자연어 처리 시작하기

1-1 한글 폰트 설정

1-2 한글 적용 확인

1-3 konlpy 소개 및 설치

1-4 한글 엔진을 이용한 간단한 예제

1-5 LLM과 konlpy

Last Update 24.11

In [2]:
```
import matplotlib as mpl            # 기본 설정 만지는 용도
import matplotlib.pyplot as plt     # 그래프 그리는 용도
import matplotlib.font_manager as fm # 폰트 관련 용도
```

colab 환경에서 한글 적용을 위한 나눔 고딕 설치

In [3]:
```
### 나눔 고딕 설치
!apt-get update -qq   # 설치를 업데이트   -qq  : 로그를 최소한으로
!apt-get install fonts-nanum* -qq # 설치한다. fonts-nanum*   =>  ttf-nanum, ttf-
```

```
W: Skipping acquire of configured file 'main/source/Sources' as repository 'http
s://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide it (so
urces.list entry misspelt?)
Selecting previously unselected package fonts-nanum.
(Reading database ... 123623 files and directories currently installed.)
Preparing to unpack .../fonts-nanum_20200506-1_all.deb ...
Unpacking fonts-nanum (20200506-1) ...
Selecting previously unselected package fonts-nanum-coding.
Preparing to unpack .../fonts-nanum-coding_2.5-3_all.deb ...
Unpacking fonts-nanum-coding (2.5-3) ...
Selecting previously unselected package fonts-nanum-eco.
Preparing to unpack .../fonts-nanum-eco_1.000-7_all.deb ...
Unpacking fonts-nanum-eco (1.000-7) ...
Selecting previously unselected package fonts-nanum-extra.
Preparing to unpack .../fonts-nanum-extra_20200506-1_all.deb ...
Unpacking fonts-nanum-extra (20200506-1) ...
Setting up fonts-nanum-extra (20200506-1) ...
Setting up fonts-nanum (20200506-1) ...
Setting up fonts-nanum-coding (2.5-3) ...
Setting up fonts-nanum-eco (1.000-7) ...
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...
```

In [5]:
```
path = '/usr/share/fonts/truetype/nanum/NanumGothicEco.ttf' # 나눔 고딕 폰트체
font_name = fm.FontProperties(fname=path, size=10).get_name()
print(font_name)
plt.rc('font', family=font_name)

fm.fontManager.addfont(path) # register the font
```

```
NanumGothic Eco
```

In [6]:
```
import matplotlib as mpl # 기본 설정 만지는 용도
import matplotlib.pyplot as plt # 그래프 그리는 용도
import matplotlib.font_manager as fm # 폰트 관련 용도
import numpy as np

path = '/usr/share/fonts/truetype/nanum/NanumGothicEco.ttf' # 설치된 나눔글꼴중
font_name = fm.FontProperties(fname=path, size=10).get_name()
print(font_name)
plt.rc('font', family=font_name)

## 음수 표시되도록 설정
mpl.rcParams['axes.unicode_minus'] = False
```

