

Лабораторная работа 4.4

Тема: Работа с исключениями в языке Python

Цель работы: приобретение навыков по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.

Ссылка на GitHub: https://github.com/RomanGorchakov/Py3_5


Порядок выполнения работы

1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *	Repository name *
 RomanGorchakov ▾	/ Test
✔ Test is available.	

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)


Choose a license



▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

 **Test** Public

 Pin  Unwatch 1

 main  1 branch  0 tags

Go to file

Add file 

 Code 

 **RomanGorchakov** Initial commit 87a4818 now  1 commit

 .gitignore

Initial commit now

 LICENSE


















Initial commit now

Help people interested in this repository understand your project by adding a README.

Add a README

 © 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#)

2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».

	Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
	PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
	Plone.gitignore	Covered by global vim template	10 years ago
	Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
	Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
	PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
	Python.gitignore	Update Python.gitignore	last year
	Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
	Qt.gitignore	Remove trailing whitespace	2 years ago
	R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
	README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
	ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
	Racket.gitignore	Update Racket.gitignore	2 years ago
	Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
	Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
	RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
	Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```

1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28

```

3. Теперь создаём файл «README.md», где вносим ФИО и теоретический конспект лекции. Сохраняем набранный текст через кнопку «Commit changes».

The screenshot shows the GitHub web interface for a repository named 'Py3.5' by user 'RomanGorchakov'. The 'Code' tab is selected, and the file 'README.md' is open in the editor. The editor shows the following content:

```

1  Горчаков Роман Владимирович. Вариант 2
2  # Лабораторная работа 4.4. Работа с исключениями в языке Python
3
4  Исключениями (exceptions) в языках программирования называют проблемы, возникающие в ходе выполнения программы, которые допускают возможность дальнейшей ее работы в рамках основного алгоритма. Типичным примером
5  исключения является деление на ноль, невозможность считать данные из файла (устройства), отсутствие доступной памяти, доступ к закрытой области памяти и т.п. Для обработки таких ситуаций в языках
6  программирования, как правило, предусматривается специальный механизм, который называется обработкой исключений (exception handling).
7
8  Исключения разделяют на синхронные и асинхронные. Синхронные исключения могут возникнуть только в определенных местах программы. Например, если у вас есть код, который открывает файл и считывает из него данные,
9  то исключение типа "ошибка чтения данных" может произойти только в указанном куске кода. Асинхронные исключения могут возникнуть в любой момент работы программы, они, как правило, связаны с какими-либо
10 аппаратными проблемами, либо приходом данных. В качестве примера можно привести сигнал отключения питания.
11
12 В языках программирования чаще всего предусматривается специальный механизм обработки исключений. Обработка может быть с возвратом, когда после обработки исключения выполнение программы продолжается с того места,
13 где оно возникло. И обработка без возврата, в этом случае, при возникновении исключения, осуществляется переход в специальный, заранее подготовленный, блок кода.
14
15 Различают структурную и неструктурную обработку исключений. Неструктурная обработка предполагает регистрацию функции обработчика для каждого исключения, соответственно данная функция будет вызвана при
16 возникновении конкретного исключения. Для структурной обработки язык программирования должен поддерживать специальные синтаксические конструкции, которые позволяют выделить код, который необходимо контролировать
17 и код, который нужно выполнять при возникновении исключительной ситуации.

```

The interface includes a search bar, navigation tabs (Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings), a license notice, and a commit bar with 'Cancel changes' and 'Commit changes...' buttons. The editor also shows a 'Preview' tab and a 'Code 55% faster with GitHub Copilot' badge.

4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствие с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout -b develop» для создания ветки разработки; «git branch feature_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix. Устанавливаем библиотеки isort, black и flake8 и создаём файлы .pre-commit-config.yaml и environment.yml.

```
@RomanGorchakov →/workspaces/Py3_5 (main) $ git checkout -b develop
Switched to a new branch 'develop'
● @RomanGorchakov →/workspaces/Py3_5 (develop) $ git branch feature_branch
● @RomanGorchakov →/workspaces/Py3_5 (develop) $ git branch release/1.0.0
● @RomanGorchakov →/workspaces/Py3_5 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov →/workspaces/Py3_5 (main) $ git branch hotfix
● @RomanGorchakov →/workspaces/Py3_5 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py3_5 (develop) $ █

