

Отчёт

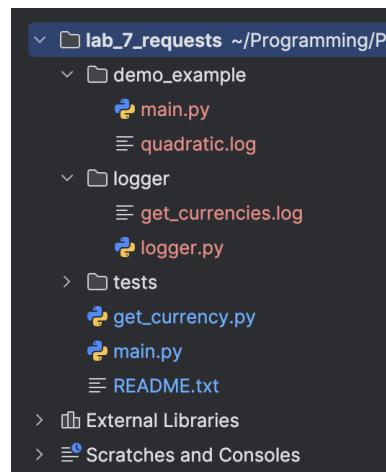
Лабораторная работа №7. Логирование и
обработка ошибок в Python

Выполнил: Голубев Артём

Ису: 502712

Группа: Р3124

Структура проекта



Код декоратора

logger/logger.py

```
1 import functools
2 import logging
3 import sys
4
5
6 def logByHandle(handle, message, level="INFO"):
7     """
8         Функция для логирования сообщений разными способами
9     Args:
10         handle: способ логирования
11         message: сообщение для логирования
12         level: уровень логирования
13     """
14     if isinstance(handle, logging.Logger):
15         if level == "ERROR":
16             handle.error(message)
17         else:
18             handle.info(message)
19
20     else:
21         try:
22             handle.write(f"{level}: {message}\n")
23         except:
24             sys.stderr.write(f"[Unknown handle] {level}: {message}\n")
```

```
28  def logger(func=None, *, handle=sys.stdout): 6 usages
29      """
30          Декоратор, с помощью которого логируются ошибки обёрнутой функции
31          Args:
32              func: функция которую требуется обернуть
33              handle: способ логирования
34          Returns:
35              wrapper: обёрнутая функция func
36      """
37      if func is None:
38          return lambda func: logger(func, handle=handle)
39
40      @functools.wraps(func)
41      def wrapper(*args, **kwargs):
42          # Запись о начале работы функции
43          all_args = [repr(a) for a in args]
44          all_kwargs = [f"{k}={repr(v)}" for k, v in kwargs.items()]
45          all_args_kwargs = ", ".join(all_args + all_kwargs)
46          start_msg = f"Вызов функции {func.__name__}({all_args_kwargs})"
47          logByHandle(handle, start_msg)
48
49          try:
50              result = func(*args, **kwargs)
51              # Запись о успешном выполнении работы
52              res_msg = f"Функция {func.__name__} отработала успешно! Результат: {result}"
53              logByHandle(handle, res_msg)
54              return result
55
56          except Exception as e:
57              # Запись об ошибке
58              err_msg = f"Ошибка в {func.__name__}: {type(e).__name__}: {e}"
59              logByHandle(handle, err_msg, level="ERROR")
60
61              raise
62
63      return wrapper
```

Исходный код get_currencies

get_currency.py

```

import requests
from logger.logger import logger
import logging
import io

# Инициализация логгера
logging.basicConfig(
    filename="logger/get_currencies.log",
    level=logging.DEBUG,
    format"%(levelname)s: %(message)s",
    filemode='w'
)
log = logging.getLogger('currency')

# Инициализация нестандартного потока вывода (в моём случае StringIO)
nonStandartStream = io.StringIO()

```

```

20     # @Logger(handle=nonStandartStream)
21     @Logger 10 usages  ↳ slw0000 *
22     #@logger(handle=log)
23     def get_currencies(currency_codes: list, url: str = "https://www.cbr-xml-daily.ru/daily_json.js")->dict:
24         """
25             Получает курсы валют с API Центробанка России.
26             Args:
27                 currency_codes (list): Список символьных кодов валют (например, ['USD', 'EUR']).
28                 url (str): Get запрос для получения курса валют (по умол. https://www.cbr-xml-daily.ru/daily\_json.js).
29             Returns:
30                 dict: Словарь, где ключи - символьные коды валют, а значения - их курсы.
31                 | Возвращает None в случае ошибки запроса.
32             """
33
34     response = requests.get(url)
35     response.raise_for_status()
36     data = response.json()
37     currencies = {}
38
39     if "Valute" in data:
40         for code in currency_codes:
41             if code in data["Valute"]:
42                 currencies[code] = data["Valute"][code]["Value"]
43                 if not isinstance(currencies[code], (int, float)):
44                     raise TypeError("Курс валюты имеет неверный тип")
45                 else:
46                     currencies[code] = f"Код валюты '{code}' не найден."
47
48     return currencies