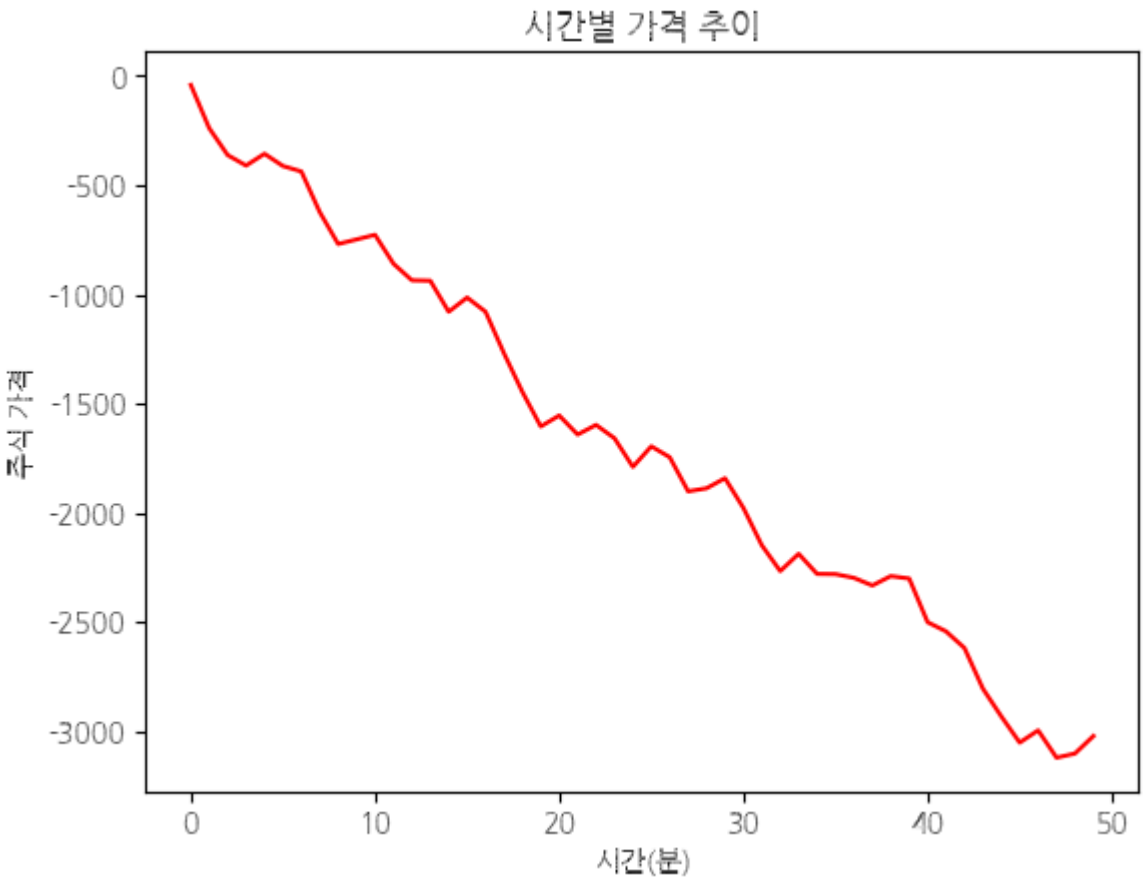
```
NanumGothic Eco
```

In [7]:
```
# 데이터 준비
data = np.random.randint(-200, 100, 50).cumsum()
data
```

Out[7]:
```
array([  -41,  -239,  -362,  -410,  -356,  -412,  -437,  -623,  -769,
        -748,  -727,  -859,  -935,  -937, -1079, -1014, -1079, -1269,
       -1446, -1604, -1554, -1641, -1597, -1658, -1789, -1695, -1745,
       -1901, -1887, -1841, -1977, -2148, -2266, -2187, -2279, -2280,
       -2297, -2332, -2289, -2300, -2500, -2542, -2618, -2804, -2931,
       -3051, -2995, -3120, -3101, -3021])
```

In [8]:
```
# 그래프를 그려보자. 이번에는 정상
plt.plot(range(50), data, 'r')
plt.title('시간별 가격 추이')
plt.ylabel('주식 가격')
plt.xlabel('시간(분)')
plt.style.use('seaborn-pastel')
plt.show()
```

```
<ipython-input-8-b59247eb28e3>:6: MatplotlibDeprecationWarning: The seaborn style
s shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to
the styles shipped by seaborn. However, they will remain available as 'seaborn-v0
_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-pastel')
```



