perplexity

위 내용을 토대로 넘파이에서 학습해야하는 주요한 것들을 키워드별로 분류하고 주석등으로 이해하기 쉽게 풀어 설명하고 정리해줘

네, 제공해주신 `01_numpy.ipynb` 파일의 내용을 바탕으로 NumPy 학습 시 반드시 알아야 할 핵심 기능들을 키워드별로 분류하고, 각 기능의 역할과 중요성을 이해하기 쉽게 정리해 드리겠습니다.

NumPy 핵심 기능 요약 (학습 키워드별)

NumPy는 데이터 분석과 머신러닝의 가장 기본적인 도구입니다. 대규모 숫자 데이터를 빠르고 효율적으로 처리하기 위한 배열(Array) 객체와 관련 함수들을 제공합니다. 아래는 꼭 학습해야 할 핵심 기능들입니다.

1. 배열 생성 및 초기화

가장 먼저 데이터를 NumPy가 처리할 수 있는 '배열' 형태로 만드는 방법을 알아야 합니다.

키워드 (함수)	설명
`np.array()`	파이썬의 리스트나 튜플을 NumPy 배열로 변환하는 가장 기본적인 방법입니다. 데이터 분석의 시작점입니다[1].
`np.arange()`	`range()` 함수와 비슷하지만, 정수뿐만 아니라 소수점 단위의 간격(step)도 지정할 수 있어 연속적인 데이터를 만드는 데 유용합니다[1].
`np.zeros()`, `np.ones()`	배열을 만들 때 모든 요소를 각각 0 또는 1로 채웁니다. 특정 형태로 배열의 틀을 먼저 잡을 때 사용됩니다[1].
`np.linspace()`	시작 값과 끝 값 사이를 지정한 개수만큼 균일한 간격으로 나눈 배열을 생성합니다. 그래프를 그리거나 데이터 구간을 나눌 때 편리합니다[1].
`np.random.*`	난수(무작위 수)로 채워진 배열을 생성합니다. `np.random.rand()`는 균등분포, `np.random.randn()`은 정규분포, `np.random.randint()`는 정수 난수를 만듭니다. 머신러닝 모델의 초기 가중치를 설정하거나 데이터를 무작위로 섞을 때 필수적입니다[1].

2. 배열의 형태 변환 (Reshaping)

데이터의 차원을 바꾸거나 구조를 재배열하는 작업은 분석 목적에 맞게 데이터를 가공하는 데 매우 중요합니다.

키워드 (함수/속성)	설명
`.reshape()`	배열의 전체 요소 수는 유지하면서 차원과 모양을 변경합니다. 예를 들어, 12개의 요소를 가진 1차원 배열을 3x4 크기의 2차원 배열(행렬)로 변환할 수 있습니다. 이미지 데이터 처리나 신경망 입력 데이터 구성에 핵심적입니다[1].

키워드 (함수/속성)	설명
`.T` **(Transpose)**	배열의 행과 열을 서로 맞바꿉니다. (3x4 배열 → 4x3 배열). 행렬 연산에서 자주 사용됩니다[1].
`.flatten()`, `.ravel()`	다차원 배열을 1차원으로 평평하게 만듭니다. `.flatten()`은 새로운 복사본을 만들고, `.ravel()`은 원본 배열의 '뷰(view)'를 반환하여 더 빠르지만 원본 데이터에 영향을 줄 수 있습니다[1].
`np.expand_dims()`, `np.squeeze()`	`expand_dims()`는 배열에 불필요한 차원을 추가하고, `squeeze()`는 불필요한 차원을 제거합니다. 딥러닝에서 모델이 요구하는 입력 데이터의 차원(shape)을 맞출 때 매우 유용합니다[1].

3. 인덱싱(Indexing)과 슬라이싱(Slicing)

배열에서 원하는 데이터만 정확하고 효율적으로 골라내는 기술입니다.

키워드	설명
기본 인덱싱/ 슬라이싱	`arr[행, 열]` 또는 `arr[시작:끝:간격]` 형태로 특정 위치의 데이터나 특정 범위의 데이터를 잘라냅니다. 파이썬 리스트보다 훨씬 강력하며, 다차원 데이터의 특정 부분을 손쉽게 추출할 수 있습니다[1].
불리언 인덱싱	`arr[arr > 5]`와 같이 조건식을 사용해 True에 해당하는 요소들만 추출하는 방식입니다. **데이터 필터링**에 가장 많이 사용되는 강력한 기능입니다[1].
팬시 인덱싱	`arr[]`처럼 인덱스 번호가 담긴 리스트(배열)를 사용해 원하는 위치의 요소들을 순서에 상관없이 자유롭게 추출하는 방식입니다[1].

4. 연산 (Operations)

NumPy의 진정한 강점은 반복문 없이 배열 전체를 대상으로 한 번에 연산하는 능력에 있습니다.

키워드	설명
요소별 연산	`+`, `*` 등 일반적인 연산자를 사용하면 배열의 같은 위치에 있는 요소끼리 한 번에 계산됩니다. for문을 사용하지 않아 코드가 간결해지고 속도가 매우 빠릅니다[1].
브로드캐스팅	크기가 다른 배열끼리 연산할 때, NumPy가 자동으로 작은 배열의 형태를 큰 배열에 맞춰 확장한 뒤 연산을 수행하는 기능입니다. 예를 들어, 행렬의 모든 행에 동일한 벡터를 더하는 작업을 간단하게 처리할 수 있습니다[1].
유니버설 함수 (Ufunc)	`np.sqrt()`(제곱근), `np.log()`(로그), `np.sin()`(사인) 등 배열의 모든 요소에 대해 특정 수학 함수를 적용하는 함수들입니다. 이 역시 반복문 없이 매우 빠르게 동작합니다[1].

5. 통계 및 집계

데이터의 특징을 파악하기 위한 필수적인 기능들입니다.

키워드 (함수)	설명
기본 통계 함수	`np.sum()`, `np.mean()`, `np.std()`, `np.min()`, `np.max()` 등을 사용해 전체 데이터의 합계, 평균, 표준편차, 최소/최대값 등을 손쉽게 계산할 수 있습니다[1].

키워드 (함수)	설명
축 (Axis) 기준 연산	`axis=0`은 열(세로) 방향, `axis=1`은 행(가로) 방향으로 연산을 지정합니다. 예를 들어, `scores.mean(axis=1)`는 2차원 성적표 데이터에서 각 학생(행)의 평균 점수를 계산합니다[1].
고급 통계 함수	`np.median()`(중앙값), `np.percentile()`(백분위수), `np.average()`(가중 평균) 등 데이터의 분포를 더 깊이 이해하기 위한 함수들을 제공합니다[1].
누적 함수	`np.cumsum()`(누적 합계), `np.cumprod()`(누적 곱)를 통해 데이터의 누적 변화량을 추적할 수 있습니다[1].

⁂

1. 01_numpy.ipynb