```

Демонстрационный пример (квадратное уравнение)

demo_examle/main.py

```
1 import logging
2 import math
3
4 logging.basicConfig(
5     filename="quadratic.log",
6     level=logging.DEBUG,
7     format"%(levelname)s: %(message)s",
8     filemode="w"
9 )
10
11 def solve_quadratic(a, b, c): 1 usage
12     logging.info(f"Solving equation: {a}x^2 + {b}x + {c} = 0")
13
14     # Ошибка типов
15     for name, value in zip(("a", "b", "c"), (a, b, c)):
16         if not isinstance(value, (int, float)):
17             logging.error(f"Parameter '{name}' must be a number, got: {value}")
18             raise TypeError(f"Coefficient '{name}' must be numeric")
19
20     # Ошибка: a == 0
21     if a == b == 0:
22         logging.critical("Coefficient 'a' cannot be zero")
23         raise ValueError("a cannot be zero")
24
25     d = b*b - 4*a*c
26     logging.debug(f"Discriminant: {d}")
27
28     if d < 0:
29         logging.warning("Discriminant < 0: no real roots")
30     return None
```

```

32         if d == 0:
33             x = -b / (2*a)
34             logging.info("One real root")
35             return (x,)
36
37             root1 = (-b + math.sqrt(d)) / (2*a)
38             root2 = (-b - math.sqrt(d)) / (2*a)
39             logging.info("Two real roots computed")
40             return root1, root2
41
42
43
44     # solve_quadratic(1, 1, 0)
45     # solve_quadratic(0, 0, 1)
46     solve_quadratic(a: 1, b: 1, c: 4)

```

Фрагменты логов

Стандартный поток вывода

```

/usr/local/bin/python3.14 /Users/artemgolubev/Programming/Python_ITMO_Education/lab_7_requests/main.py
INFO: Вызов функции get_currencies(['USD', 'EUR', 'GBP'])
INFO: Функция get_currencies отработала успешно! Результат: {'USD': 77.9556, 'EUR': 90.5903, 'GBP': 102.9092}
{'USD': 77.9556, 'EUR': 90.5903, 'GBP': 102.9092}

Process finished with exit code 0

```

```

/usr/local/bin/python3.14 /Users/artemgolubev/Programming/Python_ITMO_Education/lab_7_requests/main.py
INFO: Вызов функции get_currencies(['USD', 'EUR', 'GBK'])
INFO: Функция get_currencies отработала успешно! Результат: {'USD': 77.9556, 'EUR': 90.5903, 'GBK': "Код валюты 'GBK' не найден."}
{'USD': 77.9556, 'EUR': 90.5903, 'GBK': "Код валюты 'GBK' не найден."}

Process finished with exit code 0

```

```

/usr/local/bin/python3.14 /Users/artemgolubev/Programming/Python_ITMO_Education/lab_7_requests/main.py
INFO: Вызов функции get_currencies(['USD', 'EUR', 'GBR'], url='https://google.com')
ERROR: Ошибка в get_currencies: JSONDecodeError: Expecting value: line 1 column 1 (char 0)
Traceback (most recent call last):

```

Логирование в файл

```

1 INFO: Вызов функции get_currencies(['USD', 'EUR', 'GBR'])
2 DEBUG: Starting new HTTPS connection (1): www.cbr-xml-daily.ru:443
3 DEBUG: https://www.cbr-xml-daily.ru:443 "GET /daily_json.js HTTP/1.1" 200 2312
4 INFO: Функция get_currencies отработала успешно! Результат: {'USD': 77.9556, 'EUR': 90.5903, 'GBR': "Код валюты 'GBR' не найден."}
5

```

```
1 INFO: Вызов функции get_currencies(['USD', 'EUR', 'GBL'])
2 DEBUG: Starting new HTTPS connection (1): www.cbr-xml-daily.ru:443
3 DEBUG: https://www.cbr-xml-daily.ru:443 "GET /daily_json.js HTTP/1.1" 200 2312
4 INFO: Функция get_currencies отработала успешно! Результат: {'USD': 77.9556, 'EUR': 90.5903, 'GBL': "Код валюты 'GBL' не найден."}
5
```

```
1 INFO: Вызов функции get_currencies(['USD', 'EUR', 'GBR'], url='https://google.com')
2 DEBUG: Starting new HTTPS connection (1): google.com:443
3 DEBUG: https://google.com:443 "GET / HTTP/1.1" 301 220
4 DEBUG: Starting new HTTPS connection (1): www.google.com:443
5 DEBUG: https://www.google.com:443 "GET / HTTP/1.1" 200 None
6 ERROR: Ошибка в get_currencies: JSONDecodeError: Expecting value: line 1 column 1 (char 0)
7
```

