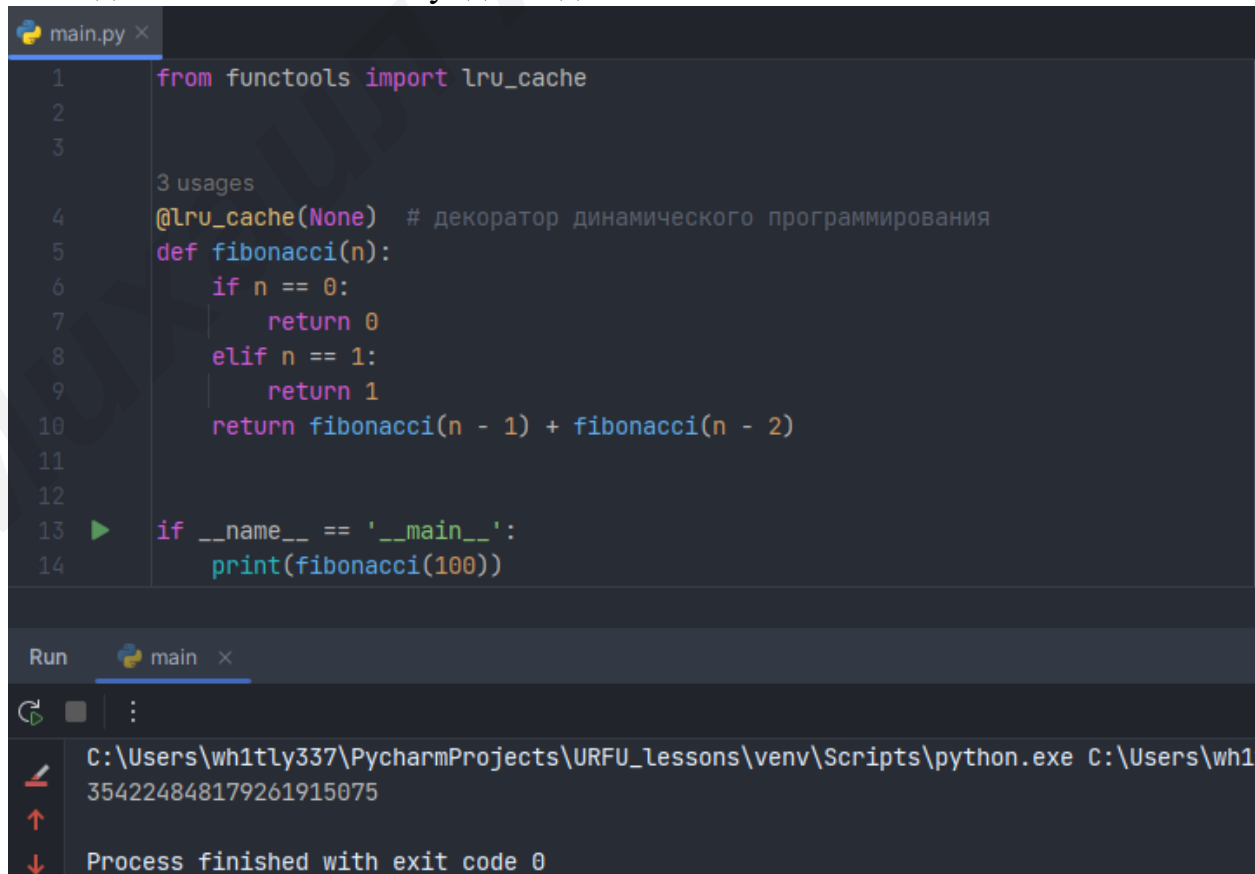


Лабораторная работа 10

ТЕМА 10. Декораторы и исключения

Лабораторные задания:

