

SLIDE 1

Logging ██████████

```
92.63.107.227 - - [04/Nov/2020:06:30:48 +0000] "GET /ru/hosted-open-vpn-server/ HTTP/1.1" 301 169
 "-" "python-requests/2.11.1" "-" 92.63.107.227 - - [04/Nov/2020:06:30:49 +0000] "GET
 /ru/data-engineering-course/ HTTP/1.1" 301 169 "-" "python-requests/2.11.1" "-" 213.180.203.50 - -
 [04/Nov/2020:06:36:07 +0000] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (compatible;
 YandexMetrika/2.0; +http://yandex.com/bots yabs01)" "-" 114.119.160.75 - - [04/Nov/2020:06:36:41
 +0000] "GET /robots.txt HTTP/1.1" 301 169 "-" "(compatible;PetalBot;+https://aspiegel.com/petalbot)"
 "10.179.80.67" 90.180.35.207 - - [04/Nov/2020:06:47:11 +0000] "GET / HTTP/1.0" 301 169 "-" "-" "-"
 46.246.122.77 - - [04/Nov/2020:06:53:22 +0000] "GET / HTTP/1.1" 301 169 "" "Mozilla/5.0 (Macintosh;
 Intel Mac OS X 10_12_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181
 Safari/537.36" "-" 66.249.76.16 - - [04/Nov/2020:06:53:30 +0000] "GET / HTTP/1.1" 301 169 "-"
 "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" "-"
```

[illegible]

« »

SLIDE 2

```
■■■ ■■■■■ ■■■■■■■■■■■■ ("■■■■■■■■■■■■■■■■■■")?
```

XXXXXXXXXXXXX A XXXXXX XXXXXXXX XXXXXXXXXXXXX XXXXXXXXXXXXX
THIS XXXXXXXX A XXXXXXXXXXX XXXXXXXXXXXXX X THIS XXXXXXXX A XXXXXXXXXXXXX
XXXXXXXXXX X XXXXXXXX X XXXXXXXX XX.

00 000000000000 000000 00000 000000000 0 0000 00000000 000000:

SLIDE 3

■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ?

██████████ ██████████,

■■■■■■■■■

■■■■■■■■■.

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ :

██████████ ■ █████ / █ API █ ████████████████████,

■■■■■■■■■■ ■■ email ■■■

■ ■ ■ ■ ■ ■ ■ ■ ■ ■ (■ ■ ■ ■ ■ ■ ■ ■ ■ ■)

■■■■ ■■■■■■ ■■■■■■■■■■: ■■■■ ■■ ■■■■■■ ■■ ■■■■■■■■■■■■ ■■■■■■■■■■
 ■■■■■■■■■■■■ ■■■■■■■■ SMS — ■■■ ■■■■■ ■■■■ ■■ ■■■■■■■■■■.

SLIDE 4

■ ■ ■ ■ ■ ■ ■ ? ■ ■ ■ ■ ■ ■ ?

XXXXXXXXXXXX XXXXXX XXXXXXXX X XXXXX XXXXXXXX XXXXXXXX:

XXXXXXXXXX XXXXXXXXXX

2023-08-12 17:49:37 User 'johndoe' successfully logged in

■■■ ■■■■■■■■■■?

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ?

«**Судьба**» — **роман** **Андрея** **Битова**, **первый** **роман** **Битова** **о** **русской** **интеллигенции** **и** **русской** **культуре**, **он** **я** **русской** **интеллигенции**, **он** **русской** **культуре**, **он** **русской** **культуре** **русской** **интеллигенции** **он** **он** **русской** **культуре**, **русской** **интеллигенции** **русской** **культуре**.

SLIDE 5

XXXXXXXXXX XXXXXXXX X XXXXX: XXXXXXXXXXXXXXXXXXXX

[illegible]

«Step initiate» «OK Transaction failed»

[illegible]

«User 'johndoe' successfully logged in from IP 1.2.3.4» «Transaction 1287786 reverted due to lost connection to DB»

[illegible]

SLIDE 6

XXXXXXXXXXXXXXXX XXXXXXXX X XXXX: XXXXXXXXXXXXXXXXXXXXXXXX / XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXX

```
«INFO Checking password hash for 'johndoe'» «DEBUG Hashes don't match» «ERROR Operation failed»
```

1. **Содержание**
 2. **Введение**
 3. **Глава 1. Общие сведения о предприятии**
 4. **Глава 2. Анализ деятельности предприятия**
 5. **Глава 3. Оценка финансового состояния предприятия**
 6. **Глава 4. Оценка эффективности деятельности предприятия**
 7. **Глава 5. Оценка перспектив развития предприятия**
 8. **Заключение**
 9. **Список литературы**
 10. **Приложение**

SLIDE 7

[illegible][illegible]

Call func DB OK

Connection to database successfully established

SLIDE 8

XXXXXXXXXXXX XXXXXXXX X XXXXX: XXXXXX XXXXXXXX

[illegible]

XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXX,
XXXXXXXXXXXXXXXXXXXX ISO8601:

2022-10-02 11:41:42,612

██████████ ████████████████████,

██████████ ████████████████████,

■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ — ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■, ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■. ■■■■■■■■■■, ■■■■■■■■■■, ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■, ■■■■■■■■■■ ■■■■■■■■■■ 10 000 ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■, ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■.

SLIDE 9

XXXXXXXXXXXX XXXXXXXX X XXXXX: XXXXXXXXXXXXXXX

1. **ВВЕДЕНИЕ**. В данном документе описаны основные принципы и правила, которые должны быть соблюдены при работе с информацией. Эти правила являются обязательными для всех сотрудников организации.