1-3 konlpy 설치 및 소개

- **한국어 자연어 처리 라이브러리**: konlpy는 한국어 텍스트를 처리하기 위한 파이썬 라이브러리로, 형태소 분석, 품사 태깅, 구문 분석 등의 기능을 제공합니다.
- **다양한 형태소 분석기 지원**: konlpy는 여러 형태소 분석기(Mecab, Okt, Komoran, Hannanum, Kkma)를 지원하여, 프로젝트의 요구에 맞는 분석기를 선택할 수 있습니다.
- **오픈소스 프로젝트**: konlpy는 오픈소스로 개발되어 있으며, 커뮤니티의 기여를 통해 지속적으로 개선되고 있습니다.
- **사용자 친화적 인터페이스**: 파이썬 기반으로 직관적이고 사용하기 쉬운 API를 제공하여, 초보자도 쉽게 접근할 수 있습니다.
- **한국어 처리에 특화**: 한국어의 복잡한 문법과 어휘 구조를 효과적으로 처리할 수 있는 기능을 갖추고 있어, 한국어 자연어 처리 작업에 적합합니다.

KoNLPy를 LLM과 함께 사용하는 이유

1. 한국어에 대한 세밀한 전처리

LLM도 한국어를 다룰 수 있지만, KoNLPy는 형태소 분석, 어미 처리, 조사 분리 등 한국어 특유의 언어적 요소를 세밀하게 다룹니다. 대규모 한국어 데이터에서 정확한 전처리를 할 때는 KoNLPy 같은 한국어 전용 도구가 유리할 수 있다.

2. 성능과 비용 효율성

LLM은 계산 자원이 많이 필요하기 때문에, 작은 프로젝트에서는 비용 부담이 될 수 있습니다. KoNLPy 같은 전처리 도구는 가벼운 분석이나 간단한 모델에 적합하며 빠르게 처리할 수 있다.

3. 커스텀 전처리 기능

KoNLPy는 LLM보다 텍스트 전처리에 대한 유연성이 큽니다. 예를 들어 특정 품사만을 추출하거나, 불용어를 맞춤형으로 설정하여 제거하는 등 맞춤형 텍스트 처리에 유용하다.

- pip install konlpy

In [11]:
```
!pip install konlpy
```

```
Collecting konlpy
  Downloading konlpy-0.6.0-py2.py3-none-any.whl.metadata (1.9 kB)
Collecting JPype1>=0.7.0 (from konlpy)
  Downloading JPype1-1.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.9 kB)
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from konlpy) (4.9.4)
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.10/dist-packages (from konlpy) (1.26.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from JPype1>=0.7.0->konlpy) (24.1)
Downloading konlpy-0.6.0-py2.py3-none-any.whl (19.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.4/19.4 MB 73.0 MB/s eta 0:00:00
Downloading JPype1-1.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (488 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 488.6/488.6 kB 30.1 MB/s eta 0:00:00
Installing collected packages: JPype1, konlpy
Successfully installed JPype1-1.5.0 konlpy-0.6.0
```

In [12]:
```
import nltk
import matplotlib.pyplot as plt
import numpy as np
import konlpy
```

꼬꼬마를 이용한 분석

- 품사 태깅, 구문 분석
- 형태소 분석
- http://kkma.snu.ac.kr/documents/index.jsp?doc=postag : 한글 형태소 분석기 품사 태그표
- (예) NNG : 일반 명사, XSV : 동사 파생 접미사, EFN : 평서형 종결 어미
- 문장 단위 분리, 명사 추출

In [13]:
```
from konlpy.tag import Kkma

# Kkma 객체 생성
kkma = Kkma()

# 분석할 문장
text = "안녕하세요. 저는 자연어 처리를 배우고 있습니다."
```

```
# 형태소 분석
morphs = kkma.morphs(text)
print("형태소 분석:", morphs)

# 품사 태깅
pos = kkma.pos(text)
print("품사 태깅:", pos)

# 구문 분석
sentences = kkma.sentences(text)
print("구문 분석:", sentences)
```

```
형태소 분석: ['안녕', '하', '세요', '.', '저', '는', '자연어', '처리', '를', '배우', '고', '있', '습니다', '.']
품사 태깅: [('안녕', 'NNG'), ('하', 'XSV'), ('세요', 'EFN'), ('.', 'SF'), ('저', 'NP'), ('는', 'JX'), ('자연어', 'NNG'), ('처리', 'NNG'), ('를', 'JKO'), ('배우', 'VV'), ('고', 'ECE'), ('있', 'VXV'), ('습니다', 'EFN'), ('.', 'SF')]
구문 분석: ['안녕하세요.', '저는 자연어 처리를 배우고 있습니다.']
```