Тесты

Проверка работы функции get_currencies

```
class TestGetCurrencies(unittest.TestCase):    new *
    """
    Тестирование функции get_currencies()
    """

    def test_currency(self):    new *
        """
        Проверяет наличие ключа в словаре и значения этого ключа
        """
        MAX_R_VALUE = 1000
        currency_list = ['USD', 'EUR']

        currency_data = get_currencies(currency_list)
        self.assertIn(currency_list[0], currency_data)
        self.assertIsInstance(currency_data['USD'], float)
        self.assertGreaterEqual(currency_data['USD'], b: 0)
        self.assertLessEqual(currency_data['USD'], MAX_R_VALUE)
```

```
29 ▶     def test_wrong_valute(self): new *
30
31         """
32             Проверяет поведение функции при несуществующей валюте
33         """
34
35         currency_list = ['USD', 'EUR', 'GBR']
36
37         self.assertIn(member="Код валюты", get_currencies(currency_list)['GBR'])
38         self.assertIn(member="не найден", get_currencies(currency_list)['GBR'])
39 ▶     def test_connection_error(self): new *
40
41         """
42             Проверяет правильность вызова ConnectionError
43         """
44
45         currency_list = ['USD', 'EUR']
46
47         with self.assertRaises(requests.exceptions.ConnectionError):
48             get_currencies(currency_list, url='https://www.notcorrectapi.ru')
49 ▶     def test_value_error(self): new *
50
51         """
52             Проверяет правильность вызова ValueError
53         """
54
55         currency_list = ['USD', 'EUR']
56
57         with self.assertRaises(ValueError):
58             get_currencies(currency_list, url='https://google.com')
```

Тесты декоратора (IOStream)

```
61 ▶ class TestIOWrite(unittest.TestCase): new *
62     """
63     Тестирование логирования работы функции и вывода ошибок через IOWrite
64     """
65
66 ⚡ def setUp(self): new *
67     """
68     Настройка функции для тестирования
69     """
70
71     self.stream = io.StringIO()
72     self.decorated_func = logger(handle=self.stream)(get_currencies)
73
74 ▶ def test_logging_error(self): new *
75     """
76     Проверка логов при ошибках и правильность выбросов ошибок
77     """
78
79     with self.assertRaises(requests.exceptions.ConnectionError):
80         self.decorated_func(['USD'], url="https://www.notcorrectapi.ru")
81
82     logs = self.stream.getvalue()
83     self.assertRegex(logs, expected_regex="ERROR")
84     self.assertIn(member="ConnectionError", logs)
85
```

```
87 ▶ def test_logging_correct_work(self): new *
88     """
89     Проверка логов при успешной работе функции
90     """
91
92     self.decorated_func(['USD', 'EUR'])
93
94     logs = self.stream.getvalue()
95     self.assertIn(member="INFO", logs)
96     self.assertIn(member="Вызов функции", logs)
97     self.assertIn(member="['USD', 'EUR']", logs)
98
99     self.assertIn(member="INFO", logs)
100    self.assertIn(member="отработала успешно", logs)
101    self.assertRegex(logs, expected_regex=r"\{'USD':\s*\d+\.\?\d*,\s*'EUR':\s*\d+\.\?\d*\}")
102
103
104 ⚡ def tearDown(self): new *
105     del self.stream
106
```

Тесты декоратора (logging)

```
109 >     class TestLogWrite(unittest.TestCase): new *
110
111     """
112         Тестирование логирования работы функции и вывода ошибок в логгер
113     """
114     @
115         def setUp(self): new *
116             """
117                 Настройка функции для тестирования
118             """
119             logging.basicConfig(
120                 filename="logger/get_currencies.log",
121                 level=logging.DEBUG,
122                 format"%(levelname)s: %(message)s",
123                 filemode='w'
124             )
125             log = logging.getLogger('currency')
126
127             self.decorated_func = logger(handle=log)(get_currencies)
128
```

```
129 ▶     def test_logging_error(self): new *
130
131     """
132         Проверка логов при ошибках и правильность выбросов ошибок
133     """
134
135         with self.assertRaises(requests.exceptions.ConnectionError):
136             self.decorated_func(['USD'], url="https://www.notcorrectapi.ru")
137
138             self.logger = open("logger/get_currencies.log")
139             log = self.logger.read()
140
141             self.assertRegex(log, expected_regex="ERROR")
142             self.assertIn(member="ConnectionError", log)
143
144 ▶     def test_logging_correct_work(self): new *
145
146     """
147         Проверка логов при успешной работе функции
148     """
149
150         self.decorated_func(['USD', 'EUR'])
151
152         self.logger = open("logger/get_currencies.log")
153         log = self.logger.read()
154
155         self.assertIn(member="INFO", log)
156         self.assertIn(member="Вызов функции", log)
157         self.assertIn(member="['USD', 'EUR']", log)
158
159         self.assertIn(member="INFO", log)
160         self.assertIn(member="отработала успешно", log)
161         self.assertRegex(log, expected_regex=r"\{'USD':\s*\d+\.\?\d*,\s*'EUR':\s*\d+\.\?\d*\}")
```

Все тесты были успешно пройдены!!!