■■■■■ :

```
##### -##### IP-####, #####  
##### 2023-09-14 14:22:45,329 172.0.0.1 GET /sitemap.xml 200
```

■■■■ ■■■■■■ ■■■-■■■■■■■ — ■■■■, ■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■

■■ HTTP-■■■■ ■■■■■■ 2023-03-10 16:59:23,556 Request to https://boogle.com failed with 403

```

████████████████████████████████████████ — id ████████████████████, ██████████
████████████████████████████████████████ 2023-02-19 00:19:18,831 Transaction 309841 successful:
$100500 transferred from 940384 to 473923

```

```

    """
    telegram_id = int(telegram_id)
    handler = handler

```

SLIDE 10

██████████ █ ████████: ████████████████: ███████████ █ ███████████

[illegible]

2023-05-15 05:59:46,102 Division by zero at mymodule.divide_by(0):18

[illegible][illegible]

SLIDE 11

[illegible]

NOTE: The following information is provided for informational purposes only. It does not constitute an offer or recommendation by the issuer.

██████████████████ (██ ██████████ ████████████) ████████ ██████████ ████████████ ████████████,

00 0000 00000000 0000000000000000 0000000000000000 000000000000:

2023-02-19 00:19:18 Transaction 309841 successful: \$100500 transferred from 940384 to 473923

[illegible]

2023-02-19 00:19:18 Transaction successful. Transaction_id: 309841, amount: 100500, From: 940384, To: 473923

SLIDE 12

XXXXXXXXXXXXXXXX XXXXXXXX X XXXX: XXXXXXXXXXXXXXXX / XXXXXXXXXXXXXXXXXXXXXXXX

[illegible]

```
{"timestamp":1676755158831,"code":"OK","transaction":{"id":309841,"amount":100500,"from":940384,"to":473923}}
```

■■■■■ ■■■■■■ ■■■■■■ — ■■■■■ ■■ ■■■■, ■■■■ ■■ Ctrl+F ■ ■■■■■■■■
■■■■■■■■■, grep ■■■ ■■■■■■ ■■■■■■■■■■ ■■■■. ■■■ ■■■■■■■■ ■■■■■■
■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ / ■■■■■■■■■■
■■■■■■■■■. ■■■■■■■■, ■■■■ ■■ ■■■-■■ ■■■■■■ user_id, ■ ■■■-■■ client identifier,
■■■-■■ login, ■ ■■■-■■ authorization, ■■ ■■■■■ ■■ ■■■■■, ■■■■■■■■■■
■■■■■■■■■■■ ■■■■■■■■■■ ■■■■■ ■■■■■■■■■■.

SLIDE 13

XXXXXXXXXX XXXXXXXX X XXXXX: XXXXXXXXXXXXXXX

[illegible]

[REDACTED] [REDACTED] [REDACTED] [REDACTED]:

SLIDE 14

□□□□, □□□ □ □□□ □□□□□□□□□□? □□□□□ □□□□□□□□□□

[TABLE CONTENT]

Numerical Code	Severity
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical: critical conditions
3	Error: error conditions
4	Warning: warning conditions
5	Notice: normal but significant condition
6	Informational: informational messages
7	Debug: debug-level messages

«...некоторые из них являются не столько объективными, сколько субъективными, и, следовательно, могут быть оспорены» — это утверждение, которое, как мы увидим, имеет серьезные последствия (severity levels).

[illegible]

SLIDE 15

[illegible]

Python is a high-level, interpreted programming language designed for readability and ease of use. It was created by Guido van Rossum in the late 1980s and has since become one of the most popular languages in the world. Python's syntax is clean and intuitive, making it accessible to beginners while still powerful enough for experienced developers. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

00000000 000000, 00000000 00000000 0000 000000000000 00-00000000.
 00000000, 00000000 INFO 00000 00000000000 0 00000000 0000,
 0000000 0 00000 00000000000 000 0000000 00000 00 0000000000
 000000-00 000000-00000000. 0 000000000 ERROR 000 CRITICAL 00000,
 000000 000000 0 0000, 0000000000000 00 000000000000 00000 0000 0
 00000000000 000 000000 00000000 000000000000000000 0000
 0000000000000000 00 00000000000 0000000000.

[TABLE CONTENT]

■■■■■■■■ | Code | ■■■■■■■■■■■■■■■■■■■■

```
DEBUG | 10 | ██████████ ██████████, ████████ █ ████████████████████ ███
██████████████████.
```

ERROR | 40 | ■■■-■■ ■■■■■■■■■■ ■■■■■■■■ ■■■■■■■■■■ ■■ ■■■■■■ ■■■■■■■■■■
■■■■■-■■ ■■■■ ■■■■■■■■, ■■ ■■■■■■■■■■ ■■■■■■■■.

[SLIDE NOTES]

```
DEBUG — [REDACTED], [REDACTED] [REDACTED]
[REDACTED] [REDACTED] [REDACTED].
```


INFO — [REDACTED] [REDACTED] [REDACTED]
[REDACTED]/[REDACTED].

[illegible]

CRITICAL — [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]
[REDACTED] [REDACTED].

SLIDE 16

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □

■■■■■■■ WARNING, ERROR, CRITICAL ■■■■■■■■■■■■ ■■■■■ ■ ■■■ ■■■■■, ■■■ ■■■
■■■■■■■■■■■.

■■■■■■■■■■ ■■■■■■■■ DEBUG ■ INFO ■■■■■■■■■■ ■■■■■■■■■■. ■■■■■■■■
 ■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■
 ■■■■■■■■■■ ■■■■■■■■■■. ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ — ■■■■■: ✓ ■■■■■■■■■■
 ■■■■■■■■■■, ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ (■■■■■■■■
 ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■); ✓ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■-■■
 ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■. ■■■■■■■■■■ ■■■■■■■■■■
 ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■, ■■■■■■■■■■, ■■■■■■■■■■
 ■■■■■■■■■■, ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■, ■■■■■■■■■■ ■■■■■■■■■■.

