**💡 Restful API에 대해 설명해주세요.**

Restful API는 HTTP 통신을 Rest 설계 규칙을 잘 지켜서 개발한 API를 Restful한 API라고 합니다.  
Rest 설계 규칙은 URI는 정보의 자원만 표현해야 하며, 자원의 상태와 행위는 HTTP Method에 명시하는걸 말합니다.  
**REST(REpresentational State Transfer)란?**

**REST의 정의**

"REpresentational State Transfer" 의 약자로,

자원을 이름(자원의 표현)으로 구분해 해당 자원의 상태(정보)를 주고 받는 모든 것을 의미합니다.

즉, **자원(resource)**의 **표현(representation)**에 의한 **상태 전달**을 뜻합니다.

* **자원** **:** 해당 소프트웨어가 관리하는 모든 것 ( 문서, 그림, 데이터, 해당 소프트웨어 자체 등 )
* **표현 :** 그 자원을 표현하기 위한 이름 ( DB의 학생 정보가 자원이면, 'students'를 자원의 표현으로 정함 )
* **상태 전달 :** 데이터가 요청되는 시점에 자원의 상태를 전달한다. ( JSON 혹은 XML을 통해 데이터를 주고 받는 것이 일반적 )

REST는 기본적으로 웹의 기존 기술과 HTTP 프로토콜을 그대로 활용하기 때문에,

**웹의 장점을 최대한 활용할 수 있는 아키텍처 스타일**입니다.

REST는 네트워크 상에서 Client와 Server 사이의 통신 방식 중 하나입니다.

**REST의 개념**

**어떤 자원에 대해 CRUD(Create, Read, Update, Delete) 연산을 수행하기 위해 URI(Resource)로**

**GET, POST 등의 방식(Method)을 사용하여 요청을 보내며, 요청을 위한 자원은 특정한 형태(Representation of Resource)로 표현됩니다.**

**💡 Restful API 기술은 어떠한 문제를 해결하기 위한 기술인지 설명해주세요.**

Restful API는 분산 시스템 간의 통신을 효율적이고 표준화된 방식으로 설계하기 위해 고안된 기술입니다.

\* HTTP 프로토콜을 기반으로 자원을 URI로 명확히 식별하고, 무상태성 설계를 통해 서버 부담을 줄이며 확장성을 확보한다. 또한 클라이언트와 서버를 독립적으로 개발하며, JSON,XML 등 여러 포맷을 통해 다양한 플랫폼에서 일관된 데이터를 주고받을 수 있도록 지원한다.

**프레임워크와 라이브러리의 차이에 대해 설명해주세요.**

이 둘의 차이점은 제어 흐름에 대한 주도권이 누구에게/어디에 있는가에 있습니다.  
프레임워크는 전체적인 흐름을 쥐고 있고, 개발자는 그 안에서 라이브러리에 대한 흐름을 쥐고 있습니다.  
여기서 개발자의 제어권을 프레임워크에게 넘김으로써 신경써야할 것을 줄일 수 있는데,  
이를 제어의 역전(Inversion of Control)이라 합니다.

**💡 Call By Value와 Call By Reference의 차이에 대해 설명해주세요.**

* **Call By Value(값에 의한 호출)** - 인자로 받은 값을 복사하여 처리하는 방식입니다.
  + 장점 - 값을 복사하여 처리하기 때문에 원래의 값이 보존된다.
  + 단점 - 복사하기 때문에 메모리 사용량이 증가한다.
* **Call By Reference(참조에 의한 호출)** - 인자로 받은 값의 주소를 참조하여 직접 저장해 값에 영향을 주는 방식입니다.
* 장점 - 복사하지 않고 직접 참조하기에 빠르다.
  + 단점 - 직접 참조를 하기에 원래의 값이 영향을 받는다.

**절차지향 프로그래밍과 객체지향 프로그래밍의 차이점에 대해 설명해주세요.**

* **절차지향 프로그래밍**
  + 물이 위에서 아래로 흐르는 것처럼 순차적인 처리를 중요시하는 프로그래밍 기법이다.
  + 대표적인 언어로 C언어가 있다.
  + 컴퓨터의 처리구조와 유사해 실행속도가 빠르다.
  + 코드의 순서가 바뀌면 동일한 결과를 보장하기 어렵다.
* **객체지향 프로그래밍**
  + 실제 세계의 사물들을 객체로 모델링하여 개발을 진행하는 프로그래밍 기법이다.
  + 대표적인 언어로 Java가 있다.
  + 캡슐화, 상속, 다형성 등과 같은 기법을 이용할 수 있다.
  + 절차지향 언어보다 실행속도가 느리다.

**절차지향 프로그래밍**은 기능중심으로 바라보는 방식으로 "무엇을 어떤 절차로 할 것인가?"가 핵심이 되며, 어떤 기능을 어떤 순서로 처리하는가에 대해 초점을 맞추고,

**객체지향 프로그래밍**은 기능이 아닌 객체 중심으로 바라보는 방식으로 "누가 어떤 일을 할 것인가?"가 핵심이며, 객체를 도출하고 각각의 역할을 정의해 나가는 것에 초점을 둔다.

**1) Call by reference란 무엇이고 보통 어떻게 쓰이는지 설명해 보세요.**

스프링은 자바에서 메서드 호출 시 "Call by value" 방식을 따릅니다. Call by value는 값에 의한 호출을 의미하며, 메서드에 변수를 전달할 때 해당 변수 값이 복사되어 사용됩니다. 따라서 사용자가 메서드 내에서 변수의 값을 변경하더라도 호출자의 변수는 변경되지 않습니다.

Call by reference (참조에 의한 호출)은 메소드에 변수를 전달할 때 변수의 참조(메모리 주소)가 전달되며, 메소드 내에서 변수를 수정하면 호출자의 변수도 변경됩니다. 스프링에서는 이러한 방식을 직접 사용하지 않고, 대신 객체를 전달하여 객체 내부의 상태를 변경할 수 있습니다. 이러한 경우 객체의 상태 변경과 관리가 더 효율적이며 예측 가능해집니다.

**2) Override와 Overload를 설명해 보세요.**

오버라이드(Override)는 상위 클래스의 메서드를 재정의하는 것을 말합니다. 메서드의 이름은 물론, 파라미터의 개수나 타입을 동일한 환경에서 주로 상위 클래스의 동작을 상속받는 하위 클래스에서 변경하기 위해 사용합니다.

오버로드(Overload)는 메서드의 이름은 같고 파라미터의 개수나 타입이 다른 함수를 정의하는 것을 말합니다. 리턴 값만을 다르게 갖는 오버로는 작성할 수 없습니다.

오버라이딩은 상속받은 메서드의 내용만 변경하는 것이고, 오버로딩은 같은 이름의 메서드를 여러 개 가지며 매개변수의 유형과 개수가 달라도 재정의할 수 있습니다.

**3) JVM이란 무엇이고 왜 필요한지 설명해 보세요.**

JVM(Java Virtual Machine)은 자바 프로그램을 실행하는 가상 머신입니다. 자바는 OS의 종속성에서 벗어나기 위해 사용되어, 각 OS에 맞는 JVM을 통해 같은 소스 코드로도 다른 OS에서 정상 실행이 가능합니다. GC를 통해 프로그램의 메모리 관리를 해 준다는 이점이 있습니다.

**5) 클래스와 인스턴스의 차이에 대해 설명해 보세요.**

클래스(Class)와 인스턴스(Instance)는 객체 지향 프로그래밍에서 중요한 개념입니다.

클래스(Class)는 객체의 속성(attribute)과 동작(behavior)을 정의하며, 객체를 생성하기 위한 템플릿 역할을 합니다.

객체를 생성하기 위한 일종의 설계도로서 여러 객체가 공유하는 특성을 정의합니다. 예를 들어, ‘핸드폰’ 클래스는 핸드폰 객체가 가져야 할 속성(색상, 모양, 모델 등)과 동작(터치, 전원 등)을 정의할 수 있습니다.

인스턴스(Instance)는 클래스를 기반으로 생성된 객체를 말합니다.

클래스가 설계도라면, 실제로 사용되는 데이터와 객체는 인스턴스 생성을 통해 만들어집니다. 클래스의 속성과 동작을 실제 값으로 가지고 있으며 각각 독립적으로 작동합니다. 예를 들어, ‘핸드폰’ 클래스에서 ‘삼성 갤럭시 S24’, ‘애플 아이폰 15’ 와 같은 여러 인스턴스를 생성할 수 있습니다.

클래스가 객체의 설계를 정의하고, 인스턴스는 클래스에 따라 생성된 실제 객체를 뜻합니다. 클래스에서 여러 인스턴스를 생성할 수 있고, 각 인스턴스는 클래스의 정의에 따라 각각 독립적인 데이터와 동작을 가지고 있습니다.

**10) Primary Key, Foreign Key에 대해 설명해 보세요.**

Primary Key(기본 키)와 Foreign Key(외래 키)는 서로 다른 개념입니다.

Primary Key(기본 키)는 한 개체(entity)를 고유하게 식별하는 것을 목표로 합니다. 각 열을 unique로 다루며 Not null 속성을 가집니다. Foreign Key(외래 키)는 다른 개체와의 관계 형성을 하기 위해 사용하는 것으로, 다른 테이블의 Primary Key(기본 키) 값을 참조합니다.

두 가지에 대해 좀 더 자세히 말씀드려 보겠습니다.

Primary Key(기본 키)는 데이터베이스 테이블에서 각 행(row)을 고유하게 식별하는 열(column) 또는 열의 조합을 말합니다. 키가 소속된 테이블에서 중복되지 않아야 하며, 모든 행은 기본 키 값을 가집니다. 주로 이 키가 식별자(identifier)로 사용됩니다. 각 행을 고유하게 식별하기 때문에 검색, 수정, 삭제, 구분 등의 작업에서 유용하게 사용됩니다.

Foreign Key(외래 키)는 한 테이블의 열(column)이 다른 테이블의 기본 키(primary key)를 참조하는 역할을 합니다. 다른 테이블과의 관계를 형성하여 데이터간의 일관성과 무결성(Integrity)을 유지하는 것에 사용합니다.

외래 키 제약 조건은 데이터베이스 시스템에서 외래 키 값을 검증하고 관리하는 것에 사용합니다. 조건은 부모 테이블(참조 테이블)의 값과 일치하지 않거나 해당 값이 null 상태일 때도 똑같이 적용됩니다.

**12) 브라우저의 작동 방식에 대해서 설명해 보세요.**

사용자 URL을 입력하면 브라우저는 해당 주소의 웹 서버로 HTTP 요청을 보냅니다. 서버는 받은 요청을 처리하여 HTML, CSS, JavaScript 파일 등을 브라우저에게 응답으로 보냅니다.

이후, 브라우저는 HTML을 패싱(passing)하여 돔(Document Object Model) 트리를 만들어 웹 페이지 구조를 나타냅니다. CSS 파일 역시 함께 패싱(passing)되어 CSSOM(CSS Object Model) 트리를 생성해 페이지 스타일을 정의합니다.

그 뒤, 돔(Document Object Model)과 CSSOM(CSS Object Model)을 결합하여 렌더 트리를 생성합니다. 이 트리는 페이지의 시각적 표현을 나타내며 각 요소별 크기와 위치 데이터를 포함하고 있습니다. 이를 기반으로 계산된 스타일과 레이아웃 정보를 통해 텍스트, 이미지, 색상 등 다양한 시각적 요소들을 포함한 페이지를 화면에 그려냅니다.

필요한 경우 자바 스크립트가 실행되어 동적 기능을 추가하거나 페이지를 변경할 수 있고, 모든 요소와 스크립트가 실행되면 페이지 로딩이 완료됩니다.

**17) JWT(Json Web Token)에 대해 설명해 보세요.**

JWT(JSON Web Token)는 인증된 사용자에게 토큰을 발급하고, 이를 통해 사용자 신원을 확인하는 프로세스에 사용합니다. Spring Security에서 JWT를 사용하여 사용자 인증 및 권한 부여를 구현할 수 있으며, 보안 및 인증을 효과적으로 관리할 수 있습니다.

JWT(JSON Web Token)의 발급 과정에 대해 설명해 보겠습니다.

첫 번째로, 사용자가 자격을 증명할 수 있는 아이디 및 패스워드를 제출해 로그인을 시도하면 UserDetailsService를 통해 사용자 정보를 검색합니다. 또한 PasswordEncoder를 사용하여 비밀번호 일치 여부를 확인하여 사용자를 인증하고 신원을 확인합니다.

두 번째로, 그 뒤 사용자가 인증된다면 서버는 사용자 정보와 권한이 포함된 JWT(JSON Web Token)를 생성합니다. 이 생성한 토큰에 서명 키를 이용하여 서명하고, 클라이언트로 반환합니다. JWT(JSON Web Token)는 보통 HTTP 헤더에 포함되거나 응답 본문을 통해 전달됩니다.

세 번째로, 이후 사용자가 엔드포인트로 요청을 보낼 시 JWT(JSON Web Token)는 요청 Authorization 헤더나 쿼리 매개변수를 통해 서버로 전송되며, 서버는 클라이언트에게서 받은 JWT의 서명을 검증하고 만료 시간 및 기타 클레임 정보를 확인하여 유효성을 검사합니다.

만일 유효한 토큰일 시 요청한 리소스 내역에 대한 액세스를 허용하거나 거부하는 결과를 응답하며, 클라이언트는 JWT(JSON Web Token)에 포함된 권한에 따라 요청 작업을 수행합니다.

**19)  Docker에 대해 설명해 보세요.**

Docker는 컨테이너화 기술로, 애플리케이션과 그 종속성을 패키지화하여 일관된 실행 환경을 제공합니다. Docker 이미지를 통해 애플리케이션을 배포하고, Docker 컨테이너를 통해 실행합니다.

환경 간의 일관성을 만들 수 있고 애플리케이션 배포 및 관리가 용이하다는 장점이 있습니다. 또한, 리소스 효율성이 높고, 빠른 배포가 가능합니다.

**20) REST API와 GraphQL의 차이점이 무엇인지 설명해 보세요.**

REST API와 GraphQL은 데이터 통신을 위한 두 가지 접근 방식입니다.

REST(Representational State Transfer) API는 리소스 기반의 접근 방식을 사용합니다.

각 URL이 고유한 리소스를 나타내는데요. HTTP 메서드(GET, POST, PUT, DELETE 등)를 사용하여 리소스를 조작합니다. 데이터는 JSON 또는 XML 형식으로 전송됩니다. 표준화된 방식으로 안정성과 캐싱을 지원이 가능하지만, 복잡한 데이터 요구 사항을 처리할 때 많은 요청이 필요할 수 있다는 단점이 있습니다.

GraphQL은 쿼리 언어를 사용하여 클라이언트가 필요한 데이터만을 요청할 수 있게 합니다. 단일 엔드포인트를 통해 다양한 쿼리를 지원하며, 데이터 구조를 클라이언트가 정의할 수 있습니다. 복잡한 쿼리와 다양한 데이터 요구를 유연하게 처리할 수 있지만, 서버 측에서 쿼리를 해석하고 최적화하는 데 추가적인 부담이 있을 수 있습니다.

**python의 특징에 대해 설명해주세요.**

* 파이썬은 인터프리터 언어로써, 코드를 작성한 후 컴파일 없이 즉싱 실행할 수 있습니다. 가독성이 높은 코드를 작성하기 쉬우며, 머신러닝, 데이터 분석 같은 다양한 분야에서 사용됩니다.

**인터프리터와 컴파일러에 대한 차이점을 설명해주세요.**

* 인터프리터는 프로그래밍의 소스 코드의 내용을 한번에 한줄씩 읽어들여서 실행합니다.
* 컴파일러는 전체 프로그램 코드를 스캔하여 소스 코드(고급 프로그래밍 언어)를 오브젝트 코드(어셈블리어(저급 프로그래밍 언어))로 바꾸어주는 역할을 합니다.
* 인터프리터는 소스 코드를 해석하는데는 적은 시간이 걸리지만 실행 시간은 느립니다. 반면 컴파일러는 소스코드를 해석하는데는 많은 시간이 걸리지만 한번 오브젝트 코드로 바꿔놓으면 실행 시간은 빠릅니다.

**pep 8에 대해서 아시나요? 읽어보셨다면 어떤 것들이 있나요?**

* pep는 파이썬의 코딩 규칙에 대한 제안서입니다.
* 들여쓰기와 tab, space를 혼합 사용 금지, 최대 줄 금지, 네이밍 규칙 등이 있습니다.
* 들여쓰기는 스페이스 4번, space와 tab은 혼합 사용을 금지해야 하며, 최대 줄 길이는 79자를 넘어가면 안됩니다. 넘어가게 된다면 백 슬래시를 사용합니다.
* 네이밍 규칙은 Class의 경우 앞 대문자, function의 경우는 소문자로 지정하고, function의 경우 언더바를 사용하여 단어를 나눕니다.

