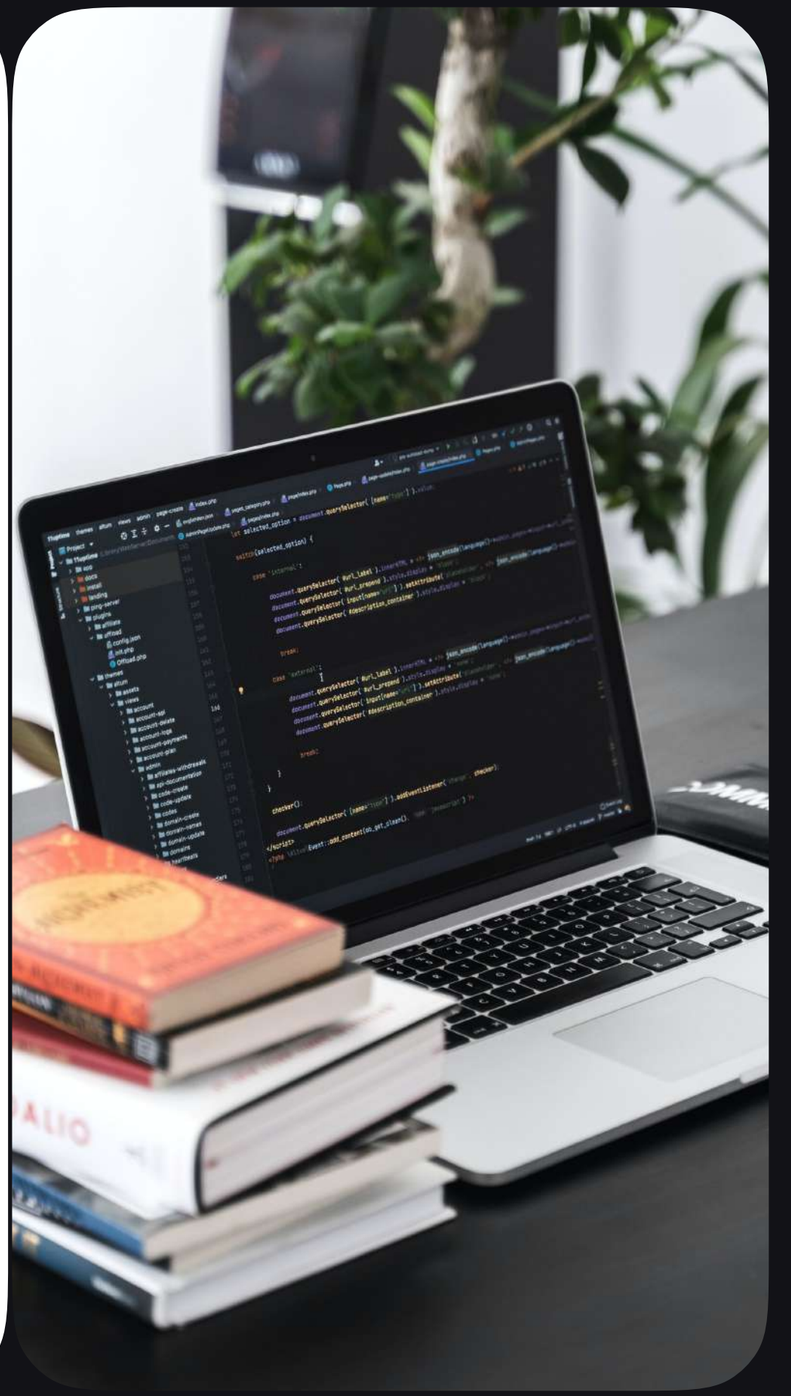
Модуль 3 Занятие 6

Модуль schedule. Решение задач



Модуль
datetime

Работа с датой
и временем

Модуль
schedule

Решение задач

Модуль datetime

Для работы с датой и временем в Python существует встроенный модуль datetime. Модуль предоставляет для работы следующие классы:

- datetime.date — класс для работы с датами
- datetime.time — класс для работы с временем
- datetime.datetime — комбинация даты и времени
- datetime.timedelta – класс для работы с временными интервалами, разница между двумя датами

Класс date

```
import datetime


date = datetime.date(2023, 1, 1)


print(type(date), date)
print(date.day, date.month, date.year)
```

Вывод:

```
<class 'datetime.date'> 2023-01-01
1 1 2023
```

Для создания объектов даты используется класс date. Экземпляр объекта этого класса представляет дату в формате ГГГГ-ММ-ДД.

При создании объекта конструктору этого класса нужны три обязательных аргумента: year, month и day.

Текущая дата

```
import datetime


date = datetime.date.today()
day = datetime.date.today().day
month = datetime.date.today().month
year = datetime.date.today().year


print(type(date), date)
print(day, month, year)
```

Вывод:

```
<class 'datetime.date'> 2023-01-31
31 1 2023
```

Получить текущую дату можно с помощью функции today() класса date.

