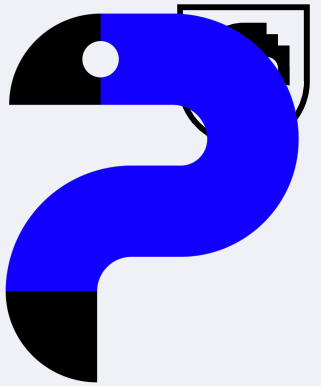
Code Academy



13 paskaita. Loginimas

1 LYGIS

2023

Aurimas Aleksandras Nausėdas

Šiandien išmoksite

01 Susipažinsime su logging biblioteka

02 Įrašyti klaidas į failus

03 Susipažinsime su taisyklingu klaidų logginimu

Kokie yra loginimo pranešimų lygiai

- **DEBUG** Detailed information, typically of interest only when diagnosing problems.
- **INFO** Confirmation that things are working as expected.
- **WARNING** An indication that something unexpected happened, or indicative of some problem in the near future (e.g. ‘disk space low’). The software is still working as expected.
- **ERROR** Due to a more serious problem, the software has not been able to perform some function.
- **CRITICAL** A serious error, indicating that the program itself may be unable to continue running.

```python
def dalyba(a, b):
    return a / b

a = 10
b = 5

padalinom = dalyba(a, b)
print(f"Dalyba: {a} / {b} = {padalinom}")

# Dalyba: 10 / 5 = 2.0
```

Kaip tikriname kodą PRINT funkcijų pagalba

```python
import logging

def dalyba(a, b):
    return a / b

a = 10
b = 5

padalinom = dalyba(a, b)
logging.warning(f"Dalyba: {a} / {b} = {padalinom}")

# WARNING:root:Dalyba: 10 / 5 = 2.0
```

Kaip tikrinti kodą loginimo pagalba

Informacija neturi būti warning lygio

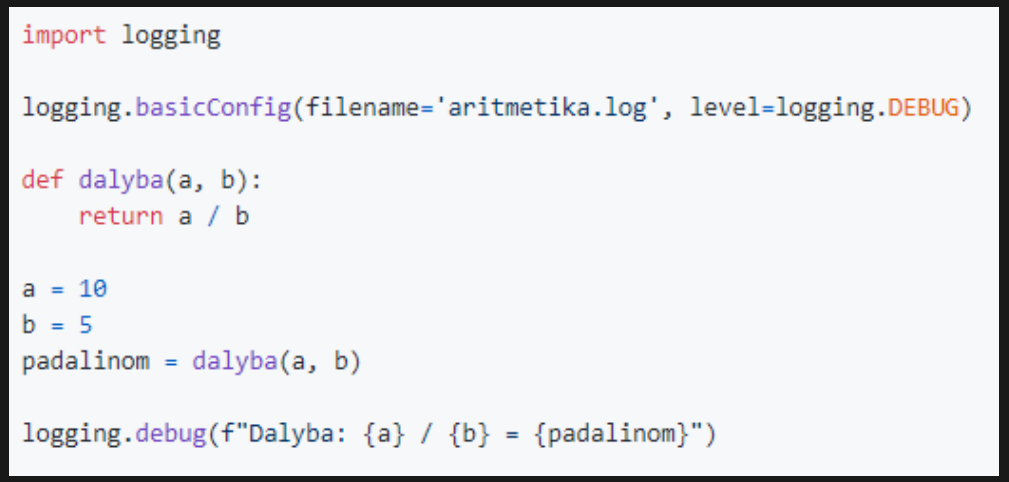
```python
def dalyba(a, b):
    return a / b

a = 10
b = 5

padalinom = dalyba(a, b)
print(f"Dalyba: {a} / {b} = {padalinom}")

# Dalyba: 10 / 5 = 2.0
```

Kaip nustatyti loginimo pranešimų lygį



```python
import logging

logging.basicConfig(filename='aritmetika.log', level=logging.DEBUG)

def dalyba(a, b):
    return a / b

a = 10
b = 5
padalinom = dalyba(a, b)

logging.debug(f"Dalyba: {a} / {b} = {padalinom}")
```

Kaip nustatyti loginimą į failą

Kaip nustatyti loginimo pranešimų formatą

```python
import logging

logging.basicConfig(filename='aritmetika.log', level=logging.DEBUG, format='%(asctime)s:%(levelname)s:%(message)s')

def dalyba(a, b):
    return a / b

a = 10
b = 5

padalinom = dalyba(a, b)
logging.debug(f"Dalyba: {a} / {b} = {padalinom}")
```

```python
import logging

logging.basicConfig(filename='asmenys.log', level=logging.INFO, format='%(asctime)s:%(levelname)s:%(message)s')

class Asmuo:

    def __init__(self, vardas, pavarde):
        self.vardas = vardas
        self.pavarde = pavarde
        logging.info(f"Sukurtas darbuotojas: {self.vardas} {self.pavarde}")

tadas = Asmuo("Tomas", "Kucinskas")
rokas = Asmuo("Rokas", "Radzevicius")
```

Loginimas su objektais

Failas asemeys.py

```python
import logging

logging.basicConfig(filename='asmenys.log', level=logging.INFO, format='%(asctime)s:%(levelname)s:%(message)s')

class Asmuo:

    def __init__(self, vardas, pavarde):
        self.vardas = vardas
        self.pavarde = pavarde
        logging.info(f"Sukurtas darbuotojas: {self.vardas} {self.pavarde}")

tadas = Asmuo("Tomas", "Kucinskas")
rokas = Asmuo("Rokas", "Radzevicius")
```