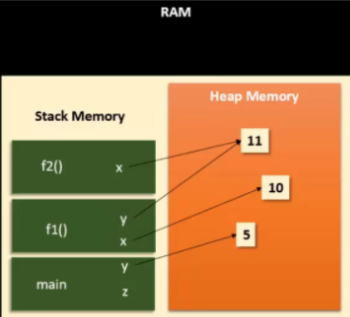
**list와 tuple의 공통점과 차이점에 대해서 설명해주세요.**

* 공통점으로는 둘 다 여러 데이터를 담을 수 있는 컨테이너형 변수입니다.
* 그리고 인덱스를 통해 특정 요소에 접근할 수 있으며, iterable합니다. 즉, for 문에 넣고 돌릴 수 있습니다.
* 차이점으로는 list는 mutable 하지만 tuble은 immutable합니다.
* 따라서 list는 딕셔너리의 key 값으로 쓸 수 없지만, tuple은 가능합니다.
* 왜냐면 딕셔너리의 key 값은 immutable한 객체만 올 수 있기 때문입니다.
* 그리고 속도의 차이점이 있습니다.
* 같은 수의 요소를 넣은 후, for 문을 돌리면 tuple이 list보다 속도가 빠릅니다.

**왜 tuple이 list보다 빠를까요?**

* list는 mutable이기 때문에 객체가 생성된 후 크기를 확정해야 하는 경우를 대비하여 추가 메모리 블록을 할당해줍니다.
* 반대로 tuple은 한번 생성하면 변경할 수 없어 크기가 고정적이기 때문에 최소 메모리 블록을 할당해줍니다.
* 그렇기 때문에 tuple이 list보다 속도가 더 빠릅니다.
* 디버깅 관련해서도 tuple의 immutable한 속성으로 인해 더 쉽게 추적할 수 있어서 디버그가 list 보다 쉽습니다.

**파이썬은 메모리 할당을 어떻게 할까요?**



* 파이썬의 method와 그 안에 있는 변수들은 stack memory에 저장됩니다.
* 힙 메모리에서는 객체들을 할당해줍니다.
* 파이썬의 모든 것은 객체이기 때문에 변수들에 대응하는 객체들은 모두 힙에 저장되고, stack 메모리에 저장되어 있는 변수들이 객체를 가리키게 됩니다.

**메모리 할당 해제하는 로직을 설명해주세요.**

* 우선 stack에 저장되어 있는 method가 리턴되면 해당 method의 영역의 할당이 해제됩니다.
* 할당이 해제되면서 힙에 있는 객체들을 가리키지 않게 되는데, 만약 다른 method의 변수들도 가리키지 않는다면 reference counting이 0이 되면서 객체도 자동으로 해제가 됩니다. 이것이 파이썬의 가비지 콜렉터입니다.

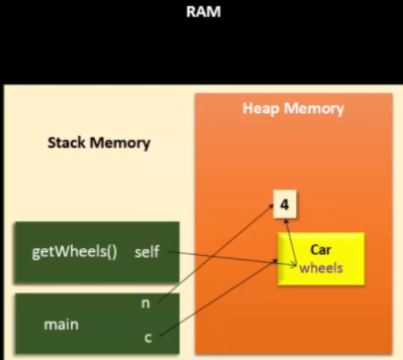
**가비지 콜렉터란?(GC)**

* **메모리를 자동으로 할당해주고 해제해주는 것**
* **즉, 위의 두 질문은 파이썬의 가비지 콜렉터의 로직을 설명하는 것이다.**
* **결론은 파이썬은 reference counting을 이용해 메모리를 관리한다.**

**파이썬에서 class, object, instance를 구분해서 설명해주세요.**

* 예시로 설명하겠습니다.
* 파이썬에서 class는 붕어빵의 틀과 같습니다.
* object(객체)는 이런 틀에서 만들어진 하나 하나의 붕어빵입니다.
* 그리고 붕어빵 틀이 여러개가 있고, 틀에 의해 만들어진 붕어빵도 여러개가 있는데, 어떠한 붕어빵 틀에서 만들어진 붕어빵인지 구분하기 위해 instance를 사용합니다.
* 즉, 붕어빵 틀1(class)에서 만들어진 붕어빵1(객체)은 붕어빵 틀1의 instance입니다.

**self는 무엇인가요?**



* 특정 class instance의 주소 값입니다.
* 특정 class로 객체를 만들면 그 class의 instance라고 불리우는데, self를 이용해 instance 값을 참조합니다.
* self는 stack memory에 저장되며, 특정 class의 instance를 참조해 그 안에 있는 attribute에 접근합니다.

**method는 무엇인가요?**

* class안에서 구현하는 함수를 말합니다.
* 위에 그림에서 보는것과 같이, stack memory에 저장되어 그 안에 있는 변수들이 해당 class 의 instance를 가리키게 됩니다.

**생성자란 무엇인가요?**

* 파이썬에서 **init** 이라는 메소드 이름을 가진 것을 생성자라고 합니다.
* 객체가 생성이 되면 자동으로 맨처음 호출돼 객체를 초기화해줍니다.

**파이썬 모든 것이 객체라고 하셨는데, 그렇다면 int 변수도 객체인가요?**

* 네. 파이썬은 x = 10처럼 변수를 할당하면 10이라는 int 객체를 만들어 놓고 변수 x가 그것을 가리킵니다.
* 그렇기에 다른 변수의 값이 10이 되면 만들어진 10의 객체의 주소를 가리키게 됩니다.

**immutable과 mutable에 대해서 설명해주세요.**

* mutable은 변경되는 객체이며 immutable은 변경되지 않는 객체입니다.
* mutable 객체로는 list, set, dict 정도가 있으며, 모든 객체를 각각 생성해서 참조합니다. 그렇기 때문에 안의 요소들이 같더라도 참조하는 메모리 주소는 다릅니다.  
  immutable 객체로는 int, float, tuple, str, bool 정도가 있으며, 값이 같은 경우에 변수에 상관없이 동일한 곳을 참조합니다.
* 그리고 immutable한 객체들은 값을 변경시킬수 없습니다.

**immutable과 mutable 둘 다 값이 변했을 때 메모리 주소는 어떻게 되나요?**

* immutable 같은 경우 변한 값에 대한 메모리 주소를 참조하게 됩니다. 그렇기 때문에 9에서 10으로 변한다면 10에 대한 메모리 주소를 참조하게 됩니다.
* mutable은 객체를 각각 생성해서 메모리 주소를 참조하기 때문에, 값이 변하더라도 참조하는 메모리 주소는 변하지 않습니다.

**str은 mutable 객체인데, 그렇다면 이것도 값이 변하더라도 같은 메모리 주소를 참조하겠네요?**

* 아닙니다. str 타입의 경우 항상 문자열이 같은지 보고 같은 곳을 참조할지 판단하기가 쉽지 않기 때문에 항상 같은 곳을 참조 하지는 않습니다.
* 예를 들어 같은 문자열로 객체를 생성한 경우 해당 문자열에 대한 같은 메모리 주소를 가리키지만, replace 메소드로 값을 변경한 객체와 변한 문자열을 수동으로 값을 넣어주면 둘의 가리키는 메모리 주소는 다릅니다.

**얕은 복사에 대해서 설명해주세요.**

* 얕은 복사는 = 연산자나 슬라이싱의 특수한 경우를 통해 이루어집니다.
* 같은 메모리 주소를 가리키기 때문에 mutable한 list의 경우 얕은 복사를 한 후 값을 변경시킨다면 둘 다 같은 값으로 변하게 됩니다.
* immutable한 객체들은 어차피 같은 값에 따라 같은 메모리 주소를 가리키기 때문에 얕은 복사를 하던 깊은 복사를 하던 상관 없이 얕은 복사를 따라가게 됩니다.

**슬라이싱의 특수한 경우는 무엇인가요?**

* mutable한 객체안에 immutable한 객체들로만 이루어진 것이 아닌, mutable한 객체가 있다면, 슬라이싱을 하더라도 얕은 복사가 이루어집니다.
* 전체적인 객체의 메모리 주소는 다르지만, 그 안의 mutable한 객체들은 서로 같은 메모리 주소를 가리킵니다.

**깊은 복사에 대해서 설명해주세요.**

* 슬라이싱이나 deepcopy 함수를 통해 이루어집니다.
* 깊은 복사는 mutable한 객체의 내부에 있는 객체 모두 새롭게 만들어줍니다.
* 그렇기 때문에 서로 다른 메모리 주소를 가리켜 안의 값들을 변경시키더라도 서로 영향을 주지 않습니다.

**list의 extend와 append의 차이점에 대해서 설명해주세요.**

* list의 extend는 삽입 대상의 iterable한 객체를 풀어서 각각의 요소로 확장해 삽입시켜줍니다.
* list의 append는 삽입하려는 객체 전체를 해당 list의 요소로 집어넣어줍니다.

**list의 extend와 + 연산자의 차이에 대해서 설명해주세요.**

* extend는 확장하려는 list의 주소값이 변하지 않은 상태로 확장이 되며, + 연산자는 두 리스트가 더해진 새로운 리스트가 반환이 됩니다.

**해당 코드의 결과물은 어떻게 나올거 같나요?**

**def** **extendList**(val, list=[]):

list.append(val)

**return** list

list1 = extendList(10)

list2 = extendList(123,[])

list3 = extendList('a')

print "list1 = %s" % list1

print "list2 = %s" % list2

print "list3 = %s" % list3

list1 = [10, 'a']

list2 = [123]

list3 = [10, 'a']

* 위의 결과로 나옵니다.
* list1과 list3가 같은 이유는 list1과 list3가 호출한 extendList에서 파라미터에 list의 기본값을 넣어주지 않았기 때문에 첫 호출에서 생성한 list의 주소를 가리키게 됩니다.
* 반면 list2는 파라미터에 별도의 빈 list를 넣어주었기 때문에, 다른 주소를 가지는 list에 123이라는 숫자를 넣습니다.

**위의 코드에서 우리가 원하는데로 각 리스트들이 다른 값을 가지게 하고 싶으면 어떻게 고쳐야 할까요?**

**def** **extendList**(val, list=None):

**if** list **is** None:

list = []

list.append(val)

**return** list

* list의 기본값을 None으로 설정해주고, 호출을 할 때 별도로 list 값을 주지 않는다면 새로운 list를 만들게 해서 서로 다른 주소를 가리키게 해줍니다.
* 이렇게 한다면 list1,2,3가 서로 다른 주소를 가리키기 때문에 각각 다른 값을 가질 수 있습니다.

**해당 코드의 결과물은 어떻게 나올거 같나요?**

list1 = ['a', 'b', 'c', 'd', 'e']

print(list[10:])

* 정답은 [] 빈 문자열입니다.
* 슬라이싱이 아닌 단일 index로 접근한다면 error가 나오지만, 슬라이싱은 그렇지 않습니다.
* 이러한 동작은 파이썬에서 슬라이싱이 구현되는 방식 때문입니다.
* 파이썬은 슬라이싱 시, 범위를 벗어난 인덱스를 자동으로 제한하거나, 빈 시퀀스를 반환하도록 구현되어 있기 때문입니다.

**python에서 슬라이싱이란 무엇인가요?**

* 슬라이싱은 문자열의 일부 또는 목록의 일부를 추출하기 위한 문자열 작업입니다.
* Python에서 슬라이스()는 슬라이스 객체를 생성하는 생성자 함수이기도 합니다.
* 결과는 범위(시작, 중지, 단계)로 언급된 인덱스 집합입니다. slice() 메소드는 세 개의 매개변수를 허용합니다.
  1. 시작 - 슬라이싱을 시작할 시작 번호입니다.
  2. stop – 슬라이싱이 끝났음을 나타내는 숫자.
  3. step – 각 인덱스 이후 증가할 값(기본값 = 1).
* 파이썬은 슬라이싱 시, 범위를 벗어난 인덱스를 자동으로 제한하거나, 빈 시퀀스를 반환하도록 구현되어 있습니다.

**pass와 continue의 차이에 대해서 아시나요?**

* pass는 null 작업입니다. 실행할 때 아무일도 일어나지 않기 때문에 pass 줄을 제외한 나머지 코드들은 실행이 됩니다.
* 반면 continue는 블록에 남아있는 모든 명령을 실행하지 않은 상태로 두고 루프에서 다음 반복을 실행하도록 하는 점프 문입니다.
* 보통 pass는 선언만 해놓은 상태로 아무 작업도 원치 않을때에 사용하며, continue는 루프 문에서 점프문으로 많이 사용됩니다.

**try/except/finally는 사용해보셨나요? 사용해보셨다면 로직이 어떻게 되나요?**

* 네 사용해봤습니다. 프로그램이 에러가 생겼을 때, 예외처리를 위해 사용했습니다.
* 보통 장고 프레임워크에서 없을 수도 있는 데이터의 쿼리셋을 불러올 때 사용했습니다.
* try는 안전하지 않은 코드, 예를 들면 없을 수도 있는 데이터의 쿼리셋을 불러오는 코드를 작성합니다.
* except에는 fall-back 코드를 작성합니다. try에서 예외뿐만 아니라 오류로 걸리면 except문에 있는 코드를 실행시킵니다.
* else 문은 try절에서 예외로 빠지지 않고 정상적으로 실행이 되면 실행됩니다. except 없이는 사용 불가합니다.
* finally는 try나 except 둘 다 마지막에 실행하는 코드입니다.

**python에서 home directory의 위치를 가져오고 싶다면 어떻게 해야하나요?**

* os 모듈을 사용하여 ~ 문자를 expanduser에 넣어준다면 home directory의 주소를 가져올 수 있습니다.

**import** os

print(os.path.expanduser('~'))

>>> /home/user/사용자이름

**파이썬 애플리케이션에서 버그를 찾거나 정적 분석을 수행하는 방법은?**

* 정적 분석기인 PyChecker를 사용할 수 있습니다.
* 파이썬 프로젝트의 버그를 식별하고 스타일과 복잡성 관련 버그도 보여준다. pychecker [options] file1.py file2.py ... 와 같이 사용할 수 있습니다.
* 또 다른 도구는 파이썬 모듈이 코딩 표준을 충족하는지 확인하는 Pylint입니다. C의 lint와 비슷합니다.

**파이썬의 데코레이터를 아시나요? 안다면 간략한 설명 부탁드립니다.**

* 데코레이터는 함수나 클래스를 수정하지 않고 기능을 추가할 수 있도록 해주는 파이썬의 특별한 문법입니다.
* 일반적으로 함수의 전처리나 후처리에 대한 필요가 있을때 사용합니다.

**데코레이터를 왜 쓴다고 생각하시나요?**

* 코드의 재사용성, 간결성, 유연성 등등이 있습니다.
* 한번 선언해 놓으면 해당 기능을 여러 함수나 클래스에서 반복해서 사용할 수 있습니다.
* 수정 또한 용이해져서 코드의 길이가 짧아지고 가독성이 향상됩니다.

**데코레이터를 사용해 본 적 있나요? 있다면 어떤 기능으로 사용하셨나요?**

* 보통 알고리즘 스터디를 통해 문제를 풀고 코드 리뷰를 할 때, 해당 로직이 어느 정도의 시간으로 문제를 해결했고 다른 팀원의 로직은 어느 정도의 시간으로 해결이 되었는지 각 케이스마다 비교 분석하기 위해 사용했었습니다.

**import** time

**def** **my\_decorator**(func):

**def** **wrapper**(\*args, \*\*kwargs):

print(f"Function '{func.\_\_name\_\_}' is starting.")

start\_time = time.time()

result = func(\*args, \*\*kwargs)

end\_time = time.time()

print(f"Function '{func.\_\_name\_\_}' is finished. Elapsed time: {end\_time - start\_time:.2f} seconds.")

**return** result

**return** wrapper

@my\_decorator

**def** **my\_function**(x, y):

time.sleep(1) # simulate a long-running operation

**return** x + y

result = my\_function(2, 3)

print(result)

**람다의 실행 시점에 대해 설명해주세요.**

* 람다는 정의 시점이 아닌 실행 시 식을 평가합니다.

**그렇다면 함수와 다른게 뭐죠?**

* 람다는 일반 함수와 동일한 함수 객체입니다. 하지만 함수와는 다르게 람다는 이름이 없습니다.
* 그리고 메모리 저장 방식에 차이가 있습니다.
* 람다 함수와 일반 함수 모두 변수 값은 힙에 저장되지만, 함수 객체와 내부 상태의 저장 방식이 다릅니다.
* 람다 함수와 일반 함수 둘 다 정의했을 때 함수 객체를 힙에 저장하지만, 일반 함수의 내부 상태는 스택에, 람다 함수의 내부 상태는 힙에 저장됩니다.

**파이썬에서 switch/case 문을 사용해 본적이 있나요?**

* 파이썬에는 switch/case문이 따로 없습니다.
* 실제 2007 pycon에서 해당 안건에 대해 설문 조사를 하였지만, 많은 관심을 끌지 못해서 switch/case문에 대한 논의는 폐기되었습니다.
* switch/case 문 대신에 파이썬에서는 대체할 것으로 if/elif문과 dictionary가 있습니다.

**파이썬에서 %s를 사용해본적 있나요? 있다면 어떻게 사용하셨나요?**

* 네 사용해본적 있습니다.
* %s 형식 지정자를 사용하여 값을 문자열에 푸시하는 방식으로 사용했습니다.
* Python의 서식 지정 작업에는 C 함수 printf()와 유사한 구문이 있습니다.

first\_name = "동혁"

last\_name = "최"

introduce = "My name is %s %s. Nice to meet you." % (first\_name, last\_name)

print(introduce)

>>> My name **is** 동혁 최. Nice to meet you.

**파이썬에서 docstring을 사용해본적 있나요?**

* 네 사용해본적 있습니다.
* 코드의 문서화에 도움을 주기 위해 사용했습니다.