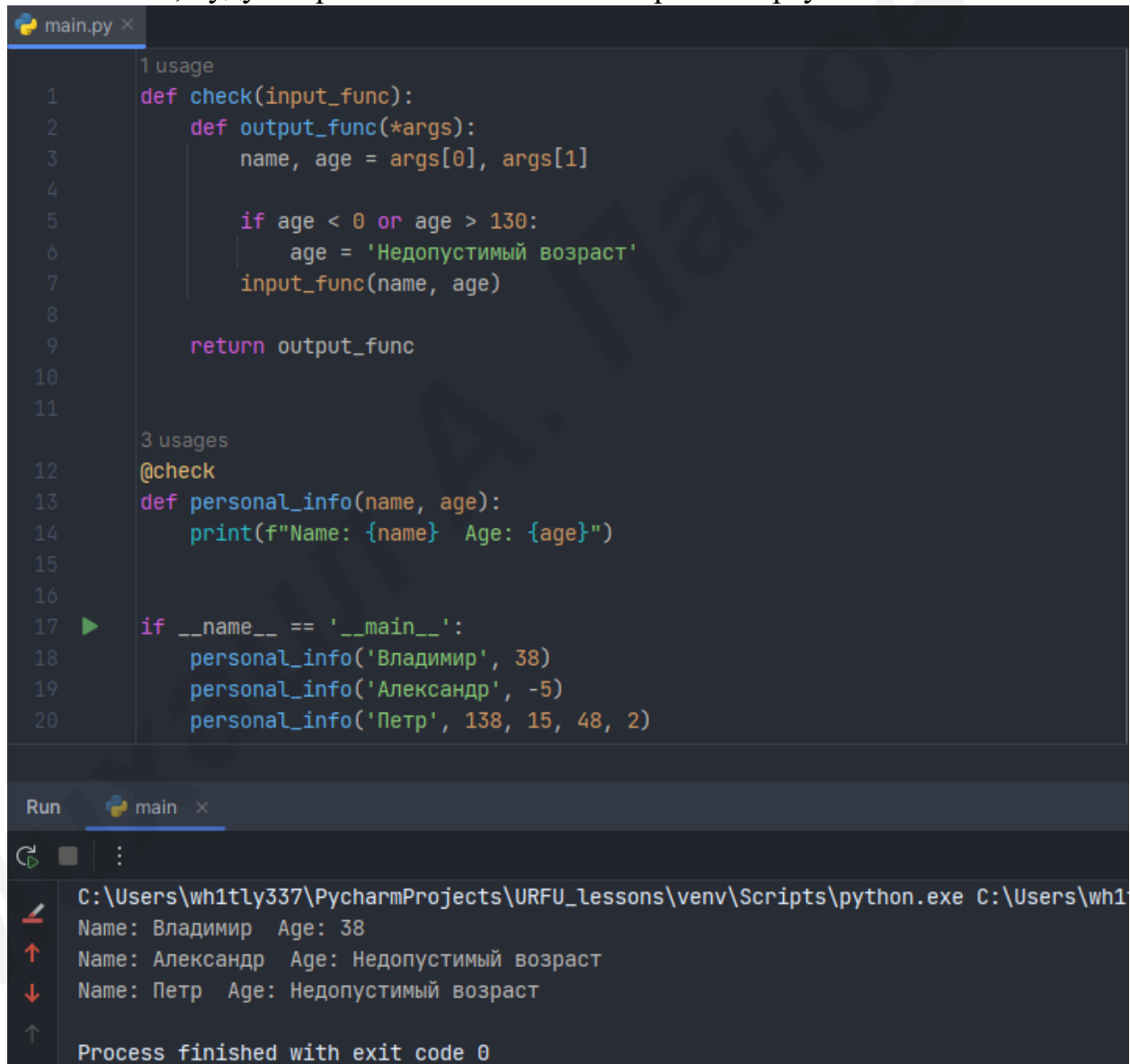
- 1) Наверняка вы думаете, что декораторы – это какая-то бесполезная вещь, которая вам никогда не пригодится, но тут вдруг на паре по математике преподаватель просит всех посчитать число Фибоначчи для 100. Кто-то будет считать вручную (так точно не нужно), кто-то посчитает на калькуляторе, а кто-то подумает, что он самый крутой и напишет рекурсивную программу на Python и немного огорчится, потому что данная программа будет достаточно долго считаться, если ее просто так запускать. Но именно тут к вам на помощь приходят декораторы, например `@lru_cache` (он предназначен для решения задач динамическим программированием, если простыми словами, то этот декоратор запоминает промежуточные результаты и при рекурсивном вызове функции программа не будет считать одни и те же значения, а просто “возьмёт их из этого декоратора”). Вам нужно написать программу, которая будет считать числа Фибоначчи для 100 и запустить ее без этого декоратора и с ним, посмотреть на разницу во времени решения поставленной задачи.
P.S. при запуске без декоратора можете долго не ждать, для наглядности хватит 10 секунд ожидания.



```
main.py x
1  from functools import lru_cache
2
3
4  3 usages
5  @lru_cache(None) # декоратор динамического программирования
6  def fibonacci(n):
7      if n == 0:
8          return 0
9      elif n == 1:
10         return 1
11     return fibonacci(n - 1) + fibonacci(n - 2)
12
13  if __name__ == '__main__':
14     print(fibonacci(100))

Run  main x
C:\Users\wh1tly337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\wh1
354224848179261915075
Process finished with exit code 0
```