Failas aritmetika.py

```python
import logging
import asmenys

logging.basicConfig(filename='aritmetika.log', level=logging.DEBUG, format='%(asctime)s:%(levelname)s:%(name)s:%(message)s')

def dalyba(a, b):
    return a / b

a = 10
b = 5

padalinom = dalyba(a, b)
logging.debug(f"Dalyba: {a} / {b} = {padalinom}")
```

Savo logerio sukūrimas

Problemos pavyzdys

Failas asemeys.py

```python
import logging

logger = logging.getLogger(__name__)
file_handler = logging.FileHandler('asmenys.log')
logger.addHandler(file_handler)
logger.setLevel(logging.DEBUG)
formatter = logging.Formatter('%(asctime)s:%(levelname)s:%(name)s:%(message)s')
file_handler.setFormatter(formatter)

class Asmuo:

    def __init__(self, vardas, pavarde):
        self.vardas = vardas
        self.pavarde = pavarde
        logger.info(f"Sukurtas darbuotojas: {self.vardas} {self.pavarde}")

tadas = Asmuo("Tomas", "Kucinskas")
rokas = Asmuo("Rokas", "Radzevicius")
```

Failas aritmetika.py

```python
import logging
logger = logging.getLogger(__name__)
file_handler = logging.FileHandler('aritmetika.log')
logger.addHandler(file_handler)

logger.setLevel(logging.DEBUG)

formatter = logging.Formatter('%(asctime)s:%(levelname)s:%(name)s:%(message)s')
file_handler.setFormatter(formatter)

def dalyba(a, b):
    return a / b

a = 10
b = 5

padalinom = dalyba(a, b)
logger.info(f"Dalyba: {a} / {b} = {padalinom}")
```

Savo logerio sukūrimas

Sprendimo pavyzdys

```python
import logging
logger = logging.getLogger(__name__)
file_handler = logging.FileHandler('aritmetika.log')
logger.addHandler(file_handler)

logger.setLevel(logging.DEBUG)

formatter = logging.Formatter('%(asctime)s:%(levelname)s:%(name)s:%(message)s')
file_handler.setFormatter(formatter)

stream_handler = logging.StreamHandler()
stream_handler.setFormatter(formatter)
logger.addHandler(stream_handler)

def dalyba(a, b):
    return a / b

a = 10
b = 5

padalinom = dalyba(a, b)
logger.info(f"Dalyba: {a} / {b} = {padalinom}")
```

Kaip loginti klaidas ir į failą ir į konsolę

```python
import logging
logger = logging.getLogger(__name__)
file_handler = logging.FileHandler('aritmetika.log')
logger.addHandler(file_handler)

logger.setLevel(logging.DEBUG)

formatter = logging.Formatter('%(asctime)s:%(levelname)s:%(name)s:%(message)s')
file_handler.setFormatter(formatter)

stream_handler = logging.StreamHandler()
stream_handler.setFormatter(formatter)
logger.addHandler(stream_handler)

def dalyba(a, b):
    try:
        rezultatas = a / b

    except ZeroDivisionError:
        logger.exception("Dalyba is nulio")
    else:
        return rezultatas

a = 20
b = 0

padalinom = dalyba(a, b)
logger.info(f"Dalyba: {a} / {b} = {padalinom}")
```

Kaip loginti klaidas

Užduotis nr. 1

Sukurti funkcijas, kurios:
- Gražintų visų paduotų skaičių sumą (su *args argumentu)
- Gražintų paduoto skaičiaus šaknį (panaudoti math.sqrt())
- Gražintų paduoto sakinio simbolių kiekį (su len())
- Gražintų rezultatą, skaičių x padalinus iš y

Nustatyti standartinį logerį (logging) taip, kad jis:
- Saugotų pranešimus į norimą failą
- Saugotų INFO lygio žinutes
- Pranešimai turi būti tokiu formatu: data/laikas, lygis, žinutė

Kiekviena funkcija turi sukurti INFO lygio log pranešimą apie tai, ką atliko, pvz.:

```
logging.info(f"Skaiciu {args} suma lygi: {sum(args)}")
```

Užduotis nr. 2

Perdaryti 1 užduoties programą, kad:

- Į šaknies funkciją padavus string tipo argumetrą, į log failą būtų išsaugoma išimties klaida su norimu tekstu
- Į dalybos funkciją antrą argumentą padavus 0, į log failą būtų išsaugoma išimties klaida su norimu tekstu

Patarimas: panaudoti try/except/else, logging.exception()

Užduotis nr. 3

Perdaryti 2 užduoties programą, kad:

- Būtų sukurtas savo logeris, kuris fikstuotus visus anksčiau aprašytus pranešimus
- Sukurtas logeris ne tik išsaugotų pranešimus faile, bet ir atvaizduotų juos konsolėje

Namų darbas

Užbaigti klasėje nepadarytas užduotis

Naudinga informacija

Išspręsti paskaitos uždaviniai (įkelti pirmadienį)	https://github.com/aurimas13/Python-Beginner-Course/tree/main/Programs
LogRecord attributes Loginimo bibliotekos atributų informacija	https://docs.python.org/3/library/logging.html#logrecord-attributes