Collecting black
  Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (79 kB)
Collecting click>=8.0.0 (from black)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting mypy_extensions>=0.4.3 (from black)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: packaging>=22.0 in /home/codespace/.local/lib/python3.12/site-packages (from black) (24.1)
Collecting pathspec>=0.9.0 (from black)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in /home/codespace/.local/lib/python3.12/site-packages (from black) (4.3.6)
Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (1.8 MB)
1.8/1.8 MB 51.8 MB/s eta 0:00:00
Downloading click-8.1.7-py3-none-any.whl (97 kB)
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, mypy_extensions, click, black
Successfully installed black-24.10.0 click-8.1.7 mypy_extensions-1.0.0 pathspec-0.12.1
● @RomanGorchakov →/workspaces/Py3_5 (develop) $ pip install flake8
Collecting flake8
  Downloading flake8-7.1.1-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting mccabe<0.8.0,>=0.7.0 (from flake8)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting pycodestyle<2.13.0,>=2.12.0 (from flake8)
  Downloading pycodestyle-2.12.1-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting pyflakes<3.3.0,>=3.2.0 (from flake8)
  Downloading pyflakes-3.2.0-py2.py3-none-any.whl.metadata (3.5 kB)
Downloading flake8-7.1.1-py2.py3-none-any.whl (57 kB)
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading pycodestyle-2.12.1-py2.py3-none-any.whl (31 kB)
Downloading pyflakes-3.2.0-py2.py3-none-any.whl (62 kB)
Installing collected packages: pyflakes, pycodestyle, mccabe, flake8
Successfully installed flake8-7.1.1 mccabe-0.7.0 pycodestyle-2.12.1 pyflakes-3.2.0
● @RomanGorchakov →/workspaces/Py3_5 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
● @RomanGorchakov →/workspaces/Py3_5 (develop) $ conda env export > environment.yml
○ @RomanGorchakov →/workspaces/Py3_5 (develop) $ █
```

5. Создаём файл «example.py», в котором нужно добавить возможность работы с исключениями и логгирование для примера 2 лабораторной работы 9.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Пример>python example.py
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Директор
Год поступления? 2007
>>> add
Фамилия и инициалы? Петров П.П.
Должность? Бухгалтер
Год поступления? 2010
>>> add
Фамилия и инициалы? Сидоров С.С.
Должность? Главный инженер
Год поступления? 2012
>>> list
+-----+-----+-----+-----+
| %s | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Иванов И.И. | Директор | 2007 |
| 2 | Петров П.П. | Бухгалтер | 2010 |
| 3 | Сидоров С.С. | Главный инженер | 2012 |
+-----+-----+-----+-----+
>>> exit

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Пример>
```

6. Создаём файл «task1.py», в котором программа запрашивает ввод двух значений. Если хотя бы одно из них не является числом, то должна выполняться конкатенация, т. е. соединение, строк. В остальных случаях введенные числа суммируются.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Задания\Задание 1>python task1.py
Первое значение: 7.2
Второе значение: 2.6
Результат: 9.8
Все верно, вы ввели оба числа 7.2 и 2.6
Программа завершена.

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Задания\Задание 1>python task1.py
Первое значение: A
Второе значение: 113
Результат: A113
Программа завершена.

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Задания\Задание 1>
```

7. Создаём файл «task2.py», в котором программа генерировать матрицу из случайных целых чисел. Пользователь может указать число строк и столбцов, а также диапазон целых чисел. Производим обработку ошибок ввода пользователя.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Задания\Задание 2>python task2.py
Введите количество строк: 5
Введите количество столбцов: 5
Введите минимальное значение: 2
Введите максимальное значение: 10
Сгенерированная матрица:
[4, 4, 2, 8, 5]
[7, 4, 9, 8, 6]
[3, 10, 10, 7, 3]
[6, 2, 10, 9, 9]
[7, 4, 7, 9, 4]

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Задания\Задание 2>python task2.py
Введите количество строк: 4.5
Ошибка: Введите целое число.

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Задания\Задание 2>
```

8. Создаём файл «individual1.py», в котором нужно выполнить индивидуальное задание 1 лабораторной работы 2.19, добавив возможность работы с исключениями и логгирование.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 1>python individual1.py add --db planes.db -r "Moscow" -n 8850 -t 9
2024-12-05 01:21:18,493 [INFO]: Database created successfully.
2024-12-05 01:21:18,523 [INFO]: Plane added successfully: Moscow, 8850, 9

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 1>python individual1.py add --db planes.db -r "Novoalexandrovsk" -n 3234 -t 62
2024-12-05 01:21:44,552 [INFO]: Database created successfully.
2024-12-05 01:21:44,600 [INFO]: Plane added successfully: Novoalexandrovsk, 3234, 62