문장, 명사 추출

In [14]:
```
from konlpy.tag import Kkma
k = Kkma()

sen_extra = k.sentences("안녕하세요! 오늘은 한글 분석을 시작합니다.")
nouns_ext = k.nouns("안녕하세요! 오늘은 한글 분석을 시작합니다.")
print(sen_extra)
print(nouns_ext)
```

```
['안녕하세요!', '오늘은 한글 분석을 시작합니다.']
['안녕', '오늘', '한글', '분석']
```

또 다른 한글 엔진 사용해 보기

- Hannanum
- Okt (예전 - Twitter)
- Kkma
- Mecab
- Komoran

In [15]:
```
# KoNLPy 라이브러리 설치가 필요하다면 주석을 제거하고 실행하세요
# !pip install konlpy

from konlpy.tag import Okt

# 1. 분석기 객체 생성
okt = Okt()

# 예제 텍스트
text = "한국어 텍스트 데이터 분석은 자연어 처리에서 중요한 부분입니다."

# 2. 토큰화 (형태소 분석)
tokens = okt.morphs(text)
print("토큰화 결과:", tokens)

# 3. 품사 태깅
```

```
pos_tags = okt.pos(text)
print("품사 태깅 결과:", pos_tags)

# 4. 불용어 제거
# 불용어 리스트 (필요에 따라 확장 가능)
stopwords = ["은", "는", "이", "가", "에서", "부분", "입니다"]

# 불용어가 제거된 단어 리스트 생성
filtered_tokens = [word for word in tokens if word not in stopwords]
print("불용어 제거 결과:", filtered_tokens)
```

```
토큰화 결과: ['한국어', '텍스트', '데이터', '분석', '은', '자연어', '처리', '에서', '중요한', '부분', '입니다', '.']
품사 태깅 결과: [('한국어', 'Noun'), ('텍스트', 'Noun'), ('데이터', 'Noun'), ('분석', 'Noun'), ('은', 'Josa'), ('자연어', 'Noun'), ('처리', 'Noun'), ('에서', 'Josa'), ('중요한', 'Adjective'), ('부분', 'Noun'), ('입니다', 'Adjective'), ('.', 'Punctuation')]
불용어 제거 결과: ['한국어', '텍스트', '데이터', '분석', '자연어', '처리', '중요한', '.']
```

영어와 다국어 자연어 처리에 널리 사용되는 라이브러리 - nltk

In [17]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.5)
```

In [19]:
```
# NLTK(Natural Language Toolkit)에서 제공하는 Punkt 토크나이저 모델을 다운로드하는
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[19]: True

`punkt` 데이터의 역할

- **문장 토큰화**: 텍스트를 문장 단위로 분할하여 리스트로 반환합니다.
- **단어 토큰화**: 각 문장을 단어 단위로 분할합니다.

In [20]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

# 영어 텍스트 예제
text = "Natural Language Processing with NLTK is fun and educational."

# 텍스트 토큰화
tokens = word_tokenize(text)
```

```
print("토큰화 결과:", tokens)

# 불용어 제거 (영어 기준)
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print("불용어 제거 결과:", filtered_tokens)
```

```
토큰화 결과: ['Natural', 'Language', 'Processing', 'with', 'NLTK', 'is', 'fun', 'and', 'educational', '.']
불용어 제거 결과: ['Natural', 'Language', 'Processing', 'NLTK', 'fun', 'educational', '.']
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Ref

- KoNLPy : 파이썬 한국어 NLP
- https://www.konlpy.org/en/latest/
- konlpy를 사용한 사용 예시
  - https://konlpy.org/ko/latest/examples/
- 사전 : https://konlpy.org/ko/latest/data/