[illegible][illegible]

SLIDE 17

XXXXXXXXXXXX XXXXXXXXXXXXXXXX: XXXXXXXX XXXXXXXXXXXXXXX

[illegible][illegible]

SLIDE 18

■■■■■■■■■■ ■■■■■■■■■■

XXXXXXXXXXXX XXXXXX XXXXXXXX XXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX:

■■ ■■■■■■ ■■■■■■■■■■ :

_____,

□□,

[illegible]

██████████,

SLIDE 19

[illegible]

«**Восстановление исторической справедливости по отношению к репрессированным в годы войны и послевоенного периода является одним из приоритетных направлений деятельности органов государственной власти. В настоящее время в стране реализуется ряд масштабных программ, направленных на реабилитацию жертв политических репрессий, в том числе на восстановление их репутации и материальную поддержку. Это включает в себя возмещение ущерба, нанесенного государством, а также создание условий для нормальной жизни и работы пострадавшим и их семьям. Особое внимание уделяется ветеранам войны и послевоенного периода, чьи интересы должны быть защищены государством. В рамках этих программ проводятся различные мероприятия, такие как возведение памятников, проведение исторических исследований, а также предоставление льгот и пособий. Важно отметить, что восстановление справедливости — это не только вопрос материальной компенсации, но и вопрос признания заслуг и восстановления чести тех, кто внес вклад в развитие страны. Поэтому органы государственной власти должны продолжать работу в этом направлении, обеспечивая всем пострадавшим необходимые меры поддержки и реабилитации.**

[illegible]

```
$ ls /var/logs/    keybagd.log.0    keybagd.log.2    usermanagerd.log.0 keybagd.log.1
keybagd.log.3    usermanagerd.log.1
```

[illegible]

SLIDE 20

[illegible]

```
/etc/logrotate.d/myapp:
```

```
/var/log/myapp/error.log { rotate 7 # ██████████ ██ 7 ████████████████████ ████████ daily #
████████████████████████████████████ compress # ██████████ ████████████████████ ████████ notifempty #
██ ████████████████████, █████ ████████████████████ create 660 mebossuser mebossuser # ████████ / user
████████████████████████████████████ }
```

```
##### 0000, 000 0 ##### 000000 00000, 000000 000000 0  
##### 000000000000, 0000000, 0 000000 rsync.
```


SLIDE 22

XXXXXXXXXX

```
import logging
logging.basicConfig(filename='example.log', level=logging.DEBUG)
logging.debug("Let's see if it works")
logging.info("For your information")
logging.warning("I warn you!")
logging.error("Something bad happened pls help!")
```

```

##### filename, ##### level #####
#####.
#####
##### DEBUG, #####, #####

```

[illegible]

SLIDE 23

■■■■■■■■■■ ■■■■■■■■■■ logging

```
■■■■■■■■■■ logging — ■■■■■■■■■■ ■ ■■■■■■■■ ■■ ■■■■■■■■■■ ■■■■■■■■■■
■■■■■■■■■■■:
```

```

##### (loggers) — #####, #####
#####;

```

■■■■■■■■■■ (handlers) — ■■■■■■■■■■ ■■ ■■■■■■■■/■■■■■■ ■■■■■■■■■■;

■■■■■■■■■ (filters) — ■■■■■■■■■■ ■■■■■■ ■■■■■■ ■■■■■■■■■■ ■■■■, ■■■■■■
■■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■ ■■■■■■■■■■;

[illegible][illegible]

SLIDE 24

Logging Flow

SLIDE 25

#####

, #####
#####:

```
import logging counter = '3rd' logging.warning('I am warning you for the %s time!', counter)
```

→ WARNING:root:I am warning you for the 3rd time!

, ##
#####:

```
logging.basicConfig(format='%(asctime)s %(levelname)s %(message)s') logging.warning('Right about time!')
```

→ 2023-08-21 13:44:38,417 WARNING Right about time!

levelname
message, ##### asctime. #####
#####-##### basicConfig() (#####).

```
logging.basicConfig(format='%(asctime)s %(message)s', datefmt='%d.%m.%Y %H:%M:%S')  
logging.warning('is when this event was logged.')
```

→ 21.08.2023 13:54:51 is when this event was logged.

SLIDE 26

#####

logging.basicConfig() ##### logging ##### Python #####
, #####
format #####
#####.

LogRecord attributes.

#####

logging.basicConfig(), ##### logging #####
#####:

[TABLE CONTENT]

|

%(asctime)s | #####

%(levelname)s | #####

%(levelno)s | #####

%(message)s | #####

%(name)s | ##### (#####)

%(filename)s | #####

%(module)s | #####

%(lineno)d | #####

%(funcName)s | #####

%(process)d | ID #####

%(processName)s | #####

```
formatter = logging.Formatter( '%(asctime)s | %(levelname)s | %(message)s' ) logger =  
logging.getLogger('my_logger') handler = logging.StreamHandler()  
handler.setFormatter(formatter) logger.addHandler(handler)
```

SLIDE 27

■ ■ ■ ■ ■ ■ ■ (loggers)

```
##### Logger #####
# #####
# #####
# #####
# #####
```

```

##### Logger #####, file #####
##### «.» #####, file #####
##### file.error, file.access. #####
##### root (##### root, #####
#####).
```

```
logger = logging.getLogger(__name__)
```

root

SLIDE 28

■■■■■■■■ / Logger

[illegible]

Logger.addHandler(), Logger.removeHandler(), Logger.addFilter(), Logger.removeFilter() —
 These methods are used to add or remove handlers and filters to the logger. The **addHandler()** method adds a new handler to the logger, while **removeHandler()** removes an existing handler. The **addFilter()** method adds a new filter to the logger, while **removeFilter()** removes an existing filter. The logger will only log messages that pass through all the filters and are handled by at least one handler.