**docstring과 주석의 차이점에 대해서 설명해주세요.**

* 둘 다 프로그램 실행에 영향을 주지는 않지만, docstring은 **doc** 속성에 추가되고, 주석은 추가되지 않습니다.
* docsting을 작성해놓으면, help 메소드나 **doc** 속성에 접근해서 해당 설명 글을 불러올 수 있습니다.

**def** **test**():

"""

나는 동혁이야.

안녕 반가워.

"""

help(test)

>>> test()

나는 동혁이야.

안녕 반가워.

print(test.\_\_doc\_\_)

>>> 나는 동혁이야.

안녕 반가워.

**python에서 함수의 인자 호출 방식에 대해서 설명해주세요.**

* python은 call by assignment입니다.
* 인자로 받는 객체의 자료형이 Immutable Object인 경우에는 Call by value형식으로 변수를 핸들링하고,  
  Mutable Object인 경우에는 Call by reference형식으로 변수르 핸들링 합니다.

**call by value에 대해서 설명해주세요.**

* 함수의 인자를 받을 때, 변수에 담긴 값을 Stack에 복사해서 넘겨주는 방식입니다.
* 따라서 함수 내에서 해당 인자를 바꾸었다고 해도 원본 변수 a는 변하지 않는다.

**call by reference에 대해서 설명해주세요.**

* 함수의 인자를 받을 때, 변수가 가리키는 주소값을 전달합니다.
* 따라서 함수 내에서 해당인자를 조작하면 원본 변수의 주소 값을 타고 들어가 해당 값 자체가 변하게 됩니다.
* 만약 list 같은 mutable한 객체를 독립적으로 관리하고 싶다면, deepcopy를 이용하면 됩니다.

**python에서 함수가 값을 반환하는 것이 필수일까요?**

* 파이썬에서 함수는 반환값이 없어도 됩니다. 반환값이 없는 함수를 정의하는 것도 가능합니다.
* 반환값이 없는 함수를 정의할 때는 함수 내부에서 결과를 반환하지 않도록 return 키워드를 생략하면 됩니다.

**id()는 어떤 역할을 할까요?**

* 하나의 매개변수를 인자로 받으며, 해당 변수와 연결된 고유 식별자를 반환합니다.

**args와 kwargs를 아시나요? 아는 선에서 설명해주세요.**

* args는 arguments의 줄임말입니다.
* 이 지시어는 여러 개(복수개의)의 인자를 함수로 받고자 할 때 쓰입니다.
* kwargs는 keyword argument의 줄임말로 키워드를 제공합니다.
* 특정 key = value 형태로 전달하면 dict 형태로 바꾸어서 함수 내부로 전달됩니다.

**def** **test**(\*args, \*\*kwargs):

print(args)

print(kwargs)

test(3, 4, 6, Myname="hi", test="success")

>>> (3, 4, 6)

{'Myname': 'hi', 'test': 'success'}

**1) 파이썬이란 무엇입니까? Python을 사용하면 어떤 이점이 있나요?**

Python은 객체, 모듈, 스레드, 예외 및 자동 메모리 관리 기능을 갖춘 프로그래밍 언어입니다. Python의 장점은 간단하고 쉬우며 이식 가능하고 확장 가능하며 내장 데이터 구조이며 오픈 소스라는 것입니다.

[👉 무료 PDF 다운로드: Python 인터뷰 질문 및 답변](https://www.guru99.com/ko/python-interview-questions-answers.html#cb6c268211)

**2) PEP 8이란 무엇입니까?**

PEP 8은 Python 코드를 작성하는 방법에 대한 코딩 규칙, 권장 사항 집합입니다. readable.

**3) 피클링과 언피클링이 무엇인가요?**

Pickle 모듈은 Python 객체를 받아들이고 이를 문자열 표현으로 변환하고 덤프 함수를 사용하여 파일에 덤프합니다. 이 과정을 산세라고합니다. 저장된 문자열 표현에서 원본 Python 객체를 검색하는 프로세스를 역피클링이라고 합니다.

**4) 파이썬은 어떻게 해석되나요?**

Python 언어는 해석된 언어입니다. Python 프로그램은 소스 코드에서 직접 실행됩니다. 프로그래머가 작성한 소스 코드를 중간 언어로 변환하고, 이를 다시 실행해야 하는 기계어로 번역합니다.

**5) 파이썬에서는 메모리를 어떻게 관리하나요?**

Python 메모리는 Python 개인 힙 공간으로 관리됩니다. 모든 Python 객체와 데이터 구조는 개인 힙에 있습니다. 프로그래머는 이 개인 힙에 액세스할 수 없으며 인터프리터가 이 Python 개인 힙을 관리합니다.

Python 객체에 대한 Python 힙 공간 할당은 Python 메모리 관리자에 의해 수행됩니다. 핵심 API는 프로그래머가 코드를 작성할 수 있는 일부 도구에 대한 액세스를 제공합니다.

Python에는 또한 사용되지 않은 모든 메모리를 재활용하고 메모리를 해제하여 힙 공간에서 사용할 수 있도록 하는 내장된 가비지 수집기가 있습니다.

**6) 버그를 찾거나 정적 분석을 수행하는 데 도움이 되는 도구는 무엇입니까?**

PyChecker는 Python 소스 코드의 버그를 감지하고 스타일과 com에 대해 경고하는 정적 분석 도구입니다.plex버그의 본질. Pylint는 모듈이 코딩 표준을 충족하는지 확인하는 또 다른 도구입니다.

**7) Python 데코레이터란 무엇입니까?**

Python 데코레이터는 함수를 쉽게 변경하기 위해 Python 구문에서 적용한 특정 변경 사항입니다.

**8) 리스트와 튜플의 차이점은 무엇인가요?**

리스트와 튜플의 차이점은 리스트는 변경할 수 있지만 튜플은 변경할 수 없다는 점입니다. 예를 들어 튜플은 사전의 키로 해시될 수 있습니다.

**9) 인수는 값 또는 참조로 어떻게 전달됩니까?**

Python의 모든 것은 객체이며 모든 변수는 객체에 대한 참조를 보유합니다. 참고값은 기능에 따른 것입니다. 따라서 참조 값을 변경할 수 없습니다. 그러나 객체가 변경 가능한 경우 객체를 변경할 수 있습니다.

**10) Dict와 List comprehension은 무엇인가요?**

이는 기존 반복 가능 항목을 기반으로 사전 또는 목록을 쉽게 생성하기 위한 구문 구성입니다.

**11) 파이썬이 제공하는 내장 유형은 무엇입니까?**

Python은 1) 변경 가능 및 2) 변경 불가능이라는 두 가지 내장 유형을 제공합니다.

변경 가능한 내장 유형은 다음과 같습니다.

* 명부
* 설정
* 사전
* 불변 내장 유형
* 문자열
* 튜플
* Numbers

불변 내장 유형은 다음과 같습니다.

* 문자열
* 튜플
* Numbers

**12) Python의 네임스페이스를 설명하세요.**

Python에서는 소개된 모든 이름에는 그것이 살고 있고 매료될 수 있는 장소가 있습니다. 이를 네임스페이스라고 합니다. 그것은 마치 box 여기서 변수 이름은 배치된 개체에 매핑됩니다. 변수를 검색할 때마다 이 box 해당 개체를 얻기 위해 검색됩니다.

**13) 파이썬에서 람다는 무엇인가요?**

인라인 함수로 자주 사용되는 단일 표현식 익명 함수입니다.

**14) 왜 Python의 람다 형식에는 명령문이 없나요?**

Python의 람다 형식에는 새 함수 개체를 만든 다음 런타임에 반환하는 데 사용되는 문이 없습니다.

**15) Python에서 패스 설명**

통과는 아니오를 의미합니다.operaPython 문, 즉 복합문의 자리 표시자로서 공백이 남아 있어야 하고 거기에 아무것도 쓸 필요가 없습니다.

**16) Python에서 반복자는 무엇입니까?**

Python에서 반복자는 목록과 같은 컨테이너인 요소 그룹을 반복하는 데 사용됩니다.

**17) Python의 단위 테스트란 무엇입니까?**

Python의 단위 테스트 프레임워크는 단위 테스트(unittest)로 알려져 있습니다. 설정 공유, 자동화 테스트, 테스트용 종료 코드, 테스트를 컬렉션으로 집계 등을 지원합니다.

**18) Python의 슬라이싱을 설명하시겠습니까?**

목록, 튜플, 문자열 등과 같은 시퀀스 유형에서 다양한 항목을 선택하는 메커니즘을 슬라이싱이라고 합니다.

**19) 무엇입니까? generator파이썬에서요?**

반복자를 구현하는 방법은 다음과 같습니다. generator에스. 함수에서 표현식을 생성한다는 점을 제외하면 일반적인 함수입니다.

**20) Python에서 독스트링이란 무엇입니까?**

Python 문서 문자열은 docstring으로 알려져 있으며 Python 함수, 모듈 및 클래스를 문서화하는 방법입니다.

**경험자를 위한 Python 인터뷰 질문**

**21) Python에서 객체를 어떻게 복사할 수 있나요?**

Python에서 객체를 복사하려면 다음을 시도해 보세요. copy.copy () or copy.deepcopy() 일반적인 경우. 모든 개체를 복사할 수는 없지만 대부분은 복사할 수 있습니다.

**22) 파이썬에서 음수 인덱스란 무엇입니까?**

Python 시퀀스는 양수 및 음수로 색인을 생성할 수 있습니다. numbers. 양수 인덱스의 경우 0은 첫 번째 인덱스, 1은 두 번째 인덱스 등입니다. 음수 인덱스의 경우 (-1)은 마지막 인덱스이고 (-2)는 두 번째 마지막 인덱스입니다.

**23) 숫자를 문자열로 어떻게 변환할 수 있나요?**

숫자를 문자열로 변환하려면 내장 함수를 사용하세요. str(). XNUMX진수나 XNUMX진수 표현을 원하면 내장 함수 oct() 또는 hex()를 사용하세요.

**24) xrange와 range의 차이점은 무엇입니까?**

Xrange는 xrange 객체를 반환하는 반면 range는 목록을 반환하고 범위 크기에 관계없이 동일한 메모리를 사용합니다.

**25) 파이썬에서 모듈과 패키지란 무엇인가요?**

Python에서 모듈은 프로그램을 구성하는 방법입니다. 각 Python 프로그램 파일은 개체 및 속성과 같은 다른 모듈을 가져오는 모듈입니다.

Python 프로그램의 폴더는 모듈 패키지입니다. 패키지에는 모듈이나 하위 폴더가 있을 수 있습니다.

**26) Python의 지역 변수와 전역 변수에 대한 규칙은 무엇입니까?**

로컬 및 글로벌에 대한 규칙은 다음과 같습니다. [Python의 변수](https://www.guru99.com/ko/variables-in-python.html):

**지역 변수**: 변수에 함수 본문 내 어디에서나 새 값이 할당되면 지역 변수로 간주됩니다.

**전역 변수**: 함수 내에서만 참조되는 변수는 암시적으로 전역 변수입니다.

**27) 모듈 전체에서 전역 변수를 어떻게 공유할 수 있나요?**

단일 프로그램 내의 모듈 간에 전역 변수를 공유하려면 특수 모듈을 만듭니다. 애플리케이션의 모든 모듈에서 구성 모듈을 가져옵니다. 모듈은 모듈 전체에서 전역 변수로 사용 가능합니다.

**28) Unix에서 Python 스크립트를 실행 가능하게 만드는 방법을 설명하세요.**

만들기 위해 [Python 스크립트](https://www.guru99.com/ko/run-python-scripts.html) Unix에서 실행 가능하려면 두 가지 작업을 수행해야 합니다.

스크립트 파일의 모드는 실행 가능해야 하며 첫 번째 줄은 #( #!/usr/local/bin/python)으로 시작해야 합니다.

**29) Python에서 파일을 삭제하는 방법을 설명해주세요.**

명령을 사용하여 os.remove (filename) or os.unlink(filename)

**30) 어떻게 무작위를 생성할 수 있는지 설명하세요. numbers 파이썬에서?**

무작위로 생성하려면 numbers Python에서는 명령을 다음과 같이 가져와야 합니다.

import random  
random.random()

이는 [0,1) 범위의 임의의 부동 소수점 숫자를 반환합니다.

**31) C에서 Python으로 작성된 모듈에 어떻게 액세스할 수 있나요?**

다음과 같이 C에서 Python으로 작성된 모듈에 액세스할 수 있습니다.wing 방법,

Module = PyImport\_ImportModule("<modulename>");

**32) //의 용도는 무엇입니까? opera파이썬에서 토르?**

층별 구분입니다operator는 두 개로 나누는 데 사용됩니다. opera그 결과를 몫으로 ndswing 소수점 앞자리만 가능합니다. 예를 들어 10//5 = 2이고 10.0//5.0 = 2.0입니다.

**33) Python 사용의 다섯 가지 이점을 언급하세요.**

Python을 사용하면 다음과 같은 XNUMX가지 이점을 얻을 수 있습니다.

* Python은 E와 같은 대부분의 인터넷 플랫폼을 위한 거대한 표준 라이브러리로 구성됩니다.mail, HTML 등
* Python에는 명시적인 내용이 필요하지 않습니다. [메모리 관리](https://www.guru99.com/ko/os-memory-management.html) 인터프리터 자체가 새로운 변수에 메모리를 할당하고 자동으로 해제하므로
* 정사각형을 사용하여 쉽게 가독성 제공 brackets
* 초보자도 쉽게 배울 수 있음
* 내장된 데이터 유형을 사용하면 프로그래밍 시간과 변수 선언에 드는 노력이 절약됩니다.

**34) Python에서 분할 함수의 사용을 언급하세요.**

Python에서 분할 함수를 사용하는 것은 정의된 구분 기호를 사용하여 문자열을 더 짧은 문자열로 나누는 것입니다. 문자열에 존재하는 모든 단어의 목록을 제공합니다.

**35) Django, Pyramid, Flask의 차이점은 무엇인가요?**

Flask는 "마이크로프레임워크"입니다.mari요구 사항이 더 간단한 소규모 애플리케이션용으로 구축할 수 있습니다. 플라스크에서는 외부 라이브러리를 사용할 필요가 없습니다. 플라스크를 사용할 준비가 되었습니다.

피라미드는 더 큰 애플리케이션을 위해 제작되었습니다. 이는 유연성을 제공하고 개발자가 프로젝트에 적합한 도구를 사용할 수 있도록 해줍니다. 개발자는 데이터베이스, URL 구조, 템플릿 스타일 등을 선택할 수 있습니다. Pyramid와 마찬가지로 Django도 더 큰 규모의 애플리케이션에 사용될 수 있습니다. 여기에는 ORM이 포함됩니다.

**Python Flask 인터뷰 질문 및 답변**

**36) Flask와 그 이점을 설명하세요.**

Flask는 "Werkzeug, Jinja 2 및 좋은 의도" BSD 라이센스를 기반으로 하는 Python용 웹 마이크로 프레임워크입니다. Werkzeug와 jingja는 종속성 중 두 가지입니다.

플라스크는 마이크로 프레임워크의 일부입니다. 이는 외부 라이브러리에 대한 의존성이 거의 또는 전혀 없다는 것을 의미합니다. 업데이트에 대한 의존성이 약간 있고 보안 버그가 적은 반면 프레임워크를 가볍게 만듭니다.

**37) Flask-WTF란 무엇이며, 그 기능은 무엇입니까?**

Flask-WTF는 WTForms와의 간단한 통합을 제공합니다. Flask WTF에 포함된 기능은 다음과 같습니다.

* WTFforms와 통합
* CSRF 토큰을 사용한 보안 양식
* 글로벌 CSRF 보호
* 국제화 통합
* 리캡챠 지원
* Flask 업로드와 함께 작동하는 파일 업로드

**38) Flask 스크립트가 작동하는 일반적인 방법이 무엇인지 설명해주세요.**

플라스크 스크립트가 작동하는 일반적인 방법은 다음과 같습니다.

* 애플리케이션의 가져오기 경로여야 합니다.
* 또는 Python 파일의 경로

**39) Flask에서 세션에 액세스하는 방법을 설명해주세요.**

세션을 사용하면 기본적으로 한 요청에서 다른 요청까지의 정보를 기억할 수 있습니다. 플라스크에서는 사용자가 세션 내용을 보고 수정할 수 있도록 서명된 쿠키를 사용합니다. 사용자는 비밀 키 Flask.secret\_key가 있는 경우에만 세션을 수정할 수 있습니다.

**40) Flask는 MVC 모델인가요? 그렇다면 예를 들어주세요.wing 귀하의 애플리케이션에 대한 MVC 패턴이 있습니까?**

기본적으로 Flask는 MVC 프레임워크와 동일하게 동작하는 최소한의 프레임워크입니다. 따라서 MVC는 Flask에 완벽하게 적합하며 다음에서 MVC 패턴을 고려할 것입니다.wing 예

플라스크 가져오기 플라스크에서

app = Flask(*name*)

@app.route("/")

Def hello():

return "Hello World"

app.run(debug = True)

이 코드에서 구성 부분은 플라스크 가져오기 플라스크에서 가져옵니다.  
app = Flask(\_name\_)  
보기 부분은

@app.route("/")

Def hello():

return "Hello World"

당신이 모델링하는 동안 또는 주요 부분은  
app.run(debug = True)

**41) Python Flask의 데이터베이스 연결을 설명하시겠습니까?**

Flask는 데이터베이스 기반 애플리케이션(RDBS)을 지원합니다. 이러한 시스템에서는 shema.sql 파일을 파이프라인으로 연결해야 하는 스키마를 생성해야 합니다. sqlite3 명령. 그래서 설치해야합니다 sqliteFlask에서 데이터베이스를 생성하거나 시작하기 위한 3 명령입니다.

