hillel

Python Basic

Заняття 13

import **logging**

Debug (10): найнижчий рівень логування, призначений для налагоджувальних повідомлень, для виведення діагностичної інформації про програму.

Info (20): цей рівень призначений для виведення даних про фрагменти коду, що працюють так, як очікується.

Warning (30): цей рівень логування передбачає виведення попереджень, він застосовується для запису відомостей про події, куди програміст зазвичай звертає увагу. Такі події можуть призвести до проблем під час роботи програми. Якщо явно не задати рівень логування, за замовчуванням використовується саме warning.

Error (40): цей рівень логування передбачає виведення відомостей про помилки - про те, що частина програми працює не так, як очікується, про те, що програма не змогла правильно виконатися.

Critical (50): цей рівень використовується для виведення відомостей про дуже серйозні помилки, наявність яких загрожує нормальному функціонуванню всієї програми. Якщо не виправити таку помилку, це може призвести до того, що програма припинить роботу.

level: це рівень, на якому потрібно починати логування. Якщо його встановлено в info — це означає, що всі повідомлення з рівнем debug ігноруються.

filename: цей параметр вказує на об'єкт обробника файлу. Тут можна вказати ім'я файлу, який потрібно здійснювати логування.

filemode: це необов'язковий параметр, який вказує режим, в якому передбачається працювати з файлом журналу, заданим параметром filename. Установка файламоде значення w (write, запис) призводить до того, що логи перезаписуються при кожному запуску модуля. За замовчуванням параметр filemode встановлено значення a (append, приєднання), тобто — файл будуть потрапляти записи з усіх сеансів роботи програми.

Logging from multiple modules

If your program consists of multiple modules, here's an example of how you could organize logging in it:

```
# myapp.py
import logging
import mylib

def main():
    logging.basicConfig(filename='myapp.log', level=logging.INFO)
    logging.info('Started')
    mylib.do_something()
    logging.info('Finished')

if __name__ == '__main__':
    main()
```

```
# mylib.py
import logging

def do_something():
    logging.info('Doing something')
```

If you run *myapp.py*, you should see this in *myapp.log*:

```
INFO:root:Started
INFO:root:Doing something
INFO:root:Finished
```

import **Enum**

Переліки – це набори символічних імен, пов'язаних із унікальними константними значеннями. Вони використовуються для створення простих типів даних, таких як пори року, тижня, види зброї в грі, планети, оцінки або дні. За згодою імена перерахувань починаються з великої літери і використовуються в однині.

Атрибути класу Enum конвертуються в екземпляр при парсингу. Кожен екземпляр має параметр name, у якому зберігається назва, а також value, у якому зберігається встановлене значення.

import **dataclass**

Python

```python
from dataclasses import dataclass

@dataclass
class Position:
    name: str
    lon: float
    lat: float
```

Вбудовані методи :

- __init__
- __repr__
- __str__
- __eq__

Defaults value

Python

```python
from dataclasses import dataclass

@dataclass
class Position:
    name: str
    lon: float = 0.0
    lat: float = 0.0
```

Immutable Data Classes

Python

```python
from dataclasses import dataclass

@dataclass(frozen=True)
class Position:
    name: str
    lon: float = 0.0
    lat: float = 0.0
```

Inheritance

Python

```python
from dataclasses import dataclass
from typing import List

@dataclass(frozen=True)
class ImmutableCard:
    rank: str
    suit: str

@dataclass(frozen=True)
class ImmutableDeck:
    cards: List[ImmutableCard]
```

Python

```python
from dataclasses import dataclass

@dataclass
class Position:
    name: str
    lon: float = 0.0
    lat: float = 0.0

@dataclass
class Capital(Position):
    country: str  # Does NOT work
```