Unix время

Unix-время (Unix time) — это способ представления определенной даты и времени, используемый Linux, macOS и других операционных системах. Определяется как количество секунд, прошедших с полуночи (00:00:00) 1 января 1970 года.

Unix Timestamp (временная метка) — это конкретная дата и время, представленная числом с точностью до секунды.

Например:

(00:00:00) 1 января 2023 года будет представлено числом 1672520400

Временная метка

```
import datetime


date = datetime.date.fromtimestamp(1672520400)


print(date)
```

Получить дату
из временной метки можно
с помощью функции
fromtimestamp() класса date.

Вывод:

```
2023-01-01
```

Модуль datetime

Рассмотрим некоторые методы класса:

- **isocalendar()** — возвращает кортеж (год, неделя, день недели)
- **isoformat()** — возвращает строковое представление объекта даты
- **isoweekday()** — возвращает номер дня недели (понедельник — 1, воскресенье — 7)
- **strftime()** — возвращает строковое представление даты используя форматирование

Пример

```
import datetime

weekdays = [
    'понедельник',
    'вторник',
    'среда',
    'четверг',
    'пятница',
    'суббота',
    'воскресенье',
]

today = datetime.date.today()
weekday = today.weekday()

print(f'Сегодня: {weekdays[weekday]}')
```

Рассмотрим пример задачи, которую можно решить с помощью модуля datetime.

Воспользуемся методом weekday(), а не isoweekday() (т.к. нам нужна нумерация с 0) и выведем название текущего дня недели.

Класс time

```
import datetime


time1 = datetime.time()
time2 = datetime.time(15, 30, 23)


print(type(time1), time1)
print(type(time2), time2)
print(time2.hour, time2.minute, time2.second)
```

Вывод:

```
<class 'datetime.time'> 00:00:00
<class 'datetime.time'> 15:30:23
15 30 23
```

Класс time служит для создания времени без привязки ко дню.

Экземпляр объекта этого класса представляет дату
в формате ЧЧ:ММ:СС.

При создании объекта можно указать необязательные аргументы:
hour, minute, second
(по умолчанию равны 0).

Класс datetime

```
import datetime


datetime1 = datetime.datetime(2023, 1, 1)
datetime2 = datetime.datetime(2023, 1, 1, 12, 30, 15)
print(type(datetime1), datetime1)
print(datetime2)
```

Вывод:

```
<class 'datetime.datetime'>
2023-01-01 00:00:00
2023-01-01 12:30:15
```

Класс datetime содержит информацию как о дате, так и о времени и является комбинацией классов date и time.

Экземпляр объекта этого класса представлен в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.

При создании объекта необходимо указать обязательные аргументы: year, month и day и необязательные: hour, minute, second (по умолчанию равны 0).

Текущая дата и время

```
import datetime


now = datetime.datetime.now()


print(type(now))
print(now)
```

Получить текущую дату можно с помощью функции now() класса datetime.

Вывод:

```
<class 'datetime.datetime'>
2023-01-31 11:11:34.457292
```

Методы класса datetime

Рассмотрим некоторые методы класса:

- **combine**() — получает объекты date и time и возвращает объект datetime
- **date**() — возвращает объект класса date
- **time**() — возвращает объект класса time
- **timestamp**() — возвращает значение временной метки
- **fromtimestamp**() — возвращает объект datetime соответствующий временной метке

Форматирование

```
import datetime


now = datetime.datetime.now()


print(now)
print(now.strftime('%d.%m.%Y'))
print(now.strftime('%H:%M'))
```

Вывод:

```
2023-01-31 11:34:37.550387
31.01.2023
11:34
```

Для форматирования объектов datetime можно использовать функции strptime и strftime.

- **strftime**() — возвращает строковое представление даты и времени, используя заданное форматирование
- **strptime**() — возвращает объект datetime, соответствующий строке

Формат даты

Обозначение	Значение	Пример
%a	Сокращенное название дня недели	Sun, Mon, …, Sat
%A	Полное название дня недели	Sunday, Monday, …, Saturday
%w	День недели в виде целого числа	0, 1, …, 6
%d	День месяца в виде целого числа, дополненного нулями	01, 02, …, 31
%b	Сокращенное название месяца	Jan, Feb, …, Dec
%B	Полное название месяца	January, February, …, December
%m	Месяц в виде целого числа, дополненного нулями	01, 02, …, 12
%y	Год без века в виде целого числа, дополненного нулями	00, 01, …, 99
%Y	Год с веком в виде целого числа	0001, 0002, …, 2013, 2014, …, 9998, 9999