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 1>python individual1.py add --db planes.db -r "Saint-Petersburg" -n 9864 -t 48
2024-12-05 01:22:19,832 [INFO]: Database created successfully.
2024-12-05 01:22:19,881 [INFO]: Plane added successfully: Saint-Petersburg, 9864, 48

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 1>python individual1.py add --db planes.db -r "Stavropol" -n 9120 -t 82
2024-12-05 01:22:38,688 [INFO]: Database created successfully.
2024-12-05 01:22:38,720 [INFO]: Plane added successfully: Stavropol, 9120, 82

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 1>python individual1.py list --db planes.db
usage: workers [-h] [--version] {add,display,select} ...
workers: error: argument command: invalid choice: 'list' (choose from 'add', 'display', 'select')

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 1>python individual1.py display --db planes.db
2024-12-05 01:23:01,276 [INFO]: Database created successfully.

+-----+-----+-----+-----+
| No | Destination | Race number | Plane type |
+-----+-----+-----+-----+
| 1 | Kazan | 3780 | 1 |
+-----+-----+-----+-----+
| 2 | Moscow | 8850 | 9 |
+-----+-----+-----+-----+
| 3 | Novoalexandrovsk | 3234 | 62 |
+-----+-----+-----+-----+
| 4 | Saint-Petersburg | 9864 | 48 |
+-----+-----+-----+-----+
| 5 | Stavropol | 9120 | 82 |
+-----+-----+-----+-----+

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 1>
```

9. Создаём файл «individual2.py», в котором нужно добавить для предыдущего задания вывод в файлы лога даты и времени выполнения пользовательской команды с точностью до миллисекунды.

```
Command Prompt
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 2>python individual2.py add --db planes.db -r "Kazan" -n 3780 -t 1
2024-12-05 01:25:59.129 INFO: Command executed: add, Arguments: {'command': 'add', 'db': 'planes.db', 'race': 'Kazan', 'number': 3780, 'type': 1}

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 2>python individual2.py add --db planes.db -r "Moscow" -n 8850 -t 9
2024-12-05 01:26:10.744 INFO: Command executed: add, Arguments: {'command': 'add', 'db': 'planes.db', 'race': 'Moscow', 'number': 8850, 'type': 9}

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 2>python individual2.py add --db planes.db -r "Novoalexandrovsk" -n 3234 -t 62
2024-12-05 01:26:24.237 INFO: Command executed: add, Arguments: {'command': 'add', 'db': 'planes.db', 'race': 'Novoalexandrovsk', 'number': 3234, 'type': 62}

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 2>python individual2.py add --db planes.db -r "Saint-Petersburg" -n 9864 -t 48
2024-12-05 01:26:35.236 INFO: Command executed: add, Arguments: {'command': 'add', 'db': 'planes.db', 'race': 'Saint-Petersburg', 'number': 9864, 'type': 48}

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 2>python individual2.py add --db planes.db -r "Stavropol" -n 9120 -t 82
2024-12-05 01:26:47.236 INFO: Command executed: add, Arguments: {'command': 'add', 'db': 'planes.db', 'race': 'Stavropol', 'number': 9120, 'type': 82}

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 2>python individual2.py display --db planes.db
2024-12-05 01:26:54.182 INFO: Command executed: display, Arguments: {'command': 'display', 'db': 'planes.db'}

+-----+-----+-----+-----+
| No | Destination | Race number | Plane type |
+-----+-----+-----+-----+
| 1 | Kazan | 3780 | 1 |
+-----+-----+-----+-----+
| 2 | Moscow | 8850 | 9 |
+-----+-----+-----+-----+
| 3 | Novoalexandrovsk | 3234 | 62 |
+-----+-----+-----+-----+
| 4 | Saint-Petersburg | 9864 | 48 |
+-----+-----+-----+-----+
| 5 | Stavropol | 9120 | 82 |
+-----+-----+-----+-----+

C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\5\Индивидуальные задания\Задание 2>
```

10. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.

