



파이썬 문자열에 대한 기본부터 고급까지 모두 알려줘.

좋아요! 문자열(String)은 파이썬에서 매우 중요한 자료형입니다. 초보부터 고급까지 **순차적으로** 설명드릴게요. 워크북이나 교육자료처럼 단계별로 구성해드릴게요.

## 1단계: 문자열 기초

### 문자열이란?

- 문자의 집합: "hello" 또는 'world'
- 작은따옴표('), 큰따옴표(") 모두 사용 가능

```
python
```

```
s1 = "Hello"  
s2 = 'World'
```

### 여러 줄 문자열

```
python
```

```
s3 = """여러 줄 문자열"""  
s4 = '''줄바꿈도 포함됨'''
```

## 2단계: 문자열 기본 연산

### 연결 (Concatenation)

```
python
```

```
a = "Hello"  
b = "World"  
print(a + " " + b) # Hello World
```

### 반복

```
python
```

```
print("Hi!" * 3) # Hi!Hi!Hi!
```

### 길이 구하기

python

```
len("Python") # 6
```

## 3단계: 인덱싱 & 슬라이싱

### 인덱싱 (문자 하나 추출)

python

```
word = "Python"
print(word[0]) # 'P'
print(word[-1]) # 'n'
```

### 슬라이싱 (문자열 일부 추출)

python

```
print(word[0:2]) # 'Py'
print(word[2:]) # 'thon'
print(word[:]) # 'Python'
```

## 4단계: 문자열 메서드 (자주 쓰는 것만)

메서드	설명
<code>.lower()</code>	소문자로
<code>.upper()</code>	대문자로
<code>.strip()</code>	양쪽 공백 제거
<code>.replace(a, b)</code>	a를 b로 교체
<code>.split()</code>	문자열을 리스트로 분리
<code>.join()</code>	리스트를 문자열로 결합
<code>.find(), .index()</code>	문자열 위치 찾기
<code>.count()</code>	부분 문자열 개수 세기

python

```
s = " Python is Fun "
s.strip().lower().replace("fun", "awesome") # 'python is awesome'
```

## 5단계: f-string과 포맷

### f-string (가장 추천)

```
python

name = "Alice"
age = 25
print(f"My name is {name} and I'm {age} years old.")
```

## ✓ format 방식

```
python

print("My name is {} and I'm {} years old.".format(name, age))
```

## 🧠 6단계: 고급 문자열 처리

### ✓ 이스케이프 문자

- \n (줄바꿈), \t (탭), \', \", \\\

```
python

print("Hello\nWorld")
print("Path: C:\\Users\\User")
```

### ✓ 정렬 및 채우기

```
python

print("Hi".center(10, "-")) # ---Hi-----
print("Hi".ljust(10, "."))  # Hi.....
print("Hi".rjust(10, "*"))  # *****Hi
```

### ✓ 조건 검사

```
python

"Python123".isalnum() # True
"123".isdigit()       # True
"abc".isalpha()       # True
```

## 🔪 7단계: 정규표현식과 문자열

```
python

import re
text = "Email: test123@example.com"
match = re.search(r'\w+@\w+\.\w+', text)
if match:
    print(match.group()) # test123@example.com
```

## 8단계: 실전 예제

python

```
def extract_domain(email):  
    return email.split("@")[1]  
  
extract_domain("user@naver.com") # 'naver.com'
```

## 퀴즈 (초~고급)

1. "Python is easy"에서 easy만 추출하는 슬라이싱 코드는?
2. "Hello World"에서 'world'로 바꾸는 방법은?
3. f-string을 사용해 이름과 나이를 출력하는 코드를 작성해보세요.
4. re 모듈을 이용해 "전화: 010-1234-5678"에서 전화번호만 추출하는 정규식 코드는?

## 요약

구분	주요 내용
기초	문자열 생성, 연결, 반복
중급	슬라이싱, 메서드, f-string
고급	정렬, 검사, 정규표현식 활용

필요하시면 워크북 PDF, 실습 파일, 퀴즈 자동채점 GUI 등도 연계해드릴 수 있어요.  
이제 어느 단계부터 더 자세히 다뤄볼까요?



실무자료와 연계해서 샘플

좋습니다! 문자열 개념을 실무에 활용할 수 있도록, 아래에 분야별 실전 샘플 예제를 제공드릴게요. 모두 실무에서 자주 사용하는 문자열 처리 패턴입니다.

각 예제는 문제 → 코드 → 해설 순서로 구성되어 있어 워크북이나 강의에도 활용할 수 있습니다.

## 실무 문자열 처리 예제 모음

### 1. 엑셀 파일명 일괄 정리

문제:

"2024년\_매출보고서\_최종(수정본).xlsx" 같은 파일들이 수십 개 있다. 이 파일 이름에서 '최종', (수정본) 등을 제거하여 깔끔하게 만들고 싶다.

**코드:**

```
python

files = [
    "2024년_매출보고서_최종(수정본).xlsx",
    "2023년_인건비_최종본(검토).xlsx"
]

cleaned = [f.replace("최종", "").replace("(수정본)", "").replace("(검토)", "") for f in files]
print(cleaned)
```

**결과:**

```
CSS

['2024년_매출보고서_.xlsx', '2023년_인건비_본.xlsx']
```

**해설:**

`replace()`를 활용하여 여러 키워드를 동시에 정리할 수 있고, `list comprehension`으로 일괄 처리.