- 2) Илья пишет свой сайт и ему необходимо сделать минимальную проверку ввода данных пользователя при регистрации. Для этого он реализовал функцию, которая выводит данные пользователя на экран и решил, что будет проверять правильность введенных данных при помощи декоратора, но в этом ему потребовалась ваша помощь. Напишите декоратор для функции, который будет принимать все параметры вызываемой функции (имя, возраст) и проверять чтобы возраст был больше 0 и меньше 130. Причем заметьте, что неважно сколько пользователь введет данных на сайт к Илье, будут обрабатываться только первые 2 аргумента.



```
main.py x
1 usage
2 def check(input_func):
3     def output_func(*args):
4         name, age = args[0], args[1]
5
6         if age < 0 or age > 130:
7             age = 'Недопустимый возраст'
8         input_func(name, age)
9
10    return output_func
11
12 3 usages
13 @check
14 def personal_info(name, age):
15     print(f"Name: {name} Age: {age}")
16
17 if __name__ == '__main__':
18     personal_info('Владимир', 38)
19     personal_info('Александр', -5)
20     personal_info('Петр', 138, 15, 48, 2)
```

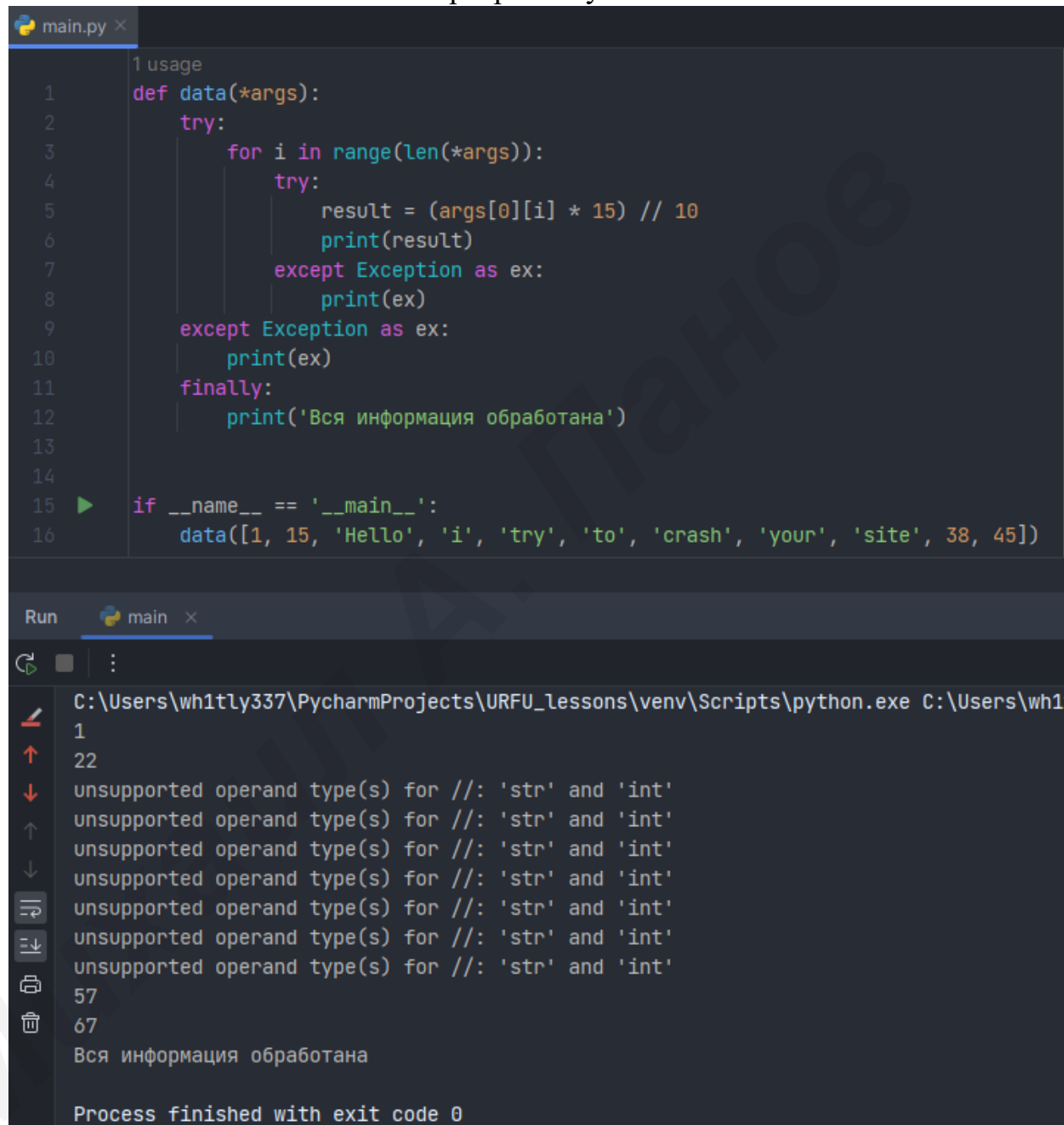
Run main x

C:\Users\wh1tLy337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\wh1
Name: Владимир Age: 38
Name: Александр Age: Недопустимый возраст
Name: Петр Age: Недопустимый возраст
Process finished with exit code 0