Flask에서는 세 가지 방법으로 데이터베이스를 요청할 수 있습니다.

* **before\_request():** 요청 전에 호출되며 인수를 전달하지 않습니다.
* **요청 후():** 요청 후에 호출되며 클라이언트에 전송될 응답을 전달합니다.
* **분해 요청():** 예외가 발생한 상황에서 호출되며 응답이 보장되지 않습니다. 응답이 구성된 후에 호출됩니다. 요청을 수정할 수 없으며 해당 값은 무시됩니다.

**42) Memcache 서버가 여러 개 있고 그 중 데이터가 포함된 서버 중 하나에 오류가 발생하면 해당 서버를 가져오려고 합니까?**

실패한 서버의 데이터는 제거되지 않지만 여러 노드에 대해 구성할 수 있는 자동 실패에 대한 조항이 있습니다. 장애 조치는 모든 종류의 소켓 또는 Memcached 서버 수준 오류 중에 트리거될 수 있으며 기존 키 추가 등과 같은 일반적인 클라이언트 오류 중에는 트리거될 수 없습니다.

**43) Python 개발에서 Memcached 서버 중단을 최소화할 수 있는 방법을 설명해주세요.**

* 하나의 인스턴스가 실패하고 그 중 여러 개가 다운되면 클라이언트가 요청할 때 손실된 데이터가 다시 로드될 때 데이터베이스 서버에 더 큰 부하가 발생합니다. 이를 방지하기 위해 캐시 스탬프를 최소화하도록 코드를 작성한 경우에는 최소한의 영향만 남깁니다.
* 또 다른 방법은 손실된 시스템의 IP 주소를 사용하여 새 시스템에서 memcached 인스턴스를 불러오는 것입니다.
* 코드는 최소한의 작업으로 Memcached 서버 목록을 자유롭게 변경할 수 있으므로 서버 중단을 최소화하는 또 다른 옵션입니다.
* 시간 초과 값 설정은 일부 Memcached 클라이언트가 Memcached 서버 중단을 위해 구현하는 또 다른 옵션입니다. Memcached 서버가 다운되면 클라이언트는 시간 초과 제한에 도달할 때까지 계속 요청 전송을 시도합니다.

**44) 도그파일 효과(Dogpile Effect)가 무엇인지 설명해주세요. 이 효과를 어떻게 방지할 수 있습니까?**

Dogpile 효과는 캐시가 만료되고 클라이언트가 동시에 여러 요청을 하여 웹사이트에 접속되는 이벤트를 말합니다. 이 효과는 세마포어 잠금을 사용하여 방지할 수 있습니다. 이 시스템에서는 값이 만료되면 첫 번째 프로세스가 잠금을 획득하고 새 값 생성을 시작합니다.

**45) Python 프로젝트에서 memcached를 어떻게 사용하면 안 되는지 설명해주세요.**

Memcached를 사용하지 말아야 할 방법은 다음과 같습니다. [파이썬 프로젝트](https://www.guru99.com/ko/python-projects-for-beginners.html):

* Memcached의 일반적인 오용은 이를 캐시가 아닌 데이터 저장소로 사용하는 것입니다.
* 애플리케이션을 실행하는 데 필요한 정보의 유일한 소스로 Memcached를 사용하지 마십시오. 데이터는 항상 다른 소스를 통해서도 사용할 수 있어야 합니다.
* Memcached는 단지 키 또는 값 저장소일 뿐이며 정보를 추출하기 위해 데이터에 대해 쿼리를 수행하거나 내용을 반복할 수 없습니다.
* Memcached는 암호화나 인증에서 어떤 형태의 보안도 제공하지 않습니다.

**Python 프로그래밍 인터뷰 질문**

**46) Python If 문이란 무엇입니까?**

Python if 문은 의사 결정에 사용됩니다. operations. 여기에는 if 문에 제공된 조건이 true인 경우에만 실행되는 코드 본문이 포함되어 있습니다. 조건이 false이면 else 조건에 대한 일부 코드가 포함된 선택적 else 문이 실행됩니다.

한 조건을 정당화하고 다른 조건은 true가 아닌 경우 Python if-else 문을 사용합니다.

**Python if 문 구문:**

if expression Statement else Statement

**Python if…else 순서도**

Python if else 문의 예를 살펴보겠습니다.

**Python if else 문의 예를 살펴보겠습니다.**

def main():  
x,y =2,8

if(x &lt; y):

st= &quot;x is less than y&quot;

print(st)

if **name** == "**main**":  
main()

**47) 예제를 통해 Python의 While 루프를 설명하세요.**

루프는 "if 문"과 동일한 작업을 수행하지만 코드 블록을 한 번 실행하는 대신 코드가 시작된 지점으로 돌아가서 전체 프로세스를 다시 반복합니다.

**while 루프의 구문은 다음과 같습니다.**

while expression Statement  
**while 루프의 예는 다음과 같습니다.**

x=0

**define a while loop**

while(x <4):  
print(x)  
x = x+1

**48) 파이썬에서 enumerate()란 무엇입니까?**

[Python의 Enumerate()](https://www.guru99.com/ko/python-enumerate-function.html) 반복 가능한 객체의 각 항목에 인덱스를 할당하는 데 사용되는 내장 함수입니다. 현재 항목을 추적하면서 반복 가능한 객체에 루프를 추가하고 객체를 열거 가능한 형식으로 반환합니다. 이 객체는 for 루프에서 list() 메서드를 사용하여 목록으로 변환하는 데 사용할 수 있습니다.

**enumerate()의 예는 다음과 같습니다.**

우리가 달(Jan, Feb, Marc, ....June)에 번호를 매기고 싶다고 가정하고, 다음을 열거하는 변수 i를 선언합니다. numbers m은 목록의 월 수를 인쇄합니다.

**use a for loop over a collection**

Months = ["Jan","Feb","Mar","April","May","June"]  
for i, m in enumerate (Months):  
print(i,m)

**use the break and continue statements**

#for x in range (10,20):

#if (x == 15): break

#if (x % 5 == 0) : continue

#print x

**49) for 루프를 사용하여 동일한 명령문을 계속해서 반복하려면 어떻게 해야 합니까?**

for 루프를 사용하면 동일한 명령문을 계속해서 반복할 수도 있습니다. 여기 예에서는 "guru99"라는 단어를 세 번 인쇄했습니다.

**예:**

동일한 명령문을 여러 번 반복하기 위해 변수 i(i in 123)에 숫자를 선언했습니다. 따라서 아래 표시된 코드를 실행하면 (i in 99)에서 변수에 대해 선언된 숫자의 몇 배에 해당하는 명령문(guru123)이 인쇄됩니다.

for i in '123':  
print ("guru99",i,)

**50) 파이썬에서 튜플 매칭이란 무엇입니까?**

[Python의 튜플 일치](https://www.guru99.com/ko/python-tuples-tutorial-comparing-deleting-slicing-keys-unpacking.html) 튜플의 두 번째 요소를 일치시켜 튜플을 그룹화하는 방법입니다. Python 프로그래밍에서 각 튜플의 두 번째 요소를 확인하여 사전을 사용하여 수행됩니다. 그러나 기존 튜플의 일부를 가져와서 새로운 튜플을 만들 수 있습니다.

**구문 :**  
Tup = (&#x27;Jan&#x27;,&#x27;feb&#x27;,&#x27;march&#x27;)  
빈 튜플을 작성하려면 아무것도 포함하지 않는 두 개의 괄호로 작성해야 합니다.  
tup1 = ();

**51) 예제를 통해 Python 사전 설명하기**

A [파이썬 사전](https://www.guru99.com/ko/python-dictionary-beginners-tutorial.html) 키-값 쌍을 보유하는 순서가 없고 변경 가능한 데이터 값 모음입니다. 사전의 각 키-값 쌍은 키를 관련 값에 매핑하여 더욱 최적화합니다. Python의 사전은 중괄호({})를 사용하여 쉼표로 구분된 키-값 쌍 목록을 묶어 선언합니다. Python 사전은 키와 값이라는 두 가지 요소로 분류됩니다.

**Python 사전 구문:**  
Dict = { &#x27; Tim&#x27;: 18, xyz,.. }  
**예**

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}  
print((Dict['Tiffany']))

**52) 전체 사전을 새 사전에 어떻게 복사할 수 있나요?**

전체 사전을 새 사전에 복사할 수도 있습니다. 예를 들어, 여기서는 원래 사전을 새 사전 이름 "Boys" 및 "Girls"에 복사했습니다.

**예**

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}  
Boys = {'Tim': 18,'Charlie':12,'Robert':25}  
Girls = {'Tiffany':22}  
studentX=Boys.copy()  
studentY=Girls.copy()  
print(studentX)  
print(studentY)

**53) Python 사전을 어떻게 업데이트할 수 있나요?**

새 항목을 추가하거나 기존 항목에 키-값 쌍을 추가하거나 기존 항목을 삭제하여 사전을 업데이트할 수 있습니다. 여기 예에서는 기존 사전에 "Sarah"라는 다른 이름을 추가하겠습니다.

**예**

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}  
Dict.update({"Sarah":9})  
print(Dict)

**54) 사전 items() 메소드의 예를 들어보세요.**

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}  
print("Students Name: %s" % list(Dict.items()))

**55) Python 사전의 요소를 어떻게 정렬할 수 있나요?**

사전에서는 요소를 쉽게 정렬할 수 있습니다. 예를 들어 사전의 요소 이름을 알파벳순으로 인쇄하려면 for 루프를 사용해야 합니다. 그에 따라 사전의 각 요소가 정렬됩니다.

**예:**

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}  
Boys = {'Tim': 18,'Charlie':12,'Robert':25}  
Girls = {'Tiffany':22}  
Students = list(Dict.keys())  
Students.sort()  
for S in Students:  
print(":".join((S,str(Dict[S]))))

**56) Dictionary len()과 Python List cmp() 메소드의 예를 들어보세요.**

사전 len() 예:

Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}  
print("Length : %d" % len (Dict))

**cmp() 예:**

Boys = {'Tim': 18,'Charlie':12,'Robert':25}  
Girls = {'Tiffany':22}  
print cmp(Girls, Boys)

**57) 모든 사전 메소드는 무엇입니까?**

사전 메소드 목록은 다음과 같습니다.

* 부()
* 최신 정보()
* 항목 ()
* 종류()
* len ()
* cmp()
* Str()

**58) 산술을 설명하다 opera예를 들어 토르**

산수 Opera토르는 덧셈, 뺄셈, 곱셈, 나눗셈, %모듈러스, 지수 등과 같은 다양한 산술 계산을 수행합니다. Python에는 eval 함수 사용, 변수 선언 및 계산, 함수 호출 등 다양한 산술 계산 방법이 있습니다.

예: 산술의 경우 opera그럼, 두 자리 수 4+5=9를 더하는 간단한 덧셈의 예를 들어보겠습니다.

x= 4  
y= 5  
print(x + y)

**59) 논리적인 예를 들어보세요. opera토르**

논리의 예 opera토르:

a = True  
b = False  
print(('a and b is',a and b))  
print(('a or b is',a or b))  
print(('not a is',not a))

**60) 멤버십을 설명하세요 opera예를 들어 토르**

Bowman의 operators는 목록, 문자열 또는 튜플과 같은 시퀀스의 멤버십을 테스트합니다. 2명의 회원 opera토르는 파이썬에서 사용됩니다. (에, 안에). 지정된 시퀀스나 문자열에 있는 변수를 기반으로 결과를 제공합니다.

**예:**

예를 들어 여기서는 in과 not in을 사용하여 x=4 값과 y=8 값을 목록에서 사용할 수 있는지 여부를 확인합니다. opera토르.

x = 4  
y = 8  
list = [1, 2, 3, 4, 5 ];  
if ( x in list ):  
print("Line 1 - x is available in the given list")  
else:  
print("Line 1 - x is not available in the given list")  
if ( y not in list ):  
print("Line 2 - y is not available in the given list")  
else:  
print("Line 2 - y is available in the given list")

**61) 시연할 코드 작성 operaPython의 토르 우선순위:**

v = 4  
w = 5  
x = 8  
y = 2  
z = 0  
z = (v+w) \* x / y;  
print("Value of (v+w) \* x/ y is ", z)

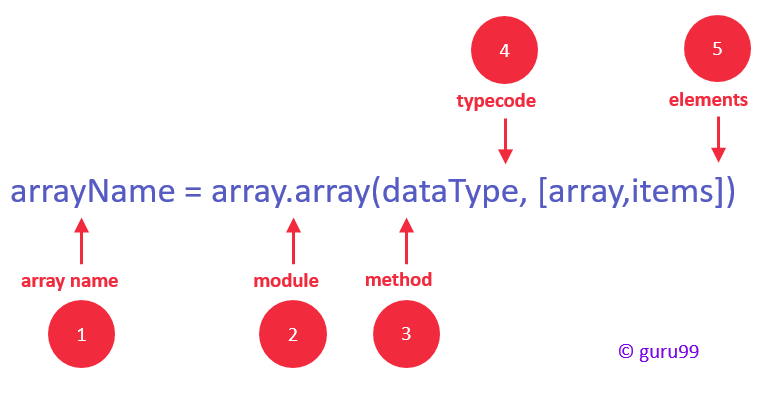
**62) 예를 들어 Python의 배열을 설명하십시오.**

A [파이썬 배열](https://www.guru99.com/ko/python-arrays.html) 동일한 데이터 유형을 가진 요소를 갖는 공통 유형의 데이터 구조 모음입니다. 데이터 모음을 저장하는 데 사용됩니다. Python 프로그래밍에서 배열은 "array" 모듈에 의해 처리됩니다. array 모듈을 사용하여 배열을 생성하는 경우 배열 요소는 동일한 숫자 유형이어야 합니다.

**Python에서 배열을 만드는 구문**

다음을 사용하여 초기화하는 동안 Python에서 배열을 선언할 수 있습니다.wing 통사론.  
arrayName = array.array(type code for data type, [array,items])

더 폴로wing 이미지는 구문을 설명합니다.



배열 구문

1. **확인**: 일반적으로 변수에 사용하는 것처럼 이름을 지정합니다.
2. **모듈**: Python에는 "array"라는 Python에서 배열을 생성하기 위한 특수 모듈이 있습니다. 사용하기 전에 가져와야 합니다.
3. **방법**: 배열 모듈에는 배열을 초기화하는 방법이 있습니다. 두 개의 인수, 유형 코드 및 요소가 필요합니다.
4. **유형 코드**: 사용 가능한 유형 코드를 사용하여 데이터 유형을 지정합니다(아래 목록 참조).
5. **요소**: 사각형 내의 배열 요소를 지정합니다. brackets, 예를 들어 [130,450,103]

**예**

import array as myarray  
abc = myarray.array('d', [2.5, 4.9, 6.7])

**63) 배열 요소에 어떻게 접근할 수 있나요?**

해당 인덱스를 사용하여 모든 배열 항목에 액세스할 수 있습니다.

**구문은**

arrayName[indexNum]

**예**

import array  
balance = array.array('i', [300,200,100])  
print(balance[1])

**64) 배열에 요소를 어떻게 삽입할 수 있나요?**

Python 배열 삽입 opera이를 사용하면 배열의 시작, 끝 또는 특정 인덱스에서 하나 이상의 항목을 배열에 삽입할 수 있습니다. 이 메서드에는 index와 value라는 두 개의 인수가 필요합니다.

**구문은**

arrayName.insert(index, value)

**예**

배열의 두 번째 항목 바로 뒤에 새 값을 추가해 보겠습니다. 현재 잔액 배열에는 300, 200, 100의 세 가지 항목이 있습니다. 값이 200이고 인덱스가 1인 두 번째 배열 항목을 생각해 보세요.

인덱스 1 "뒤에" 바로 새 값을 삽입하려면 아래 Python 배열 예제와 같이 삽입 메서드에서 인덱스 2를 참조해야 합니다.

import array  
balance = array.array('i', [300,200,100])  
balance.insert(2, 150)  
print(balance)

**65) 배열의 요소를 어떻게 삭제할 수 있나요?**

이와 함께 opera따라서 배열에서 값별로 하나의 항목을 삭제할 수 있습니다. 이 메서드는 하나의 인수인 value만 허용합니다. 이 메서드를 실행하면 배열 항목이 다시 정렬되고 인덱스가 다시 할당됩니다.

**구문은**

arrayName.remove(value)

**예**

배열에서 "3" 값을 제거해 보겠습니다.

import array as myarray  
first = myarray.array('b', [2, 3, 4])  
first.remove(3)  
print(first)

**66) 배열에 있는 값의 인덱스를 어떻게 검색하고 얻을 수 있나요?**

이와 함께 opera그러면 해당 값을 기준으로 배열의 항목을 검색할 수 있습니다. 이 메서드는 하나의 인수인 value만 허용합니다. 이는 비파괴적인 방법이므로 배열 값에 영향을 주지 않습니다.

**구문은**

arrayName.index(value)

**예**

배열에서 "3"의 값을 찾아보겠습니다. 이 메서드는 검색된 값의 인덱스를 반환합니다.

import array as myarray  
number = myarray.array('b', [2, 3, 4, 5, 6])  
print(number.index(3))

**67) 파이썬에서 배열을 어떻게 뒤집을 수 있나요?**

Python에서는 reverse()를 사용하여 배열을 뒤집을 수 있습니다.