Logger.debug(), Logger.info(), Logger.warning(), Logger.error(), Logger.critical() —

```
Logger.log() — Prints the log message to the console.
Usage: Logger.log(message, [options]).
Options:
  - message (string): The log message to print.
  - options (object): An object containing options for the log message.
    - type (string): The log type (e.g., 'info', 'error', 'warn').
    - category (string): The log category (e.g., 'network', 'ui').
    - timestamp (boolean): Whether to include a timestamp in the log message.
```

[illegible]

`logging.getLogger(loggername)` — **Return a logger object for the logger with the given name.** If the logger has not been created, a new one is created, and the hierarchy of loggers is built up. If the logger has already been created, this method returns the previously created logger object. The logger name must be a string, and the root logger is the logger with the name `root`.

SLIDE 29

Handlers (handlers)

Handlers are objects that implement the `Handler` interface. They are used to format and output log messages. The `Handler` interface defines methods for setting the log level, setting the formatter, and adding or removing filters. The `Handler` interface is implemented by several classes, including `StreamHandler`, `FileHandler`, `RotatingFileHandler`, `TimedRotatingFileHandler`, `SMTPHandler`, and `SysLogHandler`.

Handlers are created and configured as follows:

`setLevel()` — Sets the log level for the handler. The log level is a constant that determines the severity of the log messages that will be output. The log level is set by passing a constant to the `setLevel()` method. The constants are `DEBUG`, `INFO`, `WARNING`, `ERROR`, and `CRITICAL`.

`setFormatter()` — Sets the formatter for the handler. The formatter is an object that formats log messages. The formatter is set by passing an object to the `setFormatter()` method.

`addFilter()`, `removeFilter()` — Adds or removes a filter from the handler. A filter is an object that filters log messages. The filter is added or removed by passing an object to the `addFilter()` or `removeFilter()` method.

[TABLE CONTENT]

Handler | Description

`StreamHandler` | Writes log messages to a stream (e.g., `sys.stdout` or `sys.stderr`). The stream is specified by the `stream` argument.

`FileHandler` | Writes log messages to a file. The file is specified by the `filename` argument.

`RotatingFileHandler` | Writes log messages to a file, rotating the file when it reaches a certain size. The file is specified by the `filename` argument.

`TimedRotatingFileHandler` | Writes log messages to a file, rotating the file at a certain time. The file is specified by the `filename` argument.

`SMTPHandler` | Sends log messages via email. The email address is specified by the `email` argument.

`SysLogHandler` | Writes log messages to the system log (syslog).

SLIDE 30

logging.Formatter (formatters)

logging.Formatter(fmt=None, datefmt=None, style='%')

logging.Formatter(fmt=None, datefmt=None, style='%')

logging.Formatter(fmt=None, datefmt=None, style='%')

logging.Formatter(fmt=None, datefmt=None, style='%')

logging.Formatter(fmt=None, datefmt=None, style='%')

logging.Formatter(fmt=None, datefmt=None, style='%')

LogRecord:

'%' — logging.Formatter (formatters)

'{' — logging.Formatter (formatters)

'\$' — logging.Formatter (formatters)

logging.Formatter (formatters) strftime = '%(asctime)s' %(name)s %(levelname)s >

%(message)s' # logging.Formatter (formatters) datefmt = '%Y-%m-%d %H:%M:%S' #

logging.Formatter (formatters) formatter = logging.Formatter(fmt=strftime, datefmt=datefmt)

→[2020-06-08 07:42:59] [logger] [DEBUG] > debug message →[2020-06-08 07:42:59] [logger] [INFO]

> info message

SLIDE 31

[illegible]

XXXXXXXXXXXX XXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:

```

##### fileConfig().

```

```
dictConfig({
    'context_processors': (
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.messages.context_processors.messages',
    ),
    'middleware': (
        'django.middleware.common.CommonMiddleware',
        'django.contrib.sessions.middleware.SessionMiddleware',
        'django.middleware.csrf.CsrfViewMiddleware',
        'django.contrib.auth.middleware.AuthenticationMiddleware',
        'django.contrib.messages.middleware.MessageMiddleware',
    ),
    'staticfiles': {
        'BACKEND': 'django.contrib.staticfiles.backends.FileSystemStorageBackend',
    },
})
```

SLIDE 32

[REDACTED] [REDACTED] [REDACTED] ([REDACTED])

```
import logging # ■■■■■■■■■■ logger = logging.getLogger('exampleLogger')
logger.setLevel(logging.INFO) # ■■■■■■■■■■ console_handler = logging.StreamHandler()
console_handler.setLevel(logging.INFO) # ■■■■■■■■■■
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
%(message)s') # ■■■■■■■■■■
console_handler.setFormatter(formatter) # ■■■■■■■■■■
logger.addHandler(console_handler) # ■■■■■■■■■■ (■■■■■ DEBUG)
logger.debug('debug message') logger.info('info message')
logger.warning('warn message') logger.error('error message') logger.critical('critical message')
```

SLIDE 33

XXXXXXXXXXXXXXXXXXXX XXXXXX XXXXXXXX-XXXXXX

```

#####  #####  #####  # #####, #####
# ##### # #, ##### ##### #
#####  #####  fileConfig() #####
#####  dictConfig().

