Робота з мережею в Python

№ уроку: 2 **Курс:** Python Advanced

Засоби навчання: PyCharm

Огляд, мета та призначення уроку

Навчити студентів писати мережеві застосунки мовою Python. Дати базові знання мережевої моделі OSI та її рівнів, а також навчити створювати власні socket-сервери/клієнти.
Вивчити протокол HTTP. Дати базові знання цього протоколу та його особливостей, а також навчити обробляти HTTP-повідомлення з використанням мови Python.

Вивчивши матеріал цього заняття, учень зможе:

- Розуміти основи мережевої моделі OSI.
- Розробляти UDP/TCP socket-сервери.
- Розробляти UDP/TCP socket-клієнти як для власних, так і для сторонніх socket-серверів.
- Розуміти протокол HTTP.
- Розуміти типи запитів, їхні особливості й обмеження.
- Формувати запити до HTTP-серверів та обробляти відповіді від них.
- Використовувати стандартну бібліотеку urllib.
- Використовувати сторонню бібліотеку requests.
- Створювати програми мовою Python, які дають змогу автоматизувати обробку відповідей.

Зміст уроку

1. Основи мережевої моделі OSI та її рівнів.
2. Поняття адресації (IP та Port) та socket.
3. Опис протоколу UDP.
4. Створення UDP клієнта/сервера.
5. Створення TCP клієнта/сервера та порівняння з UDP.
6. Блокувальні та не блокувальні режими роботи socket.
7. Підхід ООП під час створення socket-серверів, використовуючи бібліотеку socketserver.
8. Створення простого socket-сервера засобами фреймворку Twisted.
9. Що таке протокол HTTP, як використовувати та основні типи запитів.
10. Поняття заголовків і статус-кодів.
11. Типи HTTP-запитів та їхні особливості.
12. Створення socket для демонстрації заголовків і відповідей сервера.
13. Стандартна бібліотека мови Python – urllib.
14. Бібліотека requests.
15. Конфігурація бібліотеки urllib: розмір, pull з'єднань і режими роботи.

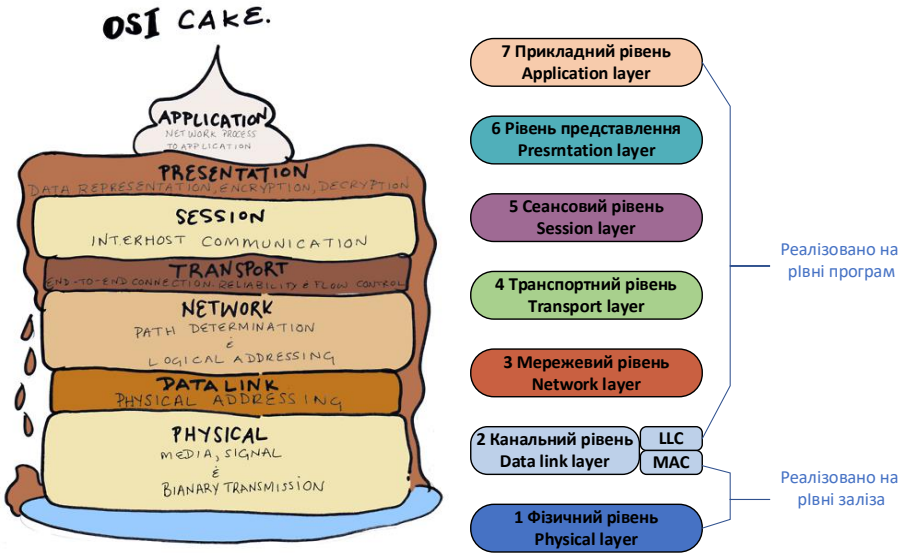
Резюме

Наприкінці 1970-х почали з'являтися громадські мережі передання даних. Передання та приймання даних – основне завдання мереж. Передавати інформацію можна різними алгоритмами. Щоби клієнт працював у мережі, він має знати та використовувати конкретний алгоритм. Щоби кожен виробник не створював свій алгоритм, було ухвалено рішення стандартизувати процес передання даних у мережах. Так у 1978 році з'явилася модель **OSI**.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Модель OSI докладно описує роботу мережі. Її можна порівняти з тортом у 7 шарів. Кожен шар цієї моделі вирішує своє конкретне завдання: логічну адресацію, шифрування чи передання даних оптоволокном. Кожен рівень є частиною єдиної моделі, як і шари створюють єдиний торт.