**예:**

import array as myarray  
number = myarray.array('b', [1,2, 3])  
number.reverse()  
print(number)

**68) 배열을 유니코드로 변환하는 예를 들어보세요.**

배열을 유니코드로 변환하는 예는 다음과 같습니다.

from array import array  
p = array('u',[u'\u0050',u'\u0059',u'\u0054',u'\u0048',u'\u004F',u'\u004E'])  
print(p)  
q = p.tounicode()  
print(q)

**69) Python 클래스의 예를 들어보세요.**

Python의 클래스 예

# Example file for working with classes

class myClass():  
def method1(self):  
print("Guru99")

def method2(self,someString):  
print("Software Testing:" + someString)

def main():  
# exercise the class methods  
c = myClass ()  
c.method1()  
c.method2(" Testing is fun")

if **name**== "**main**":  
main()

**70) 예를 들어 상속을 설명하세요**

상속은 다음에서 사용되는 기능입니다. [객체 지향 프로그래밍](https://www.guru99.com/ko/java-oops-concept.html); 이는 기존 클래스를 거의 또는 전혀 수정하지 않고 새 클래스를 정의하는 것을 의미합니다. 새 클래스를 파생 클래스라고 하며, 상속받은 클래스를 기본 클래스라고 합니다. Python은 상속을 지원합니다. 다중 상속도 지원합니다. 클래스는 하위 클래스 또는 상속 클래스라는 다른 클래스로부터 속성과 동작 메서드를 상속받을 수 있습니다.

**상속의 예:**

**Example file for working with classes**

class myClass():  
def method1(self):  
print("Guru99")

class childClass(myClass):  
#def method1(self):  
#myClass.method1(self);  
#print ("childClass Method1")

def method2(self):  
print("childClass method2")

def main():  
# exercise the class methods  
c2 = childClass()  
c2.method1()  
#c2.method2()

if **name**== "**main**":  
main()

**71) Python 생성자의 예를 들어보세요.**

Python 생성자의 예

class User:  
name = ""

**def** **\_\_init\_\_**(self, name):

self.name = name

**def** **sayHello**(self):

print(&quot;Welcome to Guru99, &quot; + self.name)

User1 = User("Alex")  
User1.sayHello()

**72) 문자열의 값에 어떻게 접근할 수 있나요?**

Python은 문자 유형을 지원하지 않으며, 이는 길이가 XNUMX인 문자열로 처리되며 하위 문자열로도 간주됩니다.

정사각형을 사용할 수 있습니다. brackets 하위 문자열을 얻기 위해 인덱스와 함께 분할합니다.

var1 = "Guru99!"  
var2 = "Software Testing"  
print ("var1[0]:",var1[0])  
print ("var2[1:5]:",var2[1:5])

**73) 모든 문자열을 설명 opera예를 들어 토르**

끈 opera예를 들어:

Opera바위 산

상품 설명

예

[]

슬라이스 - 주어진 인덱스의 문자를 제공합니다.

a[1]은 Guru라는 단어에서 "u"를 제공합니다(0=G, 1=u, 2=r 및 3=u).

[ : ]

범위 슬라이스-주어진 범위의 문자를 제공합니다.

x [1:3] Guru라는 단어에서 "ur"가 제공됩니다. G인 0을 고려하지 않고 그 이후의 단어를 ur로 간주한다는 것을 기억하십시오.

in

멤버십 - 주어진 문자열에 문자가 있으면 true를 반환합니다.

u는 Guru라는 단어에 존재하므로 1(True)을 제공합니다.

하지에

멤버십 - 주어진 문자열에 문자가 없으면 true를 반환합니다.

나는 Guru라는 단어에 존재하지 않으므로 1을 제공합니다.

r/R

원시 문자열은 이스케이프 문자의 실제 의미를 억제합니다.

r'\n' 인쇄는 \n을 인쇄하고 R'/n' 인쇄는 \n을 인쇄합니다.

% – 문자열 형식에 사용됩니다.

%r - 객체의 표준 문자열 표현을 삽입합니다(예: repr(o)) %s- 객체의 표현 문자열 표현(예: str(o))을 삽입합니다. %d- 표시할 숫자 형식을 지정합니다.

이 코드의 출력은 "guru 99"입니다.

+

2개의 문자열을 연결합니다.

문자열을 연결하고 결과를 제공합니다.

\*

반복

문자를 두 번 인쇄합니다.

**74) 파이썬에서 sleep() 함수의 예를 들어보세요.**

Python의 sleep() 함수 예

import time  
print("Welcome to guru99 Python Tutorials")  
time.sleep(5)  
print("This message will be printed after a wait of 5 seconds")

**75) Python의 타이머 메소드란 무엇입니까?**

타이머는 스레딩과 함께 사용할 수 있는 방법으로, Python 시간 절전 모드와 동일한 기능을 얻는 데 도움이 됩니다.

from threading import Timer

print('Code Execution Started')

def display():  
print('Welcome to Guru99 Tutorials')

t = Timer(5, display)  
t.start()

**76) 달력 클래스의 예를 들어보세요**

캘린더 클래스의 예

import calendar

**Create a plain text calendar**

c = calendar.TextCalendar(calendar.THURSDAY)  
str = c.formatmonth(2025, 1, 0, 0)  
print(str)

**Create an HTML formatted calendar**

hc = calendar.HTMLCalendar(calendar.THURSDAY)  
str = hc.formatmonth(2025, 1)  
print(str)

**loop over the days of a month**

**zeroes indicate that the day of the week is in a next month or overlapping month**

for i in c.itermonthdays(2025, 4):  
print(i)

# The calendar can give info based on local such a names of days and months (full and abbreviated forms)

for name in calendar.month\_name:

print(name)

for day in calendar.day\_name:

print(day)

# calculate days based on a rule: For instance an audit day on the second Monday of every month

# Figure out what days that would be for each month, we can use the script as shown here

for month in range(1, 13):

# It retrieves a list of weeks that represent the month

mycal = calendar.monthcalendar(2025, month)

# The first MONDAY has to be within the first two weeks

week1 = mycal[0]

week2 = mycal[1]

if week1[calendar.MONDAY] != 0:

auditday = week1[calendar.MONDAY]

else:

# if the first MONDAY isn&#x27;t in the first week, it must be in the second week

auditday = week2[calendar.MONDAY]

print("%10s %2d" % (calendar.month\_name[month], auditday))

**77) Python ZIP 파일을 예시로 설명해보세요.**

Python을 사용하면 zip/tar를 빠르게 만들 수 있습니다. archives.

FOLLOwing 명령은 전체 디렉토리를 압축합니다  
shutil.make\_archive(output\_filename, &#x27;zip&#x27;, dir\_name)  
FOLLOwing 명령을 사용하면 원하는 파일을 제어할 수 있습니다. archive  
ZipFile.write(filename)  
Python ZIP 파일의 예

import os  
import shutil  
from zipfile import ZipFile  
from os import path  
from shutil import make\_archive

# Check if file exists

if path.exists(&quot;guru99.txt&quot;):

# get the path to the file in the current directory

src = path.realpath(&quot;guru99.txt&quot;);

# rename the original file

os.rename(&quot;career.guru99.txt&quot;,&quot;guru99.txt&quot;)

# now put things into a ZIP archive

root\_dir,tail = path.split(src)

shutil.make\_archive(&quot;guru99 archive&quot;,&quot;zip&quot;,root\_dir)

# more fine-grained control over ZIP files

with ZipFile(&quot;testguru99.zip&quot;, &quot;w&quot;) as newzip:

newzip.write(&quot;guru99.txt&quot;)

newzip.write(&quot;guru99.txt.bak&quot;)

**78) Python에서 예외의 일반적인 예는 무엇입니까?**

Python에서 예외의 일반적인 예는 다음과 같습니다.

* XNUMX으로 나누기
* 존재하지 않는 파일에 액세스하는 중입니다.
* 호환되지 않는 두 가지 유형 추가
* 존재하지 않는 시퀀스 인덱스에 액세스하려고 합니다.
* 연결이 끊긴 데이터베이스 서버에서 테이블을 제거합니다.
* ATM 이용 가능 금액 이상 출금 가능

**79) 중요한 파이썬 오류를 설명하세요**

중요한 Python 오류는 1) ArithmeticError, 2) ImportError 및 3) IndexError입니다.

* **산술 오류:** ArithmeticError는 모든 산술 예외에 대한 기본 클래스 역할을 합니다. 산술 오류로 인해 발생합니다. opera.
* **가져오기오류:** 존재하지 않는 모듈을 가져오려고 하면 ImportError가 발생합니다. 이러한 종류의 예외는 모듈 이름을 잘못 입력했거나 표준 경로에 없는 모듈을 입력한 경우 발생합니다.
* **색인오류:** 범위를 벗어난 시퀀스를 참조하려고 하면 IndexError가 발생합니다.

**80) 예를 들어 Python의 JSON dump()를 설명하세요.**

Python의 json.dumps()는 Python의 사전 객체를 JSON 문자열 데이터 형식으로 변환하는 메서드입니다. 개체가 문자열 형식이어야 하는 경우 유용합니다. opera구문 분석, 인쇄 등과 같은 작업

예:

import json

x = {  
"name": "Ken",  
"age": 45,  
"married": True,  
"children": ("Alice","Bob"),  
"pets": ['Dog'],  
"cars": [  
{"model": "Audi A1", "mpg": 15.1},  
{"model": "Zeep Compass", "mpg": 18.1}  
]  
}

**sorting result in asscending order by keys:**

sorted\_string = json.dumps(x, indent=4, sort\_keys=True)  
print(sorted\_string)

**81) 예를 들어 JSON에서 Python(디코딩)까지 자세히 설명하세요.**

JSON 문자열 디코딩은 내장된 메소드의 도움으로 수행됩니다. json.loads() & json.load() Python의 JSON 라이브러리.

여기 변환 테이블은 JSON 문자열의 Python 디코딩을 수행하는 데 도움이 되는 Python 개체에 대한 JSON 개체의 예를 보여줍니다.

JSON

Python

목적

딕셔너리

배열

명부

끈

유니 코드

숫자 - 정수

숫자 - 정수, 길이

숫자 - 실제

흙손

참된

참된

거짓

거짓

null로

없음

json.loads 함수를 사용하여 디코딩하는 기본 JSON-Python 예제:

import json # json library imported

**json data string**

person\_data = '{ "person": { "name": "Kenn", "sex": "male", "age": 28}}'

**Decoding or converting JSON format in dictionary using loads()**

dict\_obj = json.loads(person\_data)  
print(dict\_obj)

**check type of dict\_obj**

print("Type of dict\_obj", type(dict\_obj))

**get human object details**

print("Person......", dict\_obj.get('person'))

**82) encode() 메소드에 대한 코드 작성**

encode() 메소드의 코드:

**import JSONEncoder class from json**

from json.encoder import JSONEncoder  
colour\_dict = { "colour": ["red", "yellow", "green" ]}

**directly called encode method of JSON**

JSONEncoder().encode(colour\_dict)

**83) Python Matrix를 생성하기 위해 numpy로 배열용 Python 코드를 작성하세요.**

Python Matrix를 생성하기 위한 numpy 배열 코드

import numpy as np  
M1 = np.array([[5, -10, 15], [3, -6, 9], [-4, 8, 12]])  
print(M1)

**84) 행렬 빼기를 위한 Phyhon 코드를 작성하세요**

행렬 빼기를 위한 Phyhon 코드

import numpy as np  
M1 = np.array([[3, 6, 9], [5, -10, 15], [-7, 14, 21]])  
M2 = np.array([[9, -18, 27], [11, 22, 33], [13, -26, 39]])  
M3 = M1 - M2  
print(M3)

**85) 행렬 곱셈을 위한 코드 작성**

행렬 곱셈 코드

numpy를 np로 가져 오기

M1 = np.array([[3, 6], [5, -10]])  
M2 = np.array([[9, -18], [11, 22]])  
M3 = M1.dot(M2)  
print(M3)

**86) 예를 들어 행렬의 슬라이싱을 설명하세요.**

슬라이싱은 주어진 시작/끝 인덱스를 기반으로 행렬의 요소를 반환합니다.

**슬라이싱 구문은 다음과 같습니다.**

[start:end]

* 시작 인덱스를 지정하지 않으면 0으로 간주합니다. 예를 들어 [:5]는 [0:5]를 의미합니다.
* 끝이 전달되지 않으면 배열의 길이로 사용됩니다.
* 시작/끝 값이 음수이면 배열의 끝에서부터 슬라이싱이 수행됩니다.

행렬의 슬라이싱 작업을 수행하기 전에 먼저 간단한 배열에 슬라이스를 적용하는 방법을 이해하겠습니다.

import numpy as np

arr = np.array([2,4,6,8,10,12,14,16])  
print(arr[3:6]) # will print the elements from 3 to 5  
print(arr[:5]) # will print the elements from 0 to 4  
print(arr[2:]) # will print the elements from 2 to length of the array.  
print(arr[-5:-1]) # will print from the end i.e. -5 to -2  
print(arr[:-1]) # will print from end i.e. 0 to -2

**87) 루프를 통해 평균을 구하는 Python 코드를 작성하세요.**

루프를 통해 평균을 구하는 코드는 다음과 같습니다.

def cal\_average(num):  
sum\_num = 0  
for t in num:  
sum\_num = sum\_num + t

avg = sum\_num / len(num)

**return** avg

print("The average is", cal\_average([18,25,3,41,5]))

**88) 목록 개수에 대한 코드를 작성하세요**

다음은 목록 개수에 대한 코드입니다.

list1 = ['red', 'green', 'blue', 'orange', 'green', 'gray', 'green']  
color\_count = list1.count('green')  
print('The count of color: green is ', color\_count)

**89) 주어진 목록에서 중복된 요소를 어떻게 계산할 수 있나요?**

주어진 목록에서 중복 요소 수 계산

list1 = [2,3,4,3,10,3,5,6,3]  
elm\_count = list1.count(3)  
print('The count of element: 3 is ', elm\_count)

**90) for 루프를 사용하여 목록에 있는 요소의 인덱스를 얻는 코드를 작성하세요**

for 루프를 사용하여 목록에 있는 요소의 인덱스를 얻는 코드:

my\_list = ['Guru', 'Siya', 'Tiya', 'Guru', 'Daksh', 'Riya', 'Guru']  
all\_indexes = []  
for i in range(0, len(my\_list)) :  
if my\_list[i] == 'Guru' :  
all\_indexes.append(i)  
print("Originallist ", my\_list)  
print("Indexes for element Guru : ", all\_indexes)

**91) Python type()의 예를 들어보세요.**

str\_list = "Welcome to Guru99"  
age = 50  
pi = 3.14  
c\_num = 3j+10  
my\_list = ["A", "B", "C", "D"]  
my\_tuple = ("A", "B", "C", "D")  
my\_dict = {"A":"a", "B":"b", "C":"c", "D":"d"}  
my\_set = {'A', 'B', 'C', 'D'}

print("The type is : ",type(str\_list))  
print("The type is : ",type(age))  
print("The type is : ",type(pi))  
print("The type is : ",type(c\_num))  
print("The type is : ",type(my\_list))  
print("The type is : ",type(my\_tuple))  
print("The type is : ",type(my\_dict))  
print("The type is : ",type(my\_set))

**92) 파이썬에서 개행 없이 어떻게 인쇄할 수 있나요?**

Python 3+부터 print()에 end=라는 추가 매개변수가 도입되었습니다. 이 매개변수는 print()에 기본적으로 추가되는 줄바꿈을 제거합니다.

아래의 개행 없는 Python 3 인쇄 예제에서 우리는 문자열이 Python에서 같은 줄에 인쇄되기를 원합니다. 이 작업을 수행하려면 아래 예와 같이 print() 내부에 end=""를 추가하면 됩니다.

print("Hello World ", end="")  
print("Welcome to Guru99 Tutorials")

**93) 개행과 공백 없이 별(\*) 패턴을 인쇄하는 방법은 무엇입니까?**

개행과 공백 없이 별표(\*) 패턴을 인쇄하는 코드:

for i in range(0, 20):  
print('\*', end="")

**GAN에 대해 아는대로 설명해주세요**

>**Generative Adverserial Network**  
Generator(G)와 Discriminator(D)로 역할이 나뉜 두 모델이 경쟁하며 학습하는 구조의 모델을 의미한다. D는 입력되는 데이터가 Real인지 Fake인지 구분하는 역할을 수행하며, G는 Real과 구분하기 어려운 Fake를 생성하여 D를 속인다. Loss의 구성이 흥미롭다. D는 Real 데이터와 Fake 데이터를 섞어, Real과 Fake의 구분 능력을 Loss Function으로 하여 학습하고, G는 D가 Fake를 Real로 인식하도록하는 방향으로 학습한다. 수식으로 설명하면 다음과 같다. minGmaxDLoss(D,G)=CrossEntropy(D(x))+CrossEntropy(D(G(z)))𝑚𝑖𝑛𝐺𝑚𝑎𝑥𝐷𝐿𝑜𝑠𝑠(𝐷,𝐺)=𝐶𝑟𝑜𝑠𝑠𝐸𝑛𝑡𝑟𝑜𝑝𝑦(𝐷(𝑥))+𝐶𝑟𝑜𝑠𝑠𝐸𝑛𝑡𝑟𝑜𝑝𝑦(𝐷(𝐺(𝑧))) z는G가입력으로가정하는데이터분포,latentvector(잠재벡터)이다.