```

Python.

```
import logging import logging.config # ■■■■■■■■■■ ■■■■■■■■
logging.config.fileConfig('logging.conf') # ■■■■■■■■ ■■■■■■■■ logger =
logging.getLogger('exampleLogger') # ■■■■■■■■■■■■■■■■ logger.debug('debug message')
logger.info('info message') logger.warning('warn message') logger.error('error message')
logger.critical('critical message')
```

SLIDE 34

logging.conf

```
[loggers] keys=root,exampleLogger [handlers] keys=consoleHandler [formatters] keys=simpleFormatter
[logger_root] level=DEBUG handlers=consoleHandler [logger_simpleExample] level=DEBUG
handlers=consoleHandler qualname=simpleExample propagate=0 [handler_consoleHandler]
class=StreamHandler level=DEBUG formatter=simpleFormatter args=(sys.stdout,)
[formatter_simpleFormatter] format=%(asctime)s - %(name)s - %(levelname)s - %(message)s
```

SLIDE 35

logging.config.dictConfig

[illegible]

XXXXXXXXXXXX XXXXXXXXXXXX XXXXXX XXXXXXX:

[illegible]

██████████ ██████████

XXXXXXXXXX XX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX

XXXXXXXXXXXXXXXXXXXX XXXX XXXXXXXXXXXXXXXXXXXXXXX.

■■■■■■■ ■■■■■■ ■■■■■■■■ ■ ■ YAML:

```
handlers: console: class : logging.StreamHandler formatter: brief level : INFO filters: [allow_foo] stream
: ext://sys.stdout file: class : logging.handlers.RotatingFileHandler formatter: precise filename:
logconfig.log maxBytes: 1024 backupCount: 3
```

SLIDE 36

JSON-XXXXXXXXXXXXXXXXXXXX

```
{
  "version": 1,
  "disable_existing_loggers": false,
  "formatters": {
    "simple": {
      "format": "%(asctime)s - %(name)s - %(levelname)s - %(message)s"
    },
    "extra": {
      "format": "%(asctime)s %(name)s %(filename)s %(lineno)s %(funcName)s %(levelname)s %(message)s"
    }
  },
  "handlers": {
    "console_handler": {
      "class": "logging.StreamHandler",
      "level": "DEBUG",
      "formatter": "simple",
      "stream": "ext://sys.stdout"
    },
    "info_file_handler": {
      "class": "logging.FileHandler",
      "level": "INFO",
      "formatter": "simple",
      "filename": "info.log",
      "encoding": "utf8"
    },
    "error_file_handler": {
      "class": "logging.FileHandler",
```

```
"level": "ERROR",
"formatter": "extra",
"filename": "errors.log",
"encoding": "utf8"
}
},
"loggers": {
  "MyApp": {
    "level": "WARNING",
    "handlers": ["console_handler"],
    "propagate": false
  },
  "MyApp.MyClass1": {
    "level": "DEBUG",
    "handlers": ["console_handler", "info_file_handler", "error_file_handler"],
    "propagate": false
  },
  "MyApp.MyClass2": {
    "level": "ERROR",
    "handlers": ["error_file_handler"],
    "propagate": false
  }
},
"root": {
  "level": "WARNING",
  "handlers": ["console_handler"]
}
}
```

SLIDE 37

dictConfig

```
import json
import logging
from logging.config import dictConfig

class MyAppLogger:
    def __init__(self):
        dictConfig(logging.config.dictConfig(self.config))
        self.logger = logging.getLogger('MyApp')

    def get_logger(self, name):
        return logging.getLogger('MyApp.' + name)

class MyClass1:
    def __init__(self):
        self.logger = MyAppLogger().get_logger(self.__class__.__name__)

class MyClass2:
    def __init__(self):
        self.logger = MyAppLogger().get_logger(self.__class__.__name__)

def test(object):
    object.logger.debug('debug message')
    object.logger.info('info message')
    object.logger.warning('warn message')
    object.logger.error('error message')
    object.logger.critical('critical message')

test(MyAppLogger())
test(MyClass1())
test(MyClass2())
```


SLIDE 38

■■■■■■■■■■ ■■■■■■ ■■■■■■■■■■

■■■■■■■■:

2023-08-21 17:09:40,562 - MyApp - WARNING - warn message

2023-08-21 17:09:40,563 - MyApp - ERROR - error message

2023-08-21 17:09:40,563 - MyApp - CRITICAL - critical message

2023-08-21 17:09:40,564 - MyApp.MyClass1 - DEBUG - debug message

2023-08-21 17:09:40,565 - MyApp.MyClass1 - INFO - info message

2023-08-21 17:09:40,565 - MyApp.MyClass1 - WARNING - warn message

2023-08-21 17:09:40,566 - MyApp.MyClass1 - ERROR - error message

2023-08-21 17:09:40,566 - MyApp.MyClass1 - CRITICAL - critical message

info.log:

2023-08-21 17:16:19,753 - MyApp.MyClass1 - INFO - info message

2023-08-21 17:16:19,756 - MyApp.MyClass1 - WARNING - warn message

2023-08-21 17:16:19,756 - MyApp.MyClass1 - ERROR - error message 2023-08-21 17:16:19,756 -
MyApp.MyClass1 - CRITICAL - critical message

errors.log:

2023-08-21 17:17:18,172 MyApp.MyClass1 main.py 93 test ERROR error message

2023-08-21 17:17:18,172 MyApp.MyClass1 main.py 94 test CRITICAL critical message

2023-08-21 17:17:18,173 MyApp.MyClass2 main.py 93 test ERROR error message

2023-08-21 17:17:18,173 MyApp.MyClass2 main.py 94 test CRITICAL critical message

SLIDE 39

■■■■■■■■■■ ■ ■■■■■■■■■■

■■■■■■■ MyApp (■ ■■■■■■■■ MyAppLogger) ■■■■■■ ■■■■■■
■■■■■■■■■■■ console_handler, ■■■■■■■■■■ ■■■■■■■■■■ ■■
■■■■■■■■. console_handler ■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■■■■■ DEBUG, ■■ ■■■■
■■■■■■■ MyApp — WARNING, ■■■■■■■■ ■■ ■■■■■■■■ ■■■■■■ ■■■■■■■■■■ ■■■■■■ ■■■■
■■■■■■■■■■■ — WARNING, ERROR ■ CRITICAL.

