МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет електроніки та комп’ютерних технологій

ЗВІТ

про виконання лабораторної роботи №1

з курсу “Функціональне програмування”

«Основи функціонального програмування у Python»

Виконав:

Студент 2 курсу

групи ФеП-23

Чепара Станіслав

Перевірив: Доцент Франів В.А.

Львів-2025

Мета роботи: розібратися з поняттями чистої функції та референтної прозорості, навчитися знаходити й відокремлювати побічні ефекти, переписати імперативний код у функціональній парадигмі, використати typing.Callable для параметризації обчислень.

Хід роботи:

1. **Переписаний код у функціональному стилі:**

   App.py

app.py > ...

```python
from __future__ import annotations

import argparse

from core import Order, process_orders_pure


def sample_orders() -> list[Order]:
    return [
        {
            "id": 1,
            "paid": True,
            "items": [{"price": 50.0, "qty": 2}, {"price": 20.0, "qty": 1}],
        },
        {"id": 2, "paid": False, "items": [{"price": 200.0, "qty": 1}]},
        {"id": 3, "paid": True, "items": [{"price": 30.0, "qty": 3}]},
    ]


def main() -> None:
    parser = argparse.ArgumentParser()
    parser.add_argument("--min-total", type=float, default=100.0)
    parser.add_argument("--discount", type=float, default=0.1)
    parser.add_argument("--tax", type=float, default=0.2)
    args = parser.parse_args()

    result = process_orders_pure(
        sample_orders(),
        min_total=args.min_total,
        discount=args.discount,
        tax_rate=args.tax,
    )

    for o in result["orders"]:
        print(f"Processed id={o['id']} total={o['total']:.2f}")
    print(f"Revenue: {result['revenue']:.2f} Count: {result['count']}")


if __name__ == "__main__":
    main()
```

Core.py

core.py > apply_discount_rate

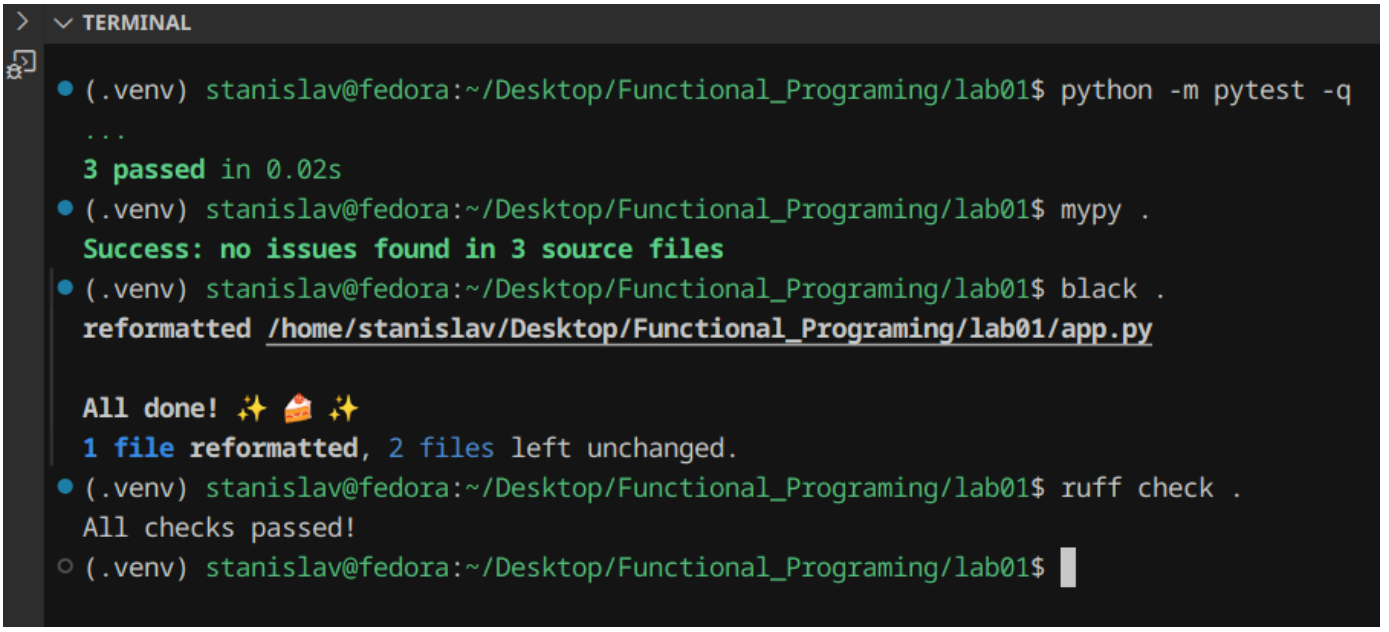
```python
from __future__ import annotations

from collections.abc import Callable
from typing import TypedDict

class Item(TypedDict):
    price: float
    qty: int


class Order(TypedDict):
    id: int
    items: list[Item]
    paid: bool


class ProcessedOrder(TypedDict):
    id: int
    total: float


class Result(TypedDict):
    orders: list[ProcessedOrder]
    revenue: float
    count: int


def subtotal(order: Order) -> float:
    """Сума по товарах (без знижок і податків)."""
    return sum(item["price"] * item["qty"] for item in order["items"])


def accept_min_total(min_total: float) -> Callable[[float], bool]:
    """Приймати замовлення з subtotal ≥ min_total."""
    return lambda s: s >= min_total


def apply_discount_rate(rate: float) -> Callable[[float], float]:
    """Знижка rate (0.1 = 10%)."""

    def _apply(s: float) -> float:
        return s * (1 - rate)

    return _apply
```

core.py > apply_discount_rate

```python
def apply_tax_rate(rate: float) -> Callable[[float], float]:
    """Податок rate (0.2 = 20%)."""

    def _apply(a: float) -> float:
        return a * (1 + rate)

    return _apply



def make_processor(
    accept: Callable[[float], bool],
    apply_discount: Callable[[float], float],
    apply_tax: Callable[[float], float],
) -> Callable[[list[Order]], Result]:
    """
    Повертає чисту функцію process(orders) -> Result.

    Ланцюжок:
      1) тільки оплачені замовлення;
      2) subtotal;
      3) accept(subtotal);
      4) знижка → податок;
      5) підсумок revenue і count.
    """

    def process(orders: list[Order]) -> Result:
        processed: list[ProcessedOrder] = []

        for o in orders:
            if not o.get("paid"):
                continue

            s = subtotal(o)
            if not accept(s):
                continue

            total = apply_tax(apply_discount(s))
            processed.append({"id": o["id"], "total": total})

        revenue = sum(p["total"] for p in processed)
        return {"orders": processed, "revenue": revenue, "count": len(processed)}

    return process
```

```python
def process_orders_pure(
    orders: list[Order],
    *,
    min_total: float,
    discount: float,
    tax_rate: float,
) -> Result:
    """Готова конфігурація конвеєра під методичку."""
    processor = make_processor(
        accept=accept_min_total(min_total),
        apply_discount=apply_discount_rate(discount),
        apply_tax=apply_tax_rate(tax_rate),
    )
    return processor(orders)
```

2.Вивід в консоль тестів



Висновок: у роботі закріплено поняття чистих функцій та референтної прозорості, відпрацьовано ізоляцію побічних ефектів, переписування імперативного коду у функціональному стилі та використання typing.Callable для параметризації обчислень.