**GAN의 단점은 무엇인가요?**

>Generator와 Discriminator 모델들의 밸런스를 잡기 어렵기 때문에 학습이 대체로 불안정하고 수렴이 어렵다. 학습 완료된 Generator의 성능 평가를 사람이 직접 해야한다.

**LSGAN에 대해 설명해주세요**

>Discriminator가 Real과 Fake를 구분할 때 CrossEntorpy 대신 Least square error를 이용한다. 학습 과정에서 최소화해야하는 Entropy는 모델이 "긴가민가"할 때 높은 값으로 나온다. 이는 역설적으로 모델이 높은 확신도로 맞춘 데이터에 대해서는 거의 학습하지 않는다는 것을 의미한다. 일반적인 Classfication 모델에서는 문제가 안되지만, GAN에서는 이 점이 Generator의 학습을 저해하는 요소이다. 따라서 최대한 "모든" 데이터에 대해 학습할 수 있도록 Squared error를 최소화하는 Loss을 채택한 것이 LSGAN이다.

**Cross Validation은 무엇이고 어떻게 해야하나요?**

가지고 있는 전체 data를 나누어서 형성한 각 batch가 모두 한번씩은 validation set으로 활용되도록 각 iteration 마다 validation set을 번갈아 바꿔 사용한다.  
주로 Train 데이터가 적을 때 사용한다.

**회귀 / 분류시 알맞은 metric은 무엇일까요?**

회귀 : MSE -> Ground Truth와 Pred. 값의 차이(continuous)를 나타낼 수 있는 metric 분류 : Precision, Recall, Accuracy -> 분류 결과(discrete)의 신뢰도를 나타낼 수 있는 통계 metric

**알고 있는 metric에 대해 설명해주세요(ex. RMSE, MAE, recall, precision …)**

* <https://velog.io/@crescent702/%EB%B2%88%EC%97%AD-Evaluation-Metrics-for-Machine-Learning-Models>

**정규화를 왜 해야할까요? 정규화의 방법은 무엇이 있나요?**

정규화 하는이유:모델이 받아들이는 데이터의 크기가 들쑥날쑥하다면 모델이 데이터를 이상하게 해석할 우려가 있어,정규화방법:Min-Max Normalization: 모든 feature들에 대해 최소점을 0, 최대값을 1로 하게끔 Scale을 조정 Z-score Normalization: z-score를 활용하여 정규화. 그러나 정확히 동일한 척도로 정규화된 데이터를 생성하지 못한다. Robust Scaler: z-score와 유사하지만, 평균과 분산이 아닌 median과 quartile값을 사용 Normalizer: 특성 벡터의 유클리드 거리가 1이 되도록 데이터를 조정. 다시 말해, 길이가 1인 원 또는 구로 투영하는 것  
- <https://mole-starseeker.tistory.com/31>

**Local Minima와 Global Minima에 대해 설명해주세요.**

optimal point를 찾아가는 과정에서 최적이라고 의심되지만 실제로 최적이 아닌 지점을 'Local Minima', 실제로 cost function에서 최적인 지점을 'Global Minima'라고 한다.  
- <https://github.com/JerryKwon/Datascience-Interview-Questions-Self-Answer/blob/master/ML/4.%20Local-Global-Minima.md>

**차원의 저주에 대해 설명해주세요**

차원의 저주는 한 샘플을 특정짓기 위해 많은 정보(다양한 차원의)를 수집할수록 오히려 학습이 어려워짐을 말한다.  
- <https://excelsior-cjh.tistory.com/167>

**dimension reduction기법으로 보통 어떤 것들이 있나요?**

기본 방법 : Projection, Manifold Projection 일반적으로, 고차원 데이터의 어떤 특성은 큰 변화가 없고, 어떤 특성은 다른 특성과 연관하여 크게 변하곤 한다. 이를 ‘데이터가 고차원 공간에서 저차원 subspace(부분 공간)에 위치한다’ 라고 하는데, 이것은 고차원 데이터의 특성 중, 일부로 해당 데이터를 표현할 수 있음을 의미한다. 이때, 고차원 데이터를 저차원 subspace로 투영하여 차원을 줄인다. Manifold 데이터가 국소적으로는 유클리드 공간에 있으나, 대역적으로는 다른 위상을 가지는 경우 manifold 공간에 있다고 할 수 있다. 고차원 데이터가 실제로는 저차원 manifold에 있다고 가정하고 학습을 진행하여 문제 난이도를 낮출 수 있으나 항상 성공적이지 않기에 가정 전에 데이터를 살펴보는 주의가 필요하다. 참조: https://excelsior-cjh.tistory.com/167 심화 방법: PCA, NMF, LDA  
- <https://excelsior-cjh.tistory.com/167>

**PCA는 차원 축소 기법이면서, 데이터 압축 기법이기도 하고, 노이즈 제거기법이기도 합니다. 왜 그런지 설명해주실 수 있나요?**

PCA는 분산이 최대가 되도록 하는 축을 찾고, 그 축과 직교하면서 분산이 최대가 되도록 하는 축을 이어 찾아나가는 방식으로 데이터를 간단히 표현합니다. 이 과정에서 차원이 축소되며, 투영 변환을 반대로 수행하면 데이터의 복원이 가능하고, PCA로 찾은 축들 중, 분산이 적은 축들을 제거함으로서 노이즈를 줄일 수 있습니다.  
>

**LSA, LDA, SVD 등의 약자들이 어떤 뜻이고 서로 어떤 관계를 가지는지 설명할 수 있나요?**

>LSA(Latent Semantic Analysis, 잠재의미분석) : SVD를 이용하여 차원을 축소 LDA(Latent Dirichlet Allocation, 잠재디리클레할당) : 주제별 단어분포와 문서별 주제 분포를 학습하여 Topic Modeling 하는 방법 SVD(Singular Value Decomposition) : 특이값 분해 LSA와 SVD를 이용하여 데이터를 분해하고, 함축된 의미(Latent Semantic)를 추출하여 이를 통해 데이터를 압축하거나 차원을 축소한다. 특징 출현 횟수를 사용하는 LSA의 구조에 확률 모델을 도입하여 특징 출현 확률 기반으로 설계된 pLSA(Probabilistic Latent Semantic Analysis)는 SVD대신 NMF(Non-negative Matrix Factorization)를 이용하였다. 이후, LDA는 주제별 단어분포와 문서별 주제분포 모두 고려하되, 디리클레 분포를 따르도록 설계하여 주제를 뽑아낸다.  
- <https://bab2min.tistory.com/585>

**Markov Chain을 고등학생에게 설명하려면 어떤 방식이 제일 좋을까요?**

Markov Chain에서 가장 중요한 개념은 n회의 상태가 n-1회의 상태에만 영향을 받는다는 가정이다. 따라서 독립 시행으로 일어나는 정사면체 주사위, 정육면체 주사위, 동전 던지기 게임을 상태(state)로, 해당 게임의 결과에 따라 다른 게임으로 종목을 바꾸는 것을 전이(transition) 생각할 수 있다. 전이 조건을 정하고, 특정한 순서대로 게임을 하게 될 확률을 계산해보는 것으로 설명할 수 있을 것이다.

**텍스트 더미에서 주제를 추출해야 합니다. 어떤 방식으로 접근해 나가시겠나요?**

간단히 unigram, bigram, TF-IDF로 키워드를 추출하거나 LDA를 이용한다.

**SVM은 왜 반대로 차원을 확장시키는 방식으로 동작할까요? 거기서 어떤 장점이 발생했나요?**

SVM을 비선형 분류 모델로 사용하기 위해 저차원 공간의 데이터를 고차원 공간으로 매핑하여 선형 분리가 가능한 데이터로 변환하여 처리한다.  
- <https://ratsgo.github.io/machine%20learning/2017/05/30/SVM3/>

**다른 좋은 머신 러닝 대비, 오래된 기법인 나이브 베이즈(naive bayes)의 장점을 옹호해보세요.**

결과 도출을 위해 조건부 확률만 계산하면 되므로 매우 빠르고, 메모리를 많이 차지하지 않는다. 데이터가의 특징들이 서로 독립되어 있을 때 좋은 결과를 얻을 수 있다. 데이터의 양이 적더라도 학습이 용이하다.

**Association Rule의 Support, Confidence, Lift에 대해 설명해주세요.**

Support : X와 Y 두 item이 얼마나 자주 발생하는지를 의미 Confidence : X가 발생했을 떄, Y도 포함되어 있는 비율 Lift : X가 발생하지 않았을 때의 Y 발생 비율과 X가 발생했을 때 Y 발생 비율의 대비. 숫자가 1보다 크거나 작은 정도에 따라 연관성을 파악할 수 있다.  
- <https://rfriend.tistory.com/191>

**최적화 기법중 Newton’s Method와 Gradient Descent 방법에 대해 알고 있나요?**

Newton’s Method는 연속적이고 미분 가능한 함수에 대해 무작위 값을 대입해서 결과값의 변화 추이를 통해 원하는 해를 근사하는 방법이다. Gradient Descent 방법은 함수의 극대, 극소를 찾는 방법이며, 마찬가지로 지점에서의 미분값에 따라 대입할 값을 갱신하므로 그 근본적인 방법은 동일하다.

**머신러닝(machine)적 접근방법과 통계(statistics)적 접근방법의 둘간에 차이에 대한 견해가 있나요?**

[저자 주관적 견해] 머신러닝과 통계적 접근방법은 서로 교집합이 큰 종목이다. 이 두 방법의 가장 큰 차이점은 문제 해결 방법의 수립 과정이라고 생각한다. 전자의 경우 머신이 스스로 학습하여 결정하며, 후자의 경우엔 사람이 직접 해결 방법을 설정한다. 통계적으로 사람이 직접 해결 방법을 명확히하기 어려운 문제에 대해 머신러닝 모델을 활용할 수 있다. 하지만 모델 설계 이외에는 제한적으로 사람이 개입되는 머신러닝의 특성상, 기본적으로 오류 가능성을 내포하고 있으며 복잡한 모델일수록 오류가 발생할 수 있는 지점에 대한 예측이 어렵다. 오류 상황에 대한 대응이 필수적인 문제의 경우에는 통계적 접근 방법을 이용하는 것이 더 유리할 수 있다.

**인공신경망(deep learning이전의 전통적인)이 가지는 일반적인 문제점은 무엇일까요?**

Parameter updating 전략과 각종 Normalization의 미비로 인해 step에 따른 Gradient의 움직임을 효과적으로 제어하기 힘들었기 때문에 모델이 깊게 설계할 수 없었다. 따라서 머신러닝을 사용해야하는 문제는 주로 비선형성이 큰 경우가 많았기 때문에, 얕은 구조의 인공신경망 모델로는 성공적인 대응을 할 수 없는 문제가 있었다.

**지금 나오고 있는 deep learning 계열의 혁신의 근간은 무엇이라고 생각하시나요?**

[저자 주관적 견해] 여러 Gradient Descent Optimizer들과 Normalization 기법들이 등장하고, GPU를 통한 부동소수점 연산의 효율화로 인해 Matrix 연산이 빨라지면서 사실상 중단되었던 인공신경망 분야가 한걸음 더 전진한 것이라고 생각한다.

**ROC 커브에 대해 설명해주실 수 있으신가요?**

ROC 커브는 FPR(False Positive Rate)와 TPR(True Positive rate)을 각각 x축과 y축으로 놓은 그래프이다. FPR은 GT(Grount Truth)가 거짓인 케이스에 대해 참이라고 잘못 예측한 비율이며, TPR은 GT가 참인 케이스에 대해 참이라고 잘 예측한 비율이다. 따라서 전자는 작을수록 성능이 좋고, 후자는 클수록 성능이 좋다. 두 Parameter에 대해 성능이 좋을수록 그래프 좌상단의 꼭짓점으로 다가가게 되며, 그래프 하단의 면적이 넓어지는 형태를 가지고 있다.

**여러분이 서버를 100대 가지고 있습니다. 이때 인공신경망보다 Random Forest를 써야하는 이유는 뭘까요?**

일반적으로 각 단계별로 의존적인 end-to-end 인공신경망과 달리 Random Forest는 여러개의 독립적인 decision tree를 이용하여 결과를 투표하는 형식이므로 다수의 서버에서 병렬적인 처리가 용이하다.

**K-means의 대표적 의미론적 단점은 무엇인가요? (계산량 많다는것 말고)**

K-means 알고리즘은 K개의 Cluster를 미리 입력받아 Expectation과 Maximization 단계를 거듭하여 군집화를 수행한다. 단점들은 다음과 같다. 초기에 사용자가 해당 데이터에 몇개의 Cluster가 있는지 알고 있어야 한다. K개의 Cluster를 초기화하는 방법에 따라 결과에 큰 차이가 있다. 클러스터의 모양이 구형이 아닌 경우, 적용이 힘들다. Maximization단계에서 각 데이터들과의 거리를 기준으로 Cluster 중심점의 위치를 업데이트하므로 데이터의 노이즈에 민감하게 반응한다.

**L1, L2 정규화에 대해 설명해주세요**

L1, L2 정규화는 모델 학습시에 값이 너무 큰 파라메터의 영향력을 줄이는 전략이다. L1 정규화는 Cost Function에 가중치의 크기(절대값)를 더해주고, L2 정규화는 가중치 크기의 제곱을 더해줌으로써 가중치가 너무 크지 않은 방향으로 학습을 유도한다. 이것은 학습된 모델의 범용성을 높이는 효과를 준다.  
- <https://light-tree.tistory.com/125>

**XGBoost을 아시나요? 왜 이 모델이 캐글에서 유명할까요?**

XGBoost는 머신러닝에서 각 Feature의 중요도를 계산할 수 있는 툴이다. 문제를 세부 단위로 쪼개어 정확도를 예측하는 부스팅 기법과 모든 리프들이 최종 결과 스코어를 내는 CART(Classification And Regression Tree) 기법을 앙상블하여 사용한다. 학습과 분류가 빠르고 캐글의 머신러닝 경연 우승자 중, 다수가 이 툴을 사용하여 주목받았다.  
- <https://brunch.co.kr/@snobberys/137>

**앙상블 방법엔 어떤 것들이 있나요?**

앙상블은 머신러닝에서 여러 모델들을 학습시켜 각 모델의 결과를 합쳐서 더 나은 결과를 내는 방법을 말한다. 앙상블 방법에는 서로 다른 모델들을 학습시켜 결과를 투표하는 Voting, 같은 모델을 다른 Train data로 학습시키는 Bagging(Bootstrap aggregating)과 Pasting, 여러개의 약한 분류기를 결합하여 높은 성능의 모델을 도출하는 Boosting 등이 있다.  
- <https://excelsior-cjh.tistory.com/166>

**SVM은 왜 좋을까요?**

SVM은 데이터들을 선형 분리하며 최대 마진의 초평면을 찾는 크게 복잡하지 않은 구조이며, 커널 트릭을 이용해 차원을 늘리면서 비선형 데이터들에도 좋은 결과를 얻을 수 있다. 또한 이진 분류 뿐만 아니라 수치 예측에도 사용될 수 있다. Overfitting 경향이 낮으며 노이즈 데이터에도 크게 영향을 받지 않는다.  
- <https://excelsior-cjh.tistory.com/166>

**feature vector란 무엇일까요?**

feature는 data를 구성하는 개별 example를 나타내는 ‘측정가능한 요소’를 나타낸다. feature vector는 한 example이 가지고 있는 feature들로 만든 vector를 의미한다.

**좋은 모델의 정의는 무엇일까요?**

좋은 모델은 여러가지 의미가 있을 수 있다고 생각한다. 상황에 따라 최고의 성능을 내는 모델이 좋은 모델일 수도 있고, 가장 적은 리소스로 적절한 성능을 내는 모델이 좋은 모델일 수도 있다. 일반적으로는 필요한 리소스만으로 좋은 성능을 내는, 최적화가 잘된 모델이 좋은 모델이라 볼 수 있을 것 같다.

**50개의 작은 의사결정 나무는 큰 의사결정 나무보다 괜찮을까요? 왜 그렇게 생각하나요?**

큰 의사결정나무 하나를 쓰는 것보다 50개의 작은 의사결정나무를 사용하여, 그들의 결과를 취합하는 것이 경우에 따라 더 좋은 성능을 발휘할 수 있을 것이다. 비선형성이 강한 문제를 해결하는데 있어, 모든 feature를 사용하여 학습한 하나의 큰 의사결정나무보다 feature 일부를 이용하여 학습된 작은 의사결정나무들이 과적합의 정도가 덜할 것이며, 데이터 노이즈에도 강할 것이기 때문이다.

**스팸 필터에 로지스틱 리그레션을 많이 사용하는 이유는 무엇일까요?**

스팸 필터링 분야에서 좋은 성능을 보이는데다가 해석이 용이하며, 계산 비용도 매우 저렴하기 때문이다.

**OLS(ordinary least squre) regression의 공식은 무엇인가요?**

* <https://darkpgmr.tistory.com/56>

**딥러닝은 무엇인가요? 딥러닝과 머신러닝의 차이는?**

딥러닝은 인공신경망을 깊은 구조로 설계하여 비선형성이 높은 문제들을 해결하는 방법을 통칭한다. 딥러닝은 머신러닝의 일종이며 모델 내부의 결정 과정을 해석하기 불가능한 ‘블랙 박스’구조로 되어있다. 보통, 딥러닝이 일반적인 머신러닝 기법보다 더 많은 데이터를 필요로 하며, 계산량이 압도적으로 많다.