■■■■■■■ MyApp.MyClass1 (■ ■■■■■■■■ MyClass1) ■■■■■■ ■■■■
■■■■■■■■■■■■■■: console_handler, info_file_handler, error_file_handler. ■ ■■■■■■■■
■■■■■■■■■■■ ■■■■ ■■■■■■■■■■, ■■■■ ■■■■ ■ ■■■■■■■■, ■■■■■■■■■■■■ ■■■■■■
■■■■■■■■ DEBUG. ■ info.log ■■■■■■■■■■ ■■■■■■■■■■■■ INFO ■ ■■■■■ ■
■■■■■■■■■■■■■■■ ■ ■■■■■■■■ info_file_handler, ■ ■ errors.log — ERROR ■ CRITICAL.

■■■■■■■ MyApp.MyClass2 (■ ■■■■■■■■ MyClass2) ■■■■■■■■■■
■■■■■■■ ERROR ■ CRITICAL ■ errors.log.

SLIDE 40

«■■■■■■■■■■ ■■■■■■» ■■■■■■ ■■■■ ■■■■■■■■

■■■■■ (■■■■■■■■■■■■■■■■■■■■■) / ■■■■■■■■■■ ■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■:

■■■■■■■■■■ ■■■■:

datetime

```
■■■■ ■■■■■■■■ ■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■ uppercase: USER_ADDED,
DOMAIN_DELETED, MAIL_SENT, PASSWORD_RECOVERY_REQUEST, SMS_SEND_ERROR ■
■■.■■.
```

service / script name

hostname

██████████ ████████████████████ ████████, ████████ █████ ████████████ / ████████████, ██████████:

```
user_id / service_id — ██████████ ██████ ███-██ ████████████████████, ███ █████ █████
██████████
```

1. **REDAKCE** – vyplnění údajů, které jsou vyznačeny červenou vlnovkou / **REDAKCE**
 2. **REDAKCE**, vyplnění údajů, které jsou vyznačeny červenou vlnovkou:

DATA

```
2023-08-27 12:12:12 | USER_ADDED | mobappbackend | cl23.mycloud.ru | 23455 | 0 |
{"Name":"Ivan Ivanov","email":"ivanoff@gmail.com","regip":"1.2.3.4"}
```

```
2023-08-27 12:12:13 | MAIL_SENT | userspam.py | cl2.mycloud.ru | 11455 | 249499 |
{"email": "prtrfff@gmail.com", "title": "████████████████████████████████████████!"}
```

■■■■■■■■■■■

SLIDE 41

[illegible]

user_id

key_id

■■■■■■■■■■ ■■■■:

ext_params

SLIDE 42

Loguru

[illegible]

SLIDE 43

How do we know if loguru is logging?

loguru has a `loguru.logger.isEnabledFor(logging)` method.

loguru has a `loguru.logger.add_handler(handler)` method to add a handler.

loguru has a `loguru.logger.remove_handler(handler)` method to remove a handler.

loguru has a `loguru.logger.get_level_of(logger)` method to get the level of a logger.

loguru has a `loguru.logger.setLevel(level)` method to set the level of a logger.

loguru has a `loguru.logger.add_filter(filter)` method to add a filter.

loguru has a `loguru.logger.add_handler(handler, level=logging.INFO)` method to add a handler with a level.

loguru has a `loguru.logger.add_handler(handler, level=logging.INFO, filter=filter)` method to add a handler with a level and a filter.

SLIDE 44

■■■■■■■■■■■■■■■■■■■■ Loguru

- ✓ Ready to use out of the box without boilerplate

```
from loguru import logger logger.debug("That's it, beautiful and simple logging!")
```

```
2023-11-13 20:29:32.531 | DEBUG | __main__:11 - That's it, beautiful and simple logging!
```

- ✓ No Handler, no Formatter, no Filter: one function to rule them all

How to add a handler? How to set up logs formatting? How to filter messages? How to set level?

```
logger.add(sys.stderr, format="{time} {level} {message}", filter="my_module", level="INFO")
```

- ✓ Easier file logging with rotation / retention / compression

```
logger.add("file_{time}.log") logger.add("file_1.log", rotation="500 MB") # Automatically rotate too big
file logger.add("file_2.log", rotation="12:00") # New file is created each day at noon
logger.add("file_3.log", rotation="1 week") # Once the file is too old, it's rotated
logger.add("file_X.log", retention="10 days") # Cleanup after some time logger.add("file_Y.log",
compression="zip") # Save some loved space
```

SLIDE 45

■■■■■■■■■■■■■■■■■■■■ Loguru

- ✓ Modern string formatting using braces style

```
logger.info("If you're using Python {}, prefer {feature} of course!", 3.6, feature="f-strings")
```

- ✓ Exceptions catching within threads or main

```
@logger.catch def my_function(x, y, z):    # An error? It's caught anyway!    return 1 / (x + y + z)
```

