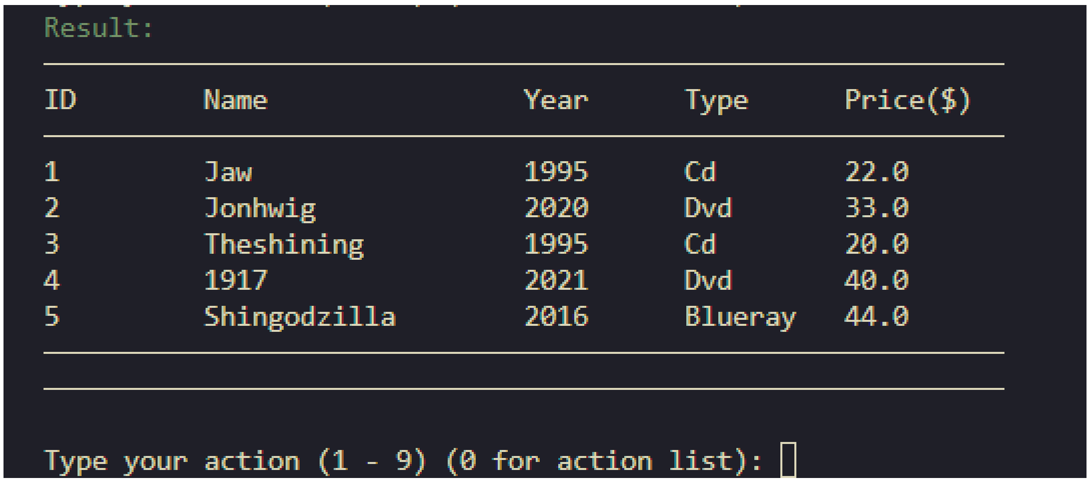
```
# Done!!
def summary_Report(filePath):
    check = os.path.exists(filePath)
    if check != True:
        print("Error: File Not Found!!")

    else:
        with open(filePath, "rb") as file:
            print(Green + "Result:" + Reset)
            result = []

            while True:
                record = file.read(struct.calcsize("i20si20sf"))
                if not record:
                    break
                nn = []
                record = struct.unpack("i20si20sf", record)
                record = (
                    record[0],                      # ID (int)
                    record[1].decode().strip(),     # Name (str)
                    record[2],                      # Year (int)
                    record[3].decode().strip(),     # Type (str)
                    record[4]                       # Price (float)
                )
                for i in record:
                    nn.append(i)

                result.append(nn)

        result = clean_data(result)

        totalCount = len(result)
        totalRevenue = 0
        typeCount = []
        yearCount = []
        averagePrice = 0
```



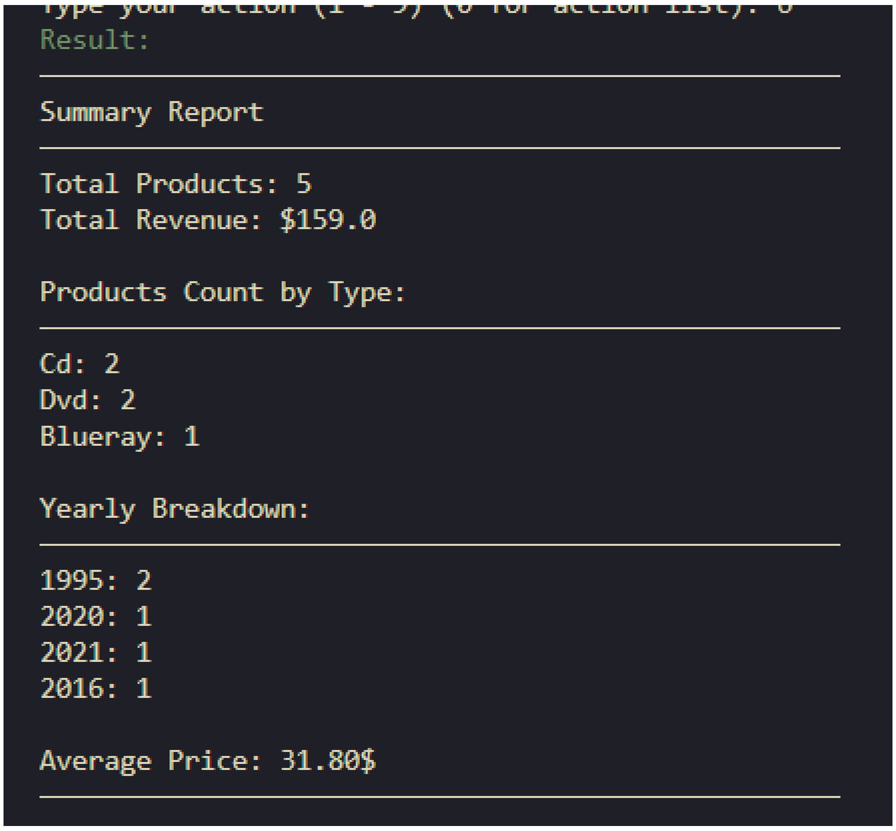
```
        for i in result:
            totalRevenue += float(i[4])

        for i in result:
            typeCount.append(i[3])

        for i in result:
            yearCount.append(i[2])

        averagePrice += totalRevenue / totalCount

        print("-" * 50)
        print(f"{'Summary Report'}")
        print("-" * 50)
        print(f"{'Total Products:'} {totalCount}")
        print(f"{'Total Revenue:'} ${totalRevenue:}")
        print("\nProducts Count by Type:")
        print("-" * 50)
        for p_type, count in (dict(collections.Counter(typeCount))).items():
            print(f"{p_type}: {count}")

        print("\nYearly Breakdown:")
        print("-" * 50)
        for year, count in (dict(collections.Counter(yearCount))).items():
            print(f"{year}: {count}")

        print(f"\n{'Average Price:'} {averagePrice:.2f}$")
        print("-" * 50)
        print()
```



```
____Welcome!!!____
Movie Store Record
__________________
1.Create new File record
2.Add new records
3.Show All records
4.Edit record
5.Find records
6.Summary Report
7.Remove record
8.Delete file record
9.Quit
__________________

Type your action (1 - 9) (0 for action list):
```



```
Result:

ID          Name                Year      Type      Price($)

1           Jaw                 1995      Cd        22.0
2           Jonhwig             2020      Dvd       33.0
3           Theshining          1995      Cd        20.0
4           1917                2021      Dvd       40.0
5           Shingodzilla        2016      Blueray   44.0



Type your action (1 - 9) (0 for action list):
```



```
Type your action (1 - 9) (0 for action list): 0
Result:

Summary Report

Total Products: 5
Total Revenue: $159.0

Products Count by Type:

Cd: 2
Dvd: 2
Blueray: 1

Yearly Breakdown:

1995: 2
2020: 1
2021: 1
2016: 1

Average Price: 31.80$
```