**왜 갑자기 딥러닝이 부흥했을까요?**

[저자 주관적 견해] 딥러닝의 기반 구조인 인공신경망은 1950년대부터 이미 있었던 개념이다. 그 당시에도 문제의 비선형도가 높아지면 인공신경망의 층을 충분히 깊게 설계해야하고, 모델 학습을 안정적으로 수행해야함을 알고 있었다. 하지만 과거에는 이를 실험적으로 뒷받침할 수 있는 하드웨어적 성능이 충분치 못했고 이에 따라 학습 전략 수립에도 차질이 있었다. 이후, GPU를 통한 부동소수점 연산의 효율화와 함께 하드웨어 문제가 완화되어, 깊은 층을 효과적으로 학습할 수 있는 방안들(Batch normalization, Optimizing Strategy 등)에 대한 연구가 활발히 이뤄질 수 있었다. 따라서 기술적 완성도가 높아졌고, 시기적으로도 Big Data 시대를 맞이하며 학습에 필요한 데이터 공급이 이전보다 수월해진 것도 결정적인 영향을 끼쳤다. 하드웨어 성능, 소프트웨어적 전략, 데이터 공급의 3가지 문제가 절묘하게 해결된 것이 딥러닝 부흥의 이유라고 생각한다.

**마지막으로 읽은 논문은 무엇인가요? 설명해주세요**

당신이 읽은 논문 말하세요.

**Cost Function과 Activation Function은 무엇인가요?**

Cost Function은 모델이 생산한 결과와 목표 결과를 어떻게 비교할 것인가를 의미한다. 두 결과의 차이를 의미하는 Cost는 Optimizer에 의해 Parameter가 갱신될 때, Step Size를 얼마나 크게 가져갈 것인가에 결정적 영향을 끼친다. Activation Function은 뉴런이 유입되는 신호로부터 도출하는 값을 정제하는 역할을 한다. Sigmoid, Relu, Hyperbolic Tangent 등의 Activation Function를 통해, 어떤 값을 버리고, 어떤 값을 내보낼지를 결정 할 수 있다.

**Tensorflow, Keras, PyTorch, Caffe, Mxnet 중 선호하는 프레임워크와 그 이유는 무엇인가요?**

PyTorch. 상속을 기반으로 한 쉬운 모델 설계 방법과 직관적인 텐서 컨트롤이 마음에 들기 때문이다.

**Data Normalization은 무엇이고 왜 필요한가요?**

Data Normalization은 입력 데이터의 최소, 최대값을 일정 범위(0~1) 내로 조절하는 것이다. 이를 통해, 특정 입력 데이터가 결과에 대해 과도한 영향을 미칠 수 있는 지위를 획득하는 것을 방지할 수 있고, 모델의 학습을 원활하게 만든다.

**알고있는 Activation Function에 대해 알려주세요. (Sigmoid, ReLU, LeakyReLU, Tanh 등)**

sigmoid, ReLU, LeakyReLU,ELU, PReLU, softplus, tanh, softsign, swish,

* <https://yeomko.tistory.com/39>

**오버피팅일 경우 어떻게 대처해야 할까요?**

Training Data를 더 마련한다. k-fold cross validation을 사용한다. Regularization을 사용한다. Dropout 을 사용한다. Parameter 수를 줄인다.

**하이퍼 파라미터는 무엇인가요?**

하이퍼 파라미터는 모델의 학습에 필요한 수동 설정값이다.

**Weight Initialization 방법에 대해 말해주세요. 그리고 무엇을 많이 사용하나요?**

Sigmoid와 같은 S자 형태의 Activation Function을 사용할 때는, 정규 분포 모양의 Weight Initialization이 효과적이지만 일반적인 상황에서는 Xiavier, ReLu에는 He를 사용한다.

* <https://gomguard.tistory.com/184>

**볼츠만 머신은 무엇인가요?**

볼츠만 머신은 가시층(Visible Layer)와 은닉층(Hidden Layer), 총 두 개의 층으로 신경망을 구성하는 방법이다. 볼츠만 머신은 모든 뉴런이 연결되어 있는 완전 그래프 형태이며, 제한된 볼츠만 머신(RBM)에서는 같은 층의 뉴런들은 연결되어 있지 않은 모양이다. 기본적으로 단층구조이며, 확률 모델이다. 분류나 선형 회귀 분석 등에 사용될 수 있다. 특히 DBN(Deep Belief Network)에서는 RBM들을 쌓아올려, 각 볼츠만 머신을 순차적으로 학습시킨다.

**요즘 Sigmoid 보다 ReLU를 많이 쓰는데 그 이유는?**

Sigmoid는 항상 1보다 작은 수를 리턴한다. 이에 의해 Gradient Vanishing 현상이 일어나, 층을 깊게 설계할 수 없기 때문이다. 또한 계산 비용 측면에서도 ReLU가 Sigmoid에 비해 훨씬 경제적이다.

**Non-Linearity라는 말의 의미와 그 필요성은?**

비선형성. 선형 결합으로 해 집합을 표현할 수 없는 성질을 말한다. 다시 말하면, 결과값이 입력값들의 사칙연산 조합으로 표현되지 않는 것이다. 우리가 해결하고 싶은 대부분의 실제 문제들은 비선형성을 띄고 있다. 하지만 만약 모델이 선형적인 방법론으로 학습한다면 비선형 문제를 풀 수 없을 것이다. 따라서 학습시에 비선형성을 가진 방법론을 추가함으로써 여기에 대응해야 할 필요가 있다.

**ReLU로 어떻게 곡선 함수를 근사하나?**

ReLU는 선형 부분과 비선형 부분이 결합된 모양이다. ReLU가 반복 적용되면서 선형 부분의 결합이 이루어지고, 곡선 함수를 근사할 수 있게 된다.

**ReLU의 문제점은?**

ReLU는 0이하의 입력을 모두 0으로 내보낸다. 이 과정에서 정보의 손실이 불가피하다. 이런 단점을 보완하기 위해 Leaky ReLU와 같이 0이하 값도 출력에 반영하는 활성화 함수들이 등장했다.

**Bias는 왜 있는걸까?**

Bias는 함수의 모양을 변경하지 않고 Shift해주어 정답에 근사할 수 있도록 도와준다.

**Gradient Descent에 대해서 쉽게 설명한다면?**

Gradient Descent는 파라미터에 대해 오차값을 미분하여 그 기울기값(Gradient)를 구하고, 경사가 하강하는 방향으로 파라미터를 업데이트 하는 방법이다.

**왜 꼭 Gradient를 써야 할까? 그 그래프에서 가로축과 세로축 각각은 무엇인가? 실제 상황에서는 그 그래프가 어떻게 그려질까?**

Gradient가 양수이면 올라가는 방향이며 음수이면 내려가는 방향이다. 실제 상황에서는 Gradient 그래프가 0을 중심으로 진동하는 모양이 될 것이다.

**GD 중에 때때로 Loss가 증가하는 이유는?**

실제로 사용되는 GD에서는 Local Optima를 피하기 위해 Momentum 등의 개념을 도입한 RMSprop, Adam 등의 Optimization 전략을 사용한다. 각 Optimization 전략에 따라 Gradient가 양수인 방향으로도 parameter update step을 가져가는 경우가 생길 수 있으며, 이 경우에는 Loss가 일시적으로 증가할 수 있다.

**중학생이 이해할 수 있게 더 쉽게 설명 한다면?**

Gradient Descent는 위 아래로 구불구불한 산길을 눈을 감고 지나면서 지금 밟고 있는 땅의 경사 방향을 토대로 가장 고도가 낮은 지점을 찾아나가는 것이다.

**Back Propagation에 대해서 쉽게 설명 한다면?**

신경망의 최종 단계에서 계산된 오차를 변화량을 바탕으로 이전 단계의 파라미터를 업데이트 하는 방향을 설정하는 방법이다.

**Local Minima 문제에도 불구하고 딥러닝이 잘 되는 이유는?**

Optimization 전략을 통해 local Minima 문제를 어느정도 피할 수 있기 때문이다.

**GD가 Local Minima 문제를 피하는 방법은?**

Momentum 등의 개념을 도입한 RMSprop, Adam 등의 Optimization 전략을 사용한다.

**찾은 해가 Global Minimum인지 아닌지 알 수 있는 방법은?**

Global Minima가 정확히 어디에 존재하는지는 알 수 없다. 다만, 학습에 사용하지 않은 Test Dataset에 대한 성능을 평가하는 것으로 모델이 Global Minima에 가까운지 유추할 수 있다.

**Training 세트와 Test 세트를 분리하는 이유는?**

실제 데이터에 대한 모델의 성능을 평가하기 위해, Training data로 사용했던 데이터를 모델의 평가에 활용하지 않는다.

**Validation 세트가 따로 있는 이유는?**

모델 학습 과정 중, Training data와 분리된 Validation data로 모델을 평가하여 그 결과를 학습에 반영하므로써 Training Data에 대한 Overfitting을 방지하는 효과가 있다.

**Test 세트가 오염되었다는 말의 뜻은?**

Test data에 Training data와 일치하거나, 매우 유사한 데이터들이 포함되어 Test data가 General한 상황에서의 성능 평가를 수행하지 못함을 말한다.

**Regularization이란 무엇인가?**

단순히 Cost function의 값이 작아지는 방향으로 모델을 학습하면 특정 가중치가 과도하게 커지는 현상이 나타날 수 있다. Regularization의 개념은, Cost Function을 계산할 때 가중치의 절대값 만큼을 더해주는 방법으로 특정 가중치의 값이 너무 커지는 현상을 억제하는 것이다.

**Batch Normalization의 효과는?**

은닉층에서의 입력값을 Normalize함으로써 입력 분포를 조정 할 수 있다. 은닉층의 입력 분포가 조정되면 학습 편의성이 개선되어 수렴 속도가 빨라지고, Local Optima를 피할 가능성이 높아진다.  
- <https://light-tree.tistory.com/132>  
- <https://light-tree.tistory.com/139>

**Dropout의 효과는?**

Dropout은 노드들의 연결을 무작위로 끊는 방식으로, 하나의 노드가 너무 큰 가중치를 가져 다른 노드들의 학습을 방해하는 현상을 억제한다. 이를 통해 모델의 일반화 성능을 높이고, Overfitting을 방지한다.

**BN 적용해서 학습 이후 실제 사용시에 주의할 점은? 코드로는?**

학습시에는 각 mini-Batch 단위의 평균과 분산을 이용해 Normalize하지만 실제 사용시에는 네트워크에 입력되는 Batch의 단위가 더 적을 수 있기 때문에 미리 학습 데이터에서 뽑아낸 평균, 분산을 이용해야한다.  
- <https://pytorch.org/docs/stable/nn.html#batchnorm1d>

**GAN에서 Generator 쪽에도 BN을 적용해도 될까?**

일반적인 GAN에서 Generator의 Output Layer와 Discriminator의 Input Layer에는 BN을 적용하지 않는다. 그 이유는 Discriminator가 조작되지 않은 Generator의 결과물의 정확한 값으로 mean, scale을 학습하기 위함이다.

**SGD, RMSprop, Adam에 대해서 아는대로 설명한다면?**

SGD : GD 인데 차이점은 GD 는 전체 데이터를 SGD 는 부분 데이터를 트레이닝 한다. RMSProp : adagrad 에 exponential moving average 방식의 gradient update 를 한다. Adam : RMSProp + Momentum. 물론 이 3개 다 step size wise 한 optimizer 그룹이라고 볼 수 있다. (크게 관성 vs step size 하자면 ㅇㅇ)

**SGD에서 Stochastic의 의미는?**

Stochastic Gradient Descent는 Train Dataset을 전부 이용하는 Batch Gradient Descent와 달리, mini-batch를 이용하여 loss function을 이용한다. ‘Stochastic’은 mini-batch가 전체 train dataset에서 무작위로 선택된다는 것을 의미한다.

**미니배치를 작게 할때의 장단점은?**

Mini-batch의 사이즈가 작으면 한 iteration의 계산량이 적어지기 때문에 step당 속도가 빨라지며 적은 Graphic Ram으로도 학습이 가능하는 장점이 있다. 하지만 Target의 Variance가 큰 경우, mini-batch-size가 너무 작을 때는 학습데이터에 대한 대표성이 떨어져 학습의 일반성(Generalization)이 불리할 수 있다.

**모멘텀의 수식을 적어 본다면?**

속도 업데이트  
vt=βvt−1+(1−β)∇J(θt)

파라미터업데이트  
θt​=θt−1​−αvt​  
- <https://deep-learning-study.tistory.com/157>

**간단한 MNIST 분류기를 MLP+CPU 버전으로 numpy로 만든다면 몇줄일까?**

fc 구현 data loader, optimizer 등등 다 고려 해 보면 ~300줄 정도가 될 듯, 6시간...??  
- <https://github.com/kozistr/ML-Study/blob/master/Questions/machine_learning_q1.md>

**어느 정도 돌아가는 녀석을 작성하기까지 몇시간 정도 걸릴까?**

reference 있이는 6시간?

**Back Propagation은 몇줄인가?**

fully-connected 만 있으면 이 부분 하나에 대한 back-propagation 은 ~10 줄 내외가 될 거다.

**CNN으로 바꾼다면 얼마나 추가될까?**

conv propagation, back-propagation 정보면 생각 해 보면 ~25줄?

**간단한 MNIST 분류기를 TF, Keras, PyTorch 등으로 작성하는데 몇시간이 필요한가?**

???

**CNN이 아닌 MLP로 해도 잘 될까?**

잘-은 된다. 그런데 acc 는 비교적 낮을 것이다.

**마지막 레이어 부분에 대해서 설명 한다면?**

클래스 사이즈 만큼으로 feature 를 줄여서 실제 classification 에 사용한다.

**학습은 BCE loss로 하되 상황을 MSE loss로 보고 싶다면?**

loss 는 BCE 를 쓰고 MSE 는 metric 으로 쓰면 된다.

**만약 한글 (인쇄물) OCR을 만든다면 데이터 수집은 어떻게 할 수 있을까?**

>부분적으로 OCR 되야 할 부분의 이미지와 그에 대한 positional label, (글자 label) 이 필요할 듯

**딥러닝할 때 GPU를 쓰면 좋은 이유는?**

GPU에는 부동소수점 계산에 특화된 수많은 코어가 있어, Matrix Multiplication이 핵심인 Deep Learning에서 병렬 처리를 수행 하기에 유리하다.

**학습 중인데 GPU를 100% 사용하지 않고 있다. 이유는?**

GPU를 100% 활용하지 못하는 경우는 두가지로 나뉜다. 첫번째는 GPU의 그래픽 메모리를 충분히 활용하지 못하는 경우, 두번째는 GPU의 연산 능력을 충분히 활용하지 못하는 경우이다. 그래픽 메모리가 100%가 아닌 경우, 작은 Batch Size가 문제일 수 있다. 또는 Multi-GPU 환경에서 Data Parallel 방법으로 GPU 로드를 분산할 시, 각 GPU의 output을 한 GPU로 모아서 loss 계산을 하여 GPU 메모리 사용량의 불균형이 발생할 수 있다. 연산 능력이 100%가 아닌 경우에는 모델의 계산 과정에서 CPU 병목이 원인일 수 있다.

**GPU를 두개 다 쓰고 싶다. 방법은?**

>TF 같은 경우면 tf.device('/gpu:N') 으로 지정하던가 keras 면 MultiGPUModel 을 쉽게 해준다. 그냥 n\_gpus 에 갯수만 집어 넣으면 된다.

**학습시 필요한 GPU 메모리는 어떻게 계산하는가?**

네트워크 크기 + batch\_size \* 데이터 shape

**. 스택과 큐 설명**

* 공통점: 선형 자료구조, 배열과 연결리스트 등을 사용해 구현할 수 있음.
* 차이점: 스택은 선입후출(LIFO)의 방식. 큐는 선입선출(FIFO)의 방식. 파이썬 리스트는 스택만 지원한다. 스택 2개로 큐를 구현할 수 있다.

**2. 배열과 연결 리스트의 장단점**

* 배열
* 장점 : 메모리에 연속된 공간으로 저장된다 그렇기 때문에 빠른 순회가 가능하다. RandomAccess가 가능하다.
* 단점 : 중간에 데이터를 삽입하거나, 배열의 데이터가 다 찬 상태에서 삽입이 이루어질 때 복사 비용이 들게 된다.
* 연결리스트
* 장점 : 삽입과 삭제 모두 O(1)시간에 가능하다.
* 단점 : 임의의 위치에 한번에 접근할 수 없다.

**3. 연결 리스트에서 한번에 중간값을 찾을 수 있는 방법은?**

1. 두 개의 포인터를 사용한다. 하나는 한 칸씩 방문하고, 다른 하나는 두 칸씩 방문한다.
2. 두 칸씩 방문하는 포인터가 연결 리스트의 마지막에 도달했을때, 한 칸씩 방문하는 포인터가 가리키는 위치가 중간 노드이다.

**4. 원형 연결 리스트인지 확인할 수 있는 방법은?**

투포인터를 이용하여 한개의 포인터는 두개씩 이동, 한개의 포인터는 한개씩 이동하여 두 포인터가 만나게 된다면 리스트가 원형임을 확인할 수 있습니다.