- ✓ Pretty logging with colors

```
logger.add( sys.stdout, colorize=True, format="{time} {message}")
```


SLIDE 46

```
logger.add("out.log", backtrace=True, diagnose=True) def func(a, b): return a / b def nested(c): try:
    func(5, c) except ZeroDivisionError: logger.exception("What?!") nested(0)
```

■■■■■■■■■■■■■■■■■■■■ Loguru

✓ Fully descriptive exceptions

```
2018-07-17 01:38:43 | ERROR | __main__:nested:10 What?
```

Traceback (most recent call last):

File "test.py", line 12, in

nested(0)

■

> File "test.py", line 8, in nested

func(5, c)

■■ ■ 0

■

File "test.py", line 4, in func

return a / b

■■ ■ 0

■ 5

ZeroDivisionError: division by zero

SLIDE 47

■■■■■■■■■■■■■■■■■■■■ Loguru

✓ Lazy evaluation of expensive functions

Sometime you would like to log verbose information without performance penalty in production, you can use the `opt()` method to achieve this.

```
logger.opt(lazy=True).debug( "If sink level <= DEBUG: {x}", x=lambda: expensive_function(2**64)) # By
the way, "opt()" serves many usages logger.opt(exception=True).info( "Error stacktrace added to the
log message (tuple accepted too)") logger.opt(colors=True).info("Per message colors")
logger.opt(record=True).info( "Display values from the record (eg. {record[thread]})")
logger.opt(raw=True).info("Bypass sink formatting\n") logger.opt(depth=1).info( "Use parent stack
context (useful within wrapped functions)") logger.opt(capture=False).info( "Keyword arguments not
added to {dest} dict", dest="extra")
```

SLIDE 48

✓ Better datetime handling

The standard logging is bloated with arguments like `datefmt` or `msecs`, `%(asctime)s` and `%(created)s`, naive datetimes without timezone information, not intuitive formatting, etc. Loguru fixes it:

```
logger.add("file.log", format="{time:YYYY-MM-DD at HH:mm:ss} | {level} | {message}")
```

✓ Suitable for scripts and libraries

```
# For scripts config = { "handlers": [ {"sink": sys.stdout, "format": "{time} - {message}"},  
{"sink": "file.log", "serialize": True}, ], "extra": {"user": "someone"} } logger.configure(**config) # For  
libraries, should be your library's `__name__` logger.disable("my_library") logger.info("No matter added  
sinks, this message is not displayed") # In your application, enable the logger in the library  
logger.enable("my_library") logger.info("This message however is propagated to the sinks")
```

SLIDE 49

- ✓ Entirely compatible with standard logging

```
handler = logging.handlers.SysLogHandler(address=('localhost', 514)) logger.add(handler)
```

- ✓ Personalizable defaults through environment variables

```
# Linux / OSX export LOGURU_FORMAT="{time} | {message}" # Windows setx  
LOGURU_DEBUG_COLOR " "
```

- ✓ Convenient parser

```
pattern = r"(?P.*) - (?P[0-9]+) - (?P.*)" # Regex / named groups caster_dict =  
dict(time=dateutil.parser.parse, level=int) # Transform matching groups for groups in  
logger.parse("file.log", pattern, cast=caster_dict): print("Parsed:", groups) # {"level": 30,  
"message": "Log example", "time": datetime(2018, 12, 09, 11, 23, 55)}
```

SLIDE 50

✓ Exhaustive notifier

Loguru can easily be combined with the great notifiers library (must be installed separately) to receive an e-mail when your program fail unexpectedly or to send many other kind of notifications:

Pushover, SimplePush, Slack, Gmail, Telegram, Gitter, Pushbullet, Join, Zulip, Twilio, Pagerduty, Mailgun, PopcornNotify, StatusPage.io, iCloud, VictorOps (Splunk)

```
import notifiers params = { "username": "you@gmail.com", "password": "abc123", "to":  
"dest@gmail.com" } # Send a single notification notifier = notifiers.get_notifier("gmail")  
notifier.notify(message="The application is running!", **params) # Be alerted on each error message  
from notifiers.logging import NotificationHandler handler = NotificationHandler("gmail",  
defaults=params) logger.add(handler, level="ERROR")
```

✓ 10x faster than built-in logging

Zero-cost logger.

SLIDE 51

```
import sys from loguru import logger # [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED]
[REDACTED] [REDACTED] [REDACTED] [REDACTED] log_format = "{time:YYYY-MM-DD HH:mm:ss.SSS}
| " + \
                "{message} | {extra[user_id]} | " + \
                "{extra[key_id]}
| {extra[ext_params]}" config = {
                "handlers": [ {"sink": sys.stdout, "format":
log_format} ],
                "extra": { "user_id": 0, "key_id": 0, "ext_params": " " } }
logger.configure(**config) # [REDACTED] [REDACTED] [REDACTED] logger.info('SOME_ACTION')
logger.info('ANOTHER_ACTION', key_id = 777) logger.info('ACTION1', user_id = '123',
ext_params = {'par1': 123, 'param2': ['A','B',456]}) logger.info('ACTION2', user_id =
'123', key_id = 77, ext_params = {'par1': 123, 'par2': 'Z'})
```

Loguru custom fields

```
2023-11-13 22:35:12.579 | SOME_ACTION | 0 | 0 |
2023-11-13 22:35:12.579 | ANOTHER_ACTION | 0 | 777 |
2023-11-13 22:35:12.579 | ACTION1 | 123 | 0 | {'par1': 123, 'param2': ['A', 'B', 456]}
2023-11-13 22:35:12.579 | ACTION2 | 123 | 77 | {'par1': 123, 'par2': 'Z'}
```