Формат времени

Обозначение	Значение	Пример
%H	Час (24-часовой формат) в виде целого числа, дополненного нулями	Sun, Mon, …, Sat
%I	Час (12-часовой формат) в виде целого числа, дополненного нулями	Sunday, Monday, …, Saturday
%M	Минуты в виде целого числа, дополненного нулями	0, 1, …, 6
%S	Секунды в виде целого числа, дополненного нулями	01, 02, …, 31
%f	Микросекунды в виде целого числа, дополненного нулями до 6 цифр	Jan, Feb, …, Dec

Пример

```
import datetime

s = '2023/01/31'

date = datetime.datetime.strptime(s, '%Y/%m/%d')
print(date.strftime('%d.%m.%Y'))
```

Вывод:

```
31.01.2023
```

Создадим объект datetime из строки с помощью метода strptime() и выведем в привычном формате с помощью метода strftime().

Класс timedelta

```
import datetime


date = datetime.datetime(2022, 12, 31)
delta = datetime.timedelta(days=1)
new_date = date + delta


print(type(delta), delta)
print(new_date)
```

Вывод:

```
<class 'datetime.timedelta'> 1 day, 0:00:00
2023-01-01 00:00:00
```

Класс timedelta используется для операций с датами.

При создании объекта timedelta можно указать необязательные аргументы: days, seconds, microseconds, minutes, hours, weeks (по умолчанию равны 0).

Класс timedelta

```
import datetime


date = datetime.datetime(2023, 1, 1)
now = datetime.datetime(2023, 1, 31)
delta = now - date


print(type(delta))
print(delta)
print(delta.days)
```

Вывод:

```
<class 'datetime.timedelta'>
30 days, 0:00:00
30
```

Класс timedelta также является результатом для вычисления разницы дат.

Объект класса timedelta имеет атрибуты: days, seconds, microseconds.

Пример

```
import datetime


date = datetime.datetime(2023, 2, 1)
month = date.month
while date.month == month:
    if date.isoweekday() in [6, 7]:
        print(f'{date.strftime("%d.%m")} - выходной!')
    date += datetime.timedelta(days=1)
```

Распечатаем список выходных дней одного месяца.

Вывод:

```
04.02 - выходной!
05.02 - выходной!
11.02 - выходной!
12.02 - выходной!
18.02 - выходной!
19.02 - выходной!
25.02 - выходной!
26.02 - выходной!
```

Пример

```
import datetime
import time

now_minute = datetime.datetime.now().minute
while True:
    now = datetime.datetime.now()
    if now.minute != now_minute:
        print(now.strftime('%H:%M'))
        now_minute = now.minute
    time.sleep(1)
```

Создадим с помощью модуля datetime периодическую задачу: будем выводить на экран время каждую минуту.

Модуль schedule

Модуль schedule — это удобный планировщик для выполнения периодических заданий.

schedule позволяет периодически запускать функции Python через заранее определенные интервалы времени, используя простой и удобный для человека синтаксис.

Для того, чтобы начать использовать модуль, выполните его установку, выполнив команду в терминале:

```
pip install schedule
```

Выполнение периодических заданий

```
import time

import schedule


def job():
    print("Работаю...")


schedule.every(10).seconds.do(job)
schedule.every(10).minutes.do(job)
schedule.every().hour.do(job)
schedule.every().day.at("10:30").do(job)
schedule.every(5).to(10).minutes.do(job)
schedule.every().monday.do(job)
schedule.every().wednesday.at("13:15").do(job)
schedule.every().minute.at(":17").do(job)


while True:
    schedule.run_pending()
    time.sleep(1)
```

schedule.every() — создает новое периодическое задание, для которого доступны методы:

- **at**("HH:MM") — выполнение задания в определенное время
- **do**(job_func, *args, **kwargs) — указание функции job_func, которая будет выполняться.
- **to**(latest) — запуск задания со случайным интервалом. Например, every(A).to(B).seconds выполняет задания каждые N секунд, так что A <= N <= B

schedule.run_pending() — запускает все запланированные задания.

Пример

```
import datetime
import time

import schedule



def print_time():
    now = datetime.datetime.now()
    print(now.strftime('%H:%M'))



schedule.every().minutes.do(print_time)

while True:
    schedule.run_pending()
    time.sleep(1)
```

Будем выводить на экран время каждую минуту с помощью модуля schedule.

Документация по модулю

Больше примеров по использованию модуля можно найти на странице официальной документации.

https://schedule.readthedocs.io/en/stable/

Итоги

- Для работы с датой и временем в Python существует встроенный модуль datetime
- Класс datetime.date — для работы с датами
- Класс datetime.time — для работы с временем
- Класс datetime.datetime — комбинация даты и времени
- Класс datetime.timedelta — для работы с временными интервалами и операциями с датами
- Для выполнения периодических заданий существует сторонний модуль schedule с простым и удобным, для человека, синтаксисом.