**5. 1에서 100까지의 정수가 있는 배열에서 한개가 중복되었다. 어떻게 찾을까?**

1에서 n까지 정수를 더한 값은 n(n+1)2이다. 배열 안의 숫자를 모두 더한 값에 이 값을 빼면 중복된 값을 찾을 수 있다.  
=> n이 매우 큰 수일 경우, 이진수로 각 수가 나타났는지 체크하는 방법이 있다.

**7. 이진 탐색 트리 설명**

이진 탐색 트리는 이진 탐색과 연결 리스트가 결합된 자료구조이다.  
- 최대 두개의 자식을 가질 수 있다. 왼쪽 서브트리의 모든 값이 부모 노드의 값보다 작고, 오른쪽 서브트리의 모든 값은 부모 노드의 값보다 크다는 특징이 있다.  
- 모든 원소는 Key를 가지며, 키 값은 unique하다.  
- 트리의 높이가 h일때 검색, 삽입, 삭제 연산의 시간복잡도는 O(h)이다.  
- 최악의 경우는 트리의 균형이 무너져있는 경우이며, 이 때 시간복잡도는 O(n)이다.

1. 검색
2. 루트 노드에서 시작한다. 키 값과 찾으려는 값을 비교하고, 찾으려는 값이 더 작으면 왼쪽, 더 크면 오른쪽 자식 노드로 이동한다. 같을 경우 검색에 성공하며, 리프 노드에 도달할 때 까지 찾지 못하면 실패한다.
3. 삽입
4. 검색과 동일하게 이루어진다. 검색에 실패하면 그 자리에 노드를 삽입한다.
5. 삽입의 시간복잡도는 O(h)이다.
6. 삭제
7. (1) 삭제하려는 노드가 단말 노드인 경우  
   단말 노드의 부모 노드를 찾아서 연결을 끊는다.
8. (2) 삭제하려는 노드가 서브트리를 하나만 가지고 있는 경우  
   서브트리를 부모 노드에 붙여주고 노드를 삭제한다.
9. (3) 삭제하려는 노드가 두개의 서브트리를 가지고 있는 경우  
   서브트리에서 삭제하려는 노드와 가장 비슷한 값을 가진 노드를 찾아 해당 위치로 옮긴다.

**8. 해시 테이블에서 Collision발생 시 해결법**

해시에서는 충돌이 발생 할 수도 있는데 최악의 경우에는 O(N), 일반적으로 잘 구현 된 경우에는 O(1)의 시간 복잡도를 가지게 됩니다.

충돌의 해결방식은 **Chaining, Open addressing** 이 있습니다.

* **Chaining** : 같은 주소로 해슁되는 원소를 모두 하나의 연결리스트에 매달아 관리하는 방법 입니다.
* 장점은 연결리스트만 사용하면 되기때문에 복잡한 계산식을 사용할 필요가 개방주소법에 비해 상대적으로 적습니다.
* 해시테이블이 채워질수록 성능저하가 발생할 가능성이 높습니다.
* **Open addressing** : 충돌 발생시 다른 버킷에 저장하는 방식입니다.
* 다른버킷을 선택하는 방법
  + **Linear Probing (선형 탐색)** : 해시 충돌 시 다음 버킷, 혹은 몇개를 건너뛰어 데이터를 삽입하는 방식.
  + **Quadratic Probing (제곱 탐색)** : 해시 충돌 시 다음 버킷, 혹은 몇개를 건너뛴 버킷에 데이터를 삽입하는 방식.
  + **Double Hashing** : 해시 충돌시 다른 해시함수를 한번 더 적용한 결과를 이용하는 방식.
* 장점은 삽입 삭제시 오버헤드가 적고 저장할 데이터가 적을때는 유리합니다.

**9. 그래프와 트리 차이점**

* 그래프: 노드와 각 노드를 이어주는 간선으로 구성된 자료구조. 길찾기에 이용됨.
* 트리: 그래프의 일종이지만, 간선의 방향성이 있고, 사이클이 되면 안된다. 루트노드가 아닌 노드는 모두 부모 노드를 하나씩 갖고 있다. 자식 노드는 최대 2개까지.

**10. 우선순위 큐 구현방법 설명**

우선순위 큐는 Complete Binary Tree로 삽입 연산때는 마지막 위치에 삽입되며 부모 노드와 비교하며 자신의 위치를 찾아간다.  
삭제 연산에는 첫번째 노드를 삭제하며 마지막 노드를 첫번째 노드로 옮겨와 자식 노드와 비교하며 두 자식노드중 가장 조건에 부합한 노드와 비교하며 자신의 위치를 찾아간다.

**11. 해시 테이블 설명**

효율적인 탐색을 위한 자료구조로, Key값을 Value에 대응시킨다.  
- 해시 테이블을 구현하기 위해서 연결 리스트와 해시 함수가 필요하다.  
- 임의의 길이의 값을 해시 함수를 통해 고정된 크기의 값으로 변환한 후, 그 결과를 배열의 인덱스로 사용하여 값을 찾는다.  
- 최악의 경우 O(N), 일반적으로 O(1)의 시간복잡도를 갖는다.  
- 해시 함수의 결과 값은 충돌이 일어날 수 있으며, 이는 Chaning, Open addressing 등의 방식으로 해결한다.

* 균형 이진 탐색 트리로도 구현할 수 있다. 이 경우 탐색하는 시간복잡도는 O(logN)이 된다. 크기가 큰 배열을 미리 할당해 놓지 않아도 되기 때문에, 작은 공간을 사용한다는 장점이 있다.

**12. 배열과 연결리스트의 삽입 삭제 시간 복잡도 설명**

1. 배열에서의 삽입 삭제는 **O(N)** 타임이 소요됩니다. 왜냐하면 삽입이나 이후 요소들을 밀고 당기는 과정이 포함되기 때문입니다.
2. 하지만 연결리스트의 삽입 삭제는 **O(1)** 이 소요됩니다. 삽입위치의 전후 포인터만 조정해주면 되기 때문입니다.

**13. 문자열 검색을 위한 자료구조와 이에 대한 장단점**

Trie 자료구조  
- 장점: 주어진 배열의 길이가 길어도 검색할 문자열의 길이만큼 탐색한다.  
- 단점: 주어진 배열 안의 문자열의 길이가 길 경우 메모리 공간을 많이 차지한다.

**14. 균형 이진 트리의 시간 복잡도**

O(log N)

**15. 스택 두개로 큐를 만드는 방법**

1. 메인 스택과 서브 스택을 만든다.
2. 큐의 enqueue는 메인 스택에 데이터를 push한다.  
   처음으로 큐에서 dequeue을 할때, 메인 스택의 데이터들을 모두 pop해서 서브 스택에 push한다.  
   그러면 메인 스택의 데이터들의 순서가 뒤집히게 된다. 이 때, 서브 스택에서 pop하면 dequeue를 구할 수 있다.  
   또 다시 dequeue해야 한다면 서브 스택에서 pop을 하면 된다.
3. enqueue를 할 때는 서브 스택의 데이터들을 모두 pop하여 메인 스택에 push하고 그 위에 데이터를 push한다.

**16. n개의 배열에서 k번째로 큰 수를 찾는 방법**

1. 퀵소트의 pivot찾기를 활용합니다.
2. pivot을 기준으로 좌측의 수들은 pivot보다 작은수 우측의 수들은 pivot 보다 큰 수 이므로 좌, 우측의 정렬 여부와 상관없이 해당 pivot의 위치는 확정적이므로 pivot의 인덱스가 k-1(인덱스시작이 0이라고 했을때)로 잡히는 경우 k번째 큰 수 라고 할 수 있습니다.
3. max heap을 이용합니다.
4. tree 의 크기가 k가 될 때 까지 입력을 받고 k보다 큰경우 heap에 넣은뒤 한개 씩 삭제하여 tree의 크기를 k로 유지하게합니다.
5. 그렇게 다 입력을 받은 경우 루트노드에 있는 값이 k번째로 큰 수입니다.

**17. 최소 스패닝 트리(Minimum Spanning Tree)에 대해서 설명해주세요.**

MST는 그래프의 Spanning Tree중 간선의 가중치 합이 최소인 Spanning Tree를 의미한다.  
여기서 Spanning Tree는 루프가 없고 모든 그룹 노드를 포함하고 있어야 한다.  
- 대표적인 알고리즘  
- Kruskal's (O(E log E))  
1. 모든 간선을 무게에 따라 오름차순으로 정렬  
2. 가장 작은 간선을 뽑고, 간선이 신장 트리에서 사이클을 이루는지 확인한다. 만약 사이클이 생기지 않는다면 간선을 포함, 아닐 경우에는 삭제.  
3. 신장 트리의 간선이 V-1이 될 때까지 2를 반복.  
- Prim(O(E log V) : 간선을 하나씩 이어나가며 MST를 만들어나가는 알고리즘.

**18. selection sort란?**

O(n^2)의 시간복잡도와 O(1)의 공간복잡도를 가지는 unstable한 정렬이다.  
0 ~ n번째 위치를 순회하며 현재 자리에 들어오기 가장 적합한 노드를 선택해 스왑한다. 그 뒤엔 1 ~ n 번째, 2 ~ n 번째.. 반복적으로 수행한다.

**19. Insertion Sort란?**

O(n^2)의 시간복잡도와 O(1)의 공간복잡도를 가지는 stable한 정렬이다.  
1번째 노드를 그 앞 노드들과 비교하며 조건에 맞다면 스왑을 하고 조건에 맞지 않으면 멈춘다.  
그 다음엔 2번째, 3번째, .. n번째 노드를 반복적으로 수행한다.  
이런 특성상 이미 정렬된 상태의 배열이라면 O(n)의 최선의 시간복잡도를 가진다.

**20. Merge Sort란?**

* 머지소트의 공간복잡도를 O(n/2)로 줄이는 방법
* 필요없는 부분을 제외한다.  
  정렬해야할 그룹이 left = {1, 2, 4}, right = {3, 5, 6} 이고 배열에 {1, 2, 4, 3, 5, 6} 순서로 존재할 때 진짜 정렬해야할 부분은 2, 4, 3 이 있는 부분이다.  
  즉 left에서 right[0] 보다 큰 부분 부터 right 에서 left[last] 보다 작은 부분 까지다.
* 필요한 데이터만 복사한다.  
  정렬해야할 그룹이 left = {3, 4, 5}, right = {1, 2} 일때 임시 버퍼를 buffer[5] 즉 {x, x, x, x, x}의 공간을 할당받지 않고 개수가 많은쪽의 데이터의 크기만큼만 복사한다.(여기서는 left)  
  그렇게 되면 정렬은 다음과 같이 진행할 수 있다.

buffer = {3, 4, 5} //임시로 할당되는 공간  
right = {1, 2} //개념적 공간임 사용하는 공간은 array의 공간을 사용하되 구분짓기 위한  
array = {3, 4, 5, 1, 2} //현재 배열에 들어가있는 데이터의 순서

이 때 정렬은 left의 데이터를 가져왔으므로 앞에서부터 수행해야 한다. (right의 공간을 가져왔으면 뒤에서부터 채워넣어야함)

buffer = {3, 4, 5}  
right = {2}  
array = {1, 4, 5, 1, 2}

buffer = {3, 4, 5}  
right = {}  
array = {1, 2, 5, 1, 2}  
이 때 right가 비게 되므로 기존 Merge Sort의 방식 그대로인 남아있는 곳의 데이터를 모두 복사하는 방식으로 진행

buffer = {}  
right = {}  
array = {1, 2, 3, 4, 5}

O(n2)의 공간 복잡도를 가지게 되지만 Big-O 표기를 하게 되면 O(n)이라고 표현하게 된다.  
그냥 이런 방법도 있다는걸 알아 두시면 좋을꺼 같아요 어짜피 표현은 둘다 O(n)이라고 표현하게 되겠지만요..

**21. quick sort란?**

퀵 소트 또한 **분할 정복을 이용하여 정렬을 수행하는 알고리즘** 입니다.

**pivot point 라는 기준이되는 값을 하나 설정하여 이값을 기준으로 작은 값은 왼쪽 큰값은 오른쪽으로 옮기는 방식으로 정렬을 진행합니다.**

이를 반복하여 분할된 배열의 크기가 1이되면 모두 정렬이 된 것 입니다.

여기서 pivot point는 주로 배열의 값중 맨 앞이나 맨뒤, 중간 혹은 랜덤값으로 정하게 되는데 배열의 맨 앞값이나 맨 뒤의 값으로 정하는 경우 **정렬된 상태에서 최악의 케이스로 시간 복잡도 O(n^2)** 을 가지게 됩니다.

그래서 주로 **배열의 중간값이나 랜덤 값으로 pivot point** 를 잡습니다.

평균적인 시간 복잡도는 **O(nlogn)** 입니다.

**22.list, set, map의 차이**

Java의 Collection을 구현한 대표적인 자료구조로 lisk, set, map이 있다.  
- list  
저장공간에 순차적으로 요소가 저장되는 리스트와 포인터로 다음 노드를 가리켜서 배열을 이루는 링크드 리스트가 있다. 선형적인 자료구조이고, 일반 리스트 같은 경우에는 인덱스로 요소를 구분/저장공간에 접근 할 수 있다.  
- set  
집합 자료구조로, 요소 값의 중복을 허용하지 않는다. Java의 Hashset은 Hash의 key값이 중복되지 않는다는 성질을 이용해 set을 구현한 것이다. (Hashset 시간복잡도 O(1))  
- map  
key와 value로 요소를 저장하는 자료구조로, java에는 대표적으로 HashMap과 TreeMap이 있다. 둘 다 key/value로 값이 구성되어 있지만, HashMap은 해쉬 테이블에 요소가 저장되고, TreeMap은 트리 구조로 저장된다. TreeMap은 요소 간 순서 보장이 필요할 때 사용하고(탐색: O(logn)), HashMap은 검색 위주로 접근할 때 사용한다(탐색: O(1)).

+LinkedHashMap  
HashMap과 비슷하지만, 각 요소가 이전 값과 다음 값의 포인터를 가지고 있어 순서를 이룰 수 있다.

++HashTable  
Map을 구현한 클래스로, Thread safe를 보장한다. key 값에 null을 허용하지 않는다.

**23. 데이터의 개수가 백만개일때 정렬방법**

**외부정렬(External Sort)**를 이용한다. 외부정렬이란 입력크기가 매우커 읽고 쓰기가 오래 걸리는 보조 기억장치에 저장할 수 밖에 없는 상태에서 수행되는 정렬이다.

메모리에서 다룰 수 있는 크기에 데이터를 다루던 정렬은 내부정렬(Internal Sort)로 분류한다.

**24. 그래프를 구현하는 방법**

* 인접 행렬(adjacency matrix)  
  그래프의 연결 관계를 이차원 배열로 나타내는 방식이다. 노드의 개수가 n이라면, n2 크기의 이차원 배열을 만들어야 한다. 각각의 배열 공간에는 행 인덱스에 해당하는 노드와 열 인덱스에 해당하는 노드가 연결되어 있는지 boolean 값으로 저장한다. 양방향 그래프는 간선을 입력받을 때마다 행->열, 열->행 간선을 표시하고, 단방향 그래프는 하나만 표시한다.

장점: 노드 i와 노드 j가 연결되었는지 확인하고 싶을 때 인덱스로 바로 접근이 가능해 O(1)의 시간 복잡도를 가진다.  
단점: 어떤 노드에 연결된 모든 노드에 방문해 보고 싶을 때 나머지 모든 노드와의 관계를 확인해야 해서 적은 수의 간선을 가지는 <b>희소 행렬</b>에는 효율이 좋지 않다. 그리고 항상 n2의 메모리 공간이 필요해 메모리를 많이 사용한다.

* 인접 리스트(adjacency list)  
  그래프의 연결 관계를 링크드 리스트로 나타내는 방식이다. 노드의 개수가 n이라면 n개의 요소를 가지고 있는 링크드 리스트를 만들어야 한다. 각 노드는 간선으로 연결되어 있는 다른 노드를 가리키는 포인터를 가지고 있다.

장점: 인접 행렬에 비해 메모리를 적게 차지한다. 연결된 노드를 확인하려면 포인터로 바로 접근이 가능해 효율적이다.  
단점: 노드 i와 노드 j가 연결되었는지 확인하고 싶을 때 직접 해당 노드까지 이동해보면서 확인해야 해 접근 속도가 느리다.

**25. AVL 트리 만드는 방법 4가지**

[참고](https://ratsgo.github.io/data%20structure&algorithm/2017/10/27/avltree/)

**26. 이중 연결 리스트란?**

이중 연결 리스트의 요소들은 이전 값과 다음 값을 가리키는 포인터를 가지고 있다. 그래서 노드를 양방향으로 탐색할 수 있고, 탐색 시 O(n/2)만큼의 시간복잡도를 가진다. 리스트를 절반으로 나눠서 앞 부분은 첫번째 노드부터, 뒷 부분은 마지막 노드부터 거꾸로 탐색할 수 있기 때문이다. 일반 연결 리스트보다 효율적으로 queue를 구현할 수 있다. 변수를 하나 더 사용해야 해서 메모리를 더 많이 사용한다는 단점이 있지만 장점이 더 크다.

**27. 1부터 100까지 정렬한다면 어떤 정렬 알고리즘을 써야 할까?**

Counting Sort 를 이용한다. 만약 숫자의 범위가 크다면 radixsort가 좋을 수도 있다