██████: ████████████-████. ████████████████████ ████████:

user_id

key_id

XXXXXXXXXXXX .

ext_params

SLIDE 52

Создадим таблицу `test` в БД `test`

Синтаксис: `CREATE TABLE <имя таблицы> (<список колонок>, <тип данных>);`

Создадим таблицу `test` в БД `test` с колонками `id` (тип `int`) и `name` (тип `varchar(255)`)

Пример: `CREATE TABLE test (id int, name varchar(255))`

Создадим таблицу `test` в БД `test` с колонками `id` (тип `int`) и `name` (тип `varchar(255)`)

В таблице `test` создадим индекс `idx_name` по колонке `name` (тип `varchar(255)`). `logrotate` — это утилита для управления ротацией логов. Она позволяет настраивать, как часто и как часто ротировать логи, а также куда их сохранять. В конфигурационном файле `/etc/logrotate.conf` можно указать, какие логи ротировать, и как это делать. Например, можно указать, что логи, заканчивающиеся на `.log`, должны ротироваться раз в день. Это можно сделать, добавив в файл `/etc/logrotate.conf` следующую строку: `logrotate /etc/logrotate.conf`

Синтаксис: `logrotate <путь к файлу>`

Пример: `logrotate /etc/logrotate.conf` (где `/etc/logrotate.conf` — это путь к файлу конфигурации `logrotate`)

Создадим таблицу `test` в БД `test` с колонками `id` (тип `int`) и `name` (тип `varchar(255)`)

Синтаксис: `CREATE TABLE <имя таблицы> (<список колонок>, <тип данных>);` (clickhouse `CREATE TABLE`):

Создадим таблицу `test` в БД `test` с колонками `id` (тип `int`) и `name` (тип `varchar(255)`)

Синтаксис: `CREATE TABLE <имя таблицы> (<список колонок>, <тип данных>);`

Пример: `CREATE TABLE test (id int, name varchar(255))` (где `test` — это имя таблицы, а `id` и `name` — это имена колонок)

SLIDE 53

Clickhouse + web logs + ■■■■■■




SLIDE 54

XXXXXXXXXXXXXXXXXXXXXXXXXXXX: ELK

ELK – Elasticsearch, Logstash, Kibana. Elasticsearch – это распределенная поисковая система, которая позволяет хранить и искать данные в реальном времени. Logstash – это инструмент для сбора, фильтрации и загрузки данных в Elasticsearch. Kibana – это веб-интерфейс для Elasticsearch, который позволяет визуализировать данные и управлять кластером.

E = Elasticsearch —   Apache Lucene.

L = Logstash — **■■■■■■■■■■ ■■■ ■■■■■■ ■■■■■■: ■■■■■■■■ ■■■■■■ ■■**
■■■■■■■■■ ■■■■■■■■■■■■, ■■■■■■■■■■, ■■■■■■■■■■■■■■■■■■ ■ ■■■■■■■■■■ ■
■■■■■■■ ■■■■■■ ■■■■■■■■■■■■.

K = Kibana Kibana — . Kibana . Kibana .

■■■■ ■■■■■■■■■■ ■■■■■ ELK?

[illegible]























Elasticsearch ■■■■■■■■■■ ■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■
■■■■■ ■ ■■■.

□□□□□□:

██████████ ██████████ ████████████████████, "████ ████████████████" █████ ███████████████████ ██████████.

SLIDE 55

Graylog

Graylog —                           

[illegible]

THE UNITED STATES DEPARTMENT OF JUSTICE
FEDERAL BUREAU OF INVESTIGATION
WASHINGTON, D. C. 20535

■■■■■ Graylog

■■■ open source ■■■■■■■■.

[illegible]

ELK

SLIDE 56

Sentry

Sentry — **exception handling** (exception), **logging** **framework**.

It **allows** **you** **to** **capture** **exceptions** **and** **log** **them** **in** **your** **application**. **It** **also** **allows** **you** **to** **capture** **errors** **in** **your** **application**, **including** **JavaScript**, **PHP**, **Node.js**, **Python**, **Ruby**, **C #**, **Java**, **Go**, **React**, **Angular**, **Vue**, **JavaScript** **and** **more**.

It **also** **allows** **you** **to** **capture** **errors** **in** **your** **application**.

It **also** **allows** **you** **to** **capture** **errors** **in** **your** **application**:

It **also** **allows** **you** **to** **capture** **errors** **in** **your** **application**, **including** **JavaScript**, **PHP**, **Node.js**, **Python**, **Ruby**, **C #**, **Java**, **Go**, **React**, **Angular**, **Vue**, **JavaScript** **and** **more**. **It** **also** **allows** **you** **to** **capture** **errors** **in** **your** **application**, **including** **JavaScript**, **PHP**, **Node.js**, **Python**, **Ruby**, **C #**, **Java**, **Go**, **React**, **Angular**, **Vue**, **JavaScript** **and** **more**. **It** **also** **allows** **you** **to** **capture** **errors** **in** **your** **application**, **including** **JavaScript**, **PHP**, **Node.js**, **Python**, **Ruby**, **C #**, **Java**, **Go**, **React**, **Angular**, **Vue**, **JavaScript** **and** **more**.

```
import logging, sentry_sdk from sentry_sdk.integrations.logging import LoggingIntegration
sentry_logging = LoggingIntegration( level=logging.INFO, event_level=logging.ERROR )
sentry_sdk.init( dsn="https://exampleKey@o0.ingest.sentry.io/0", integrations=[ sentry_logging, ], )
```

SLIDE 57

■■■■■■■■ ■■■■■■

https://github.com/xtrueman/prog_instruments/blob/main/Logging.md

logging

Logging HOWTO

Loggint Cookbook