Модель OSI складається з таких рівнів: прикладного, представницького, сеансового, транспортного, мережевого, канального та фізичного. Щоразу дані в мережі проходять шлях від прикладного до фізичного рівня й назад. Докладніше про OSI можна почитати у Вікіпедії або на сайті Infocisco.

Вирахувати за IP?
Повна адреса компонента мережі складається з IP і порту та записується так: 127.0.0.1:80 – це ваша локальна IP адреса + порт 80. 127.0.0.1 – це самоідентифікація комп'ютера, його локальна адреса, те ж саме, що для людини «Я».

Я: ping 127.0.0.1, Комп'ютер: Так, звичайно, я знаю його. Це я.
Не за кожним IP є реальний комп'ютер. Є сірі та білі адреси. Докладніше про них тут.

Протокол передання даних – набір угод логічного рівня, які визначають обмін даними між різними програмами.

Ці угоди задають єдиний спосіб передання повідомлень та обробки помилок.

Користувач може відкривати сторінки у WWW, завантажувати відео, використовувати Torrent, дивитися потокове відео (стрім) – для різних завдань підбираються ефективні протоколи.

З кожним роком протоколів стає більше. Але є два основних, які створюють основу для решти всіх протоколів: UDP і TCP.

CyberBionic systematics
CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

UDP-протокол є ненадійним протоколом передання даних. Пакети, які надіслані з використанням цього протоколу, можуть бути втрачені або може бути порушений їхній порядок під час отримання. Немає жодної гарантії 100-відсоткового доставлення пакетів.

TCP-протокол є надійним протоколом передання даних. Цей протокол здійснює так зване рукостискання й перед переданням/отриманням даних має встановити з'єднання з кінцевим вузлом мережі.

Сокет – це програмний інтерфейс для забезпечення інформаційного обміну між процесами.

Є клієнтські та серверні сокети. Серверний сокет прослуховує певний порт, а клієнтський під'єднується до сервера. Після встановлення з'єднання починається обмін даними.

Розглянемо це на простому прикладі статті Сокети в Python для початківців. Уявімо великий зал із безліччю невеликих вікон, за якими стоять касири. Є й порожні вікна, за якими немає нікого. Ті самі вікна – це порти. Там, де стоїть касир, – це відкритий порт, за яким стоїть якийсь застосунок. Тобто: якщо ви підійдете до віконця з номером 9090, то вас привітають і запитають, чим можуть допомогти.

Так само і з сокетами. Створюється застосунок, який прослуховує свій порт. Коли клієнт встановлює з'єднання із сервером на цьому порту, саме цей застосунок буде відповідальний за роботу цим клієнтом. Ви ж не підійдете до одного віконця, а кричатимуть вам із сусіднього.

Після успішного встановлення з'єднання сервер і клієнт починають обмінюватися інформацією. Наприклад, сервер посилає вітання та пропозицію ввести будь-яку команду. Зі свого боку клієнт вводить команду, сервер її аналізує, виконує необхідні операції та віддає клієнту результат.

У стандартній бібліотеці мови Python є бібліотека socket, яка дає змогу створювати socket-сервери та socket-клієнти.

Для створення socket-серверів найзручнішим підходом є ООП. Це необов'язкова вимога, однак підхід ООП надає зручніший інтерфейс для оброблення запитів і нових під'єднань. Ми можемо використовувати стандартну бібліотеку socketserver для створення сокет-серверів у стилі ООП.

Використовуйте готові бібліотеки, які перевірені часом. Такі бібліотеки дадуть вам змогу створювати стабільні та надійні сокет-сервери/клієнти, написавши лише кілька рядків коду. Водночас дані бібліотеки надають можливість зручної конфігурації ваших сокетів: порти, блокування, таймаути тощо.
Хорошим продовженням цієї теми буде стаття: Сокети в Python для початківців.

HTTP-протокол є текстовим протоколом із певною структурою. Цей протокол – надбудова над socket, адже за своєю суттю ми відкриваємо socket-з'єднання та обмінюємося даними, закриваючи його після завершення.

HTTP-протокол реалізує 9 різних методів: OPTIONS, GET, HEAD, POST, PUT, PATCH, DELETE, TRACE, CONNECT. Кожен із методів розрахований для конкретної задачі, наприклад, GET – для отримання даних, а PUT – для оновлення даних. Кожен метод має свої особливості та відмінності.