```
16 files changed, 1359 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 .python-version
create mode 100644 edu.pyoop.code.workspace
create mode 100644 environment.yml
create mode 100644 pyproject.toml
create mode 100644 setup.cfg
create mode 100644 uv.lock
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 1/task1.py"
create mode 100644 "\320\227\320\260\320\264\320\260\320\275\320\270\321\217\320\227\320\260\320\264\320\260\320\275\320\270\320\265 2/task2.py"
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 1/individual1.py"
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 1/planes.db"
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 1/python"
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 2/individual2.py"
create mode 100644 "\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\321\213\320\265 2/planes.db"
create mode 100644 "\320\236\321\202\321\207\321\221\321\202\320\233\320\2404.3_320\223\320\276\321\200\321\207\320\260\320\272\320\276\320\262\320\240\320\222.pdf"
create mode 100644 "\320\237\321\200\320\270\320\274\320\265\321\200/example.py"
● @RomanGorchakov →/workspaces/Py3_5 (main) $ git push -u
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 2 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (26/26), 867.37 KiB | 15.49 MiB/s, done.
Total 26 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/RomanGorchakov/Py3_5
23452fe..98c8ea2 main -> main
branch 'main' set up to track 'origin/main'.
○ @RomanGorchakov →/workspaces/Py3_5 (main) $
```


Py3_5Public

Pin
Unwatch1
Fork0
Star0

main1 Branch0 Tags
Go to file
Add file
Code

RomanGorchakovExceptions98c8ea2 · now4 Commits

Задания	Exceptions	now
Индивидуальные задания	Exceptions	now
Отчёт	Exceptions	now
Пример	Exceptions	now
.gitignore	Create .gitignore	last week
.pre-commit-config.yaml	Exceptions	now
.python-version	Exceptions	now
LICENSE	Create LICENSE	last week
README.md	Create README.md	last week
edu.pyoop.code-workspace	Exceptions	now
environment.yml	Exceptions	now
pyproject.toml	Exceptions	now
setup.cfg	Exceptions	now
uv.lock	Exceptions	now

README
MIT license

Горчаков Роман Владимирович. Вариант 2

Лабораторная работа 4.4. Работа с исключениями в языке Python

Python Package using AnacondaConfigure

Create and test a Python package on multiple Python versions using Anaconda for package management.

Publish Python PackageConfigure

Publish a Python Package to PyPI on release.

Контрольные вопросы

1. Какие существуют виды ошибок в языке программирования Python?

В Python выделяют два различных вида ошибок: синтаксические ошибки и исключения.

2. Как осуществляется обработка исключений в языке программирования Python?

Обработка исключений нужна для того, чтобы приложение не завершалось аварийно каждый раз, когда возникает исключение. Для этого блок кода, в котором возможно появление исключительной ситуации необходимо поместить во внутрь синтаксической конструкции `try... except`.

3. Для чего нужны блоки `finally` и `else` при обработке исключений?

Не зависимо от того, возникнет или нет во время выполнения кода в блоке `try` исключение, код в блоке `finally` все равно будет выполнен. Конструкция `else`, которая выполняется в случае, если в коде не произошло исключений.

4. Как осуществляется генерация исключений в языке Python?

Для принудительной генерации исключения используется инструкция `raise`.

5. Как создаются классы пользовательский исключений в языке Python?

В Python можно создавать собственные исключения. Такая практика позволяет увеличить гибкость процесса обработки ошибок в рамках той предметной области, для которой написана ваша программа. Для реализации собственного типа исключения необходимо создать класс, являющийся наследником от одного из классов исключений.

6. Каково назначение модуля `logging`?

С помощью `logging` на Python можно записывать в лог и исключения. Обычно лог пишется в файл, зададим его как `log.txt`

7. Какие уровни логгирования поддерживаются модулем `logging`?

Приведите примеры, в которых могут быть использованы сообщения с этим уровнем журналирования.

Самый низкий уровень логгирования – `DEBUG`. Он предназначен для отладочных сообщений и вывода диагностической информации о приложении. Уровень `INFO` указывает, что сообщения уровней ниже не будут отражаться в логге. Уровень `WARNING` предусматривает вывод предупреждений, он применяется для записи сведений о событиях, на которые программист обычно обращает внимание. Уровень `ERROR` предусматривает вывод сведений об ошибках. Уровень `CRITICAL` используется для вывода сведений об очень серьезных ошибках, наличие которых угрожает нормальному функционированию всего приложения.

```
import logging
logging.basicConfig(level = logging.DEBUG)
logging.debug("Debug message!")
logging.info("Info message!")
logging.warning("Warning message!")
logging.error("Error message!")
logging.critical("Critical message!")
```

>> DEBUG:root:Debug message!
>> INFO:root:Info message!
>> WARNING:root:Warning message!
>> ERROR:root:Error message!
>> CRITICAL:root:Critical message!