- 3) Вам понравилась идея Ильи с сайтом, и вы решили дальше работать вместе с ним. Но вот в вашем проекте появилась проблема, кто-то пытается сломать вашу функцию с получением данных для сайта. Эта функция работает только с данными integer, а какой-то недотакер пытается все сломать и вместо нужного типа данных отправляет string.

Воспользуйтесь исключениями, чтобы неподходящий тип данных не ломал ваш сайт.

Также дополнительно можете обернуть весь код функции в try/except/finally для того, чтобы программа вас оповестила о том, что выявлена какая-то ошибка или программа успешно выполнена.



```
main.py x
1 usage
2 def data(*args):
3     try:
4         for i in range(len(*args)):
5             try:
6                 result = (args[0][i] * 15) // 10
7                 print(result)
8             except Exception as ex:
9                 print(ex)
10    except Exception as ex:
11        print(ex)
12    finally:
13        print('Вся информация обработана')
14
15 if __name__ == '__main__':
16     data([1, 15, 'Hello', 'i', 'try', 'to', 'crash', 'your', 'site', 38, 45])

Run main x
C:\Users\wh1tLy337\PycharmProjects\URFU_lessons\venv\Scripts\python.exe C:\Users\wh1
1
22
unsupported operand type(s) for //: 'str' and 'int'
unsupported operand type(s) for //: 'str' and 'int'
unsupported operand type(s) for //: 'str' and 'int'
unsupported operand type(s) for //: 'str' and 'int'
unsupported operand type(s) for //: 'str' and 'int'
unsupported operand type(s) for //: 'str' and 'int'
unsupported operand type(s) for //: 'str' and 'int'
57
67
Вся информация обработана

Process finished with exit code 0
```

- 4) Продолжая работу над сайтом, вы решили написать собственное исключение, которое будет вызываться в случае, если в функцию проверки имени при регистрации передана строка длиннее десяти символов, а если имя имеет допустимую длину, то в консоль выводиться “Успешная регистрация”

```
main.py x
1 1 usage
2 class NegativeValueException(Exception):
3     pass
4
5 1 usage
6 def check_name(name):
7     if len(name) > 10:
8         raise NegativeValueException('Длина более 10 символов')
9     else:
10        print('Успешная регистрация')
11
12 12 > if __name__ == '__main__':
13     name = '12345678910'
14     check_name(name)

Run main x
/usr/local/bin/python3.11 /Users/user/PycharmProjects/URFU_lessons/main.py
Traceback (most recent call last):
  File "/Users/user/PycharmProjects/URFU_lessons/main.py", line 14, in <module>
    check_name(name)
  File "/Users/user/PycharmProjects/URFU_lessons/main.py", line 7, in check_name
    raise NegativeValueException('Длина более 10 символов')
NegativeValueException: Длина более 10 символов
Process finished with exit code 1
```

- 5) После запуска сайта вы поняли, что вам необходимо добавить логгер, для отслеживания его работы. Готовыми вариантами вы не захотели пользоваться, и поэтому решили создать очень простую пародию. Для этого создали две функции: `__init__()` (вызывается при создании класса декоратора в программе) и `__call__()` (вызывается при вызове декоратора). Создайте необходимый вам декоратор. Выведите все логи в консоль.

```
main.py x
1 1 usage
2 class SiteChecker:
3     def __init__(self, func):
4         print('> Класс SiteChecker метод __init__ успешный запуск')
5         self.func = func
6
7     def __call__(self):
8         print('> Проверка перед запуском', self.func.__name__)
9         self.func()
10        print('> Проверка безопасного выключения')
11
12 1 usage
13 @SiteChecker
14 def site():
15     print('Усердная работа сайта')
16
17 if __name__ == '__main__':
18     print('>> Сайт запущен')
19     site()
20     print('>> Сайт выключен')
```

Run main x

/usr/local/bin/python3.11 /Users/user/PycharmProjects/URFU_lessons/main.py
> Класс SiteChecker метод __init__ успешный запуск
>> Сайт запущен
> Проверка перед запуском site
Усердная работа сайта
> Проверка безопасного выключения
>> Сайт выключен
Process finished with exit code 0