Є важлива відмінність між POST-методом і GET-методом. Розмір даних GET, які передаються, обмежений методом і становить 8 кілобайтів. Обсяг даних, що передається методом POST, обчислюється мегабайтами. І як плюс – завжди можна збільшити це значення.

Дані, які передаються методом GET, передаються в URL-рядку браузера. Дані, що передаються методом POST, передаються в тілі запиту (стандартний потік введення).

Щоби зрозуміти, чи успішно завершився наш запит, ми можемо використовувати статус-коди. 200-ті статуси означають успішність запитів. Помилки сервера – 500-ті. Помилки даних, які передаються клієнтами, – 400-ті.

Мова Python містить у своїй стандартній бібліотеці інструменти для роботи з HTTP – бібліотека urllib.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Закріплення матеріалу

- Що таке мережева модель OSI та з чого вона складається?
- Чому в низці завдань краще використовувати протокол TCP замість UDP?
- У чому відмінність протоколів TCP та UDP?
- Який тип сокетів можна використовувати, якщо ви хочете налагодити взаємодію між двома програмами в межах одного сервера?
- За що відповідає константа socket.AF_INET?
- За що відповідає константа socket.AF_UNIX?
- Що таке HTTP-протокол і з чого він складається?
- Що таке заголовки?
- Які типи запитів бувають?
- Який метод необхідно використати для зміни даних про товар?
- Які відмінності між методами GET і POST?
- Який метод необхідно використовувати для передання логіну та пароля під час авторизації користувача?
- За що відповідають статус-коди діапазону 2xx?
- Якщо користувач надіслав некоректні дані на сервер, який статус-код потрібно повернути у відповіді?
- Який статус-код повернути, якщо користувач не пройшов авторизацію?
- Що таке pull з'єднань та як його можна налаштувати?
- Що таке блокувальний режим pull з'єднань і в чому його особливість?

Додаткове завдання

Завдання 1
Створити простий чат на основі протоколу TCP, який дасть змогу під'єднуватися кільком клієнтам та обмінюватися повідомленнями.

Завдання 2
Створіть HTTP-клієнта, який прийматиме URL ресурсу, тип методу та словник як передавальні дані (опціональний). Виконувати запит з отриманим методом на отриманий ресурс, передаючи дані відповідним методом, та друкувати на консоль статус-код, заголовки та тіло відповіді.

Самостійна діяльність учня

Завдання 1
Вивчіть основні поняття, розглянуті на уроці, а також особливості роботи з TCP та UDP протоколами в Python.

Завдання 2
Створіть UDP-сервер, який очікує на повідомлення про нові пристрої в мережі. Він приймає повідомлення певного формату, де буде ідентифікатор пристрою, і друкує нові під'єднання в консоль. Створіть UDP-клієнта, який надсилатиме унікальний ідентифікатор пристрою на сервер, повідомляючи про свою присутність.

Завдання 3
Створіть UNIX-сокет, який приймає повідомлення з двома числами, що розділені комою. Сервер має конвертувати рядкове повідомлення у два числа й обчислювати його суму. Після успішного обчислення повертати відповідь клієнту.

Завдання 4

CyberBionic systematics
CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Вивчіть основні поняття, розглянуті в уроці, а також особливості роботи з HTTP-протоколами в Python, використовуючи бібліотеки urllib та requests.

Завдання 5
Вивчіть докладніше та спробуйте можливості налаштування pull з'єднань і його режимів. Використовуючи утиліту ab, протестуйте ваші напрацювання (https://ua.wikipedia.org/wiki/ApacheBench).

Завдання 6
Використовуючи сервіс https://jsonplaceholder.typicode.com/, спробуйте побудувати різні типи запитів та обробити відповіді. Необхідно попрактикуватися з urllib та бібліотекою requests. Рекомендується спочатку спробувати виконати запити, використовуючи urllib, а потім спробувати реалізувати те саме, використовуючи requests.

Рекомендовані ресурси

Офіційний сайт Python (3.6)
https://docs.python.org/3.6/library/socket.html
Офіційний сайт Python (3.6):
https://docs.python.org/3.6/library/urllib.html
Офіційна документація бібліотеки requests:
http://docs.python-requests.org/en/master/

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com