## 2. 이메일 도메인만 추출하기 (고객DB 정리)

**문제:**

고객 데이터 중 이메일 주소에서 도메인만 추출해 통계 자료에 활용하고자 한다.

**코드:**

```
python

emails = ["kim@naver.com", "lee@gmail.com", "choi@kakao.com"]

domains = [email.split("@")[1] for email in emails]
print(domains)
```

**결과:**

```
CSS

['naver.com', 'gmail.com', 'kakao.com']
```

**해설:**

`split()`을 이용한 문자열 분해는 고객 DB에서 이메일 분류, 고객 그룹 분류 등에 자주 쓰임.



## 3. 공공데이터 API에서 값 추출

**문제:**

아래와 같은 JSON 응답에서 "addr1"의 값을 추출하라.

```
json

{
  "response": {
    "body": {
      "items": {
        "item": [
          {"facItnm": "서울캠핑장", "addr1": "서울시 강남구"},
          {"facItnm": "부산야영장", "addr1": "부산시 해운대구"}
        ]
      }
    }
  }
}
```

**코드:**

```
python

data = {
  "response": {
    "body": {
      "items": {
        "item": [
          {"facItnm": "서울캠핑장", "addr1": "서울시 강남구"},
          {"facItnm": "부산야영장", "addr1": "부산시 해운대구"}
        ]
      }
    }
  }
}

addresses = [x["addr1"] for x in data["response"]["body"]["items"]["item"]]
print(addresses)
```

**결과:**

```
css

['서울시 강남구', '부산시 해운대구']
```

**해설:**

딕셔너리 내 리스트 안 문자열 추출 = API 활용에서 빈번히 사용됨.



## 4. 로그 파일 분석: 오류 메시지만 추출

**문제:**

아래 로그 파일에서 **ERROR**로 시작하는 줄만 추출하라.

```
csharp

[INFO] 시스템 시작
[ERROR] DB 연결 실패
[INFO] 사용자 로그인
[ERROR] 파일 읽기 실패
```

**코드:**

python

```
logs = """
[INFO] 시스템 시작
[ERROR] DB 연결 실패
[INFO] 사용자 로그인
[ERROR] 파일 읽기 실패
"""

errors = [line for line in logs.split("\n") if line.startswith("[ERROR)")]
print(errors)
```

**결과:**

CSS

```
['[ERROR] DB 연결 실패', '[ERROR] 파일 읽기 실패']
```

**해설:**

텍스트 로그 분석 시 `split()` + 조건 필터링 조합이 유용.



## 5. 정규표현식으로 전화번호 추출

**문제:**

"문의: 010-1234-5678"에서 전화번호만 추출하라.

**코드:**

python

```
import re
text = "문의: 010-1234-5678"
match = re.search(r"\d{3}-\d{4}-\d{4}", text)
if match:
    print(match.group())
```

**결과:**

010-1234-5678

**해설:**

정규표현식 `\d{3}-\d{4}-\d{4}`는 숫자 패턴을 추출하는 실무 대표예.



## 6. 보고서에서 특정 키워드 포함 문장 찾기

**문제:**

보고서 문장에서 '변경'이 포함된 문장만 출력하라.

**코드:**

python

```
report = """이번 분기 계획이 변경되었습니다.
매출 목표는 동일합니다.
비용 구조도 일부 변경 예정입니다."""

lines = [line for line in report.split("\n") if "변경" in line]
print(lines)
```

**결과:**

CSS

```
['이번 분기 계획이 변경되었습니다.', '비용 구조도 일부 변경 예정입니다.']
```

**해설:**

단순 `in` 연산자와 `split("\n")`으로 키워드 기반 필터링 가능.

## 정리: 실무에서 문자열이 중요한 이유

분야	활용 예
파일/디렉토리 관리	이름 정리, 확장자 필터링
고객 DB	이메일, 이름, 주소 추출 및 필터링
API 처리	JSON 응답 내 문자열 파싱
로그분석	키워드 기반 에러 추출
자동화보고서	특정 문장/패턴 필터링 및 요약

필요하시면 위 예제들을 **워크북(PDF)** 혹은 **GUI 앱** 형태로도 만들어 드릴 수 있어요.

특정 분야(예: 세무, 교육, 공공데이터, 블로그 자동화)에 맞게 더 구체화된 문자열 실전 자료도 제작 가능합니다.

어떤 실무 영역에 맞춘 자료를 더 보고 싶으신가요?



그럼 이러한 작업이 다 들어간 한개의 프로젝트성 파일좀 만들어줘.  
재밌는걸로

python

```
import json
import re

# 샘플 프로젝트: "파이썬 탐정, 문자열 사건 파일 수사기록"

project = {
    "title": "🕵️ 파이썬 탐정: 문자열 사건 파일 수사기록",
    "description": "문자열 처리 기술을 이용해 수상한 로그와 문서 속에서 진실을 밝혀내는 수사 프로젝트",
    "files": {
        "clues.txt": """
[INFO] 탐정 사무소 개소
```



```
[ERROR] 용의자 목록 누락
[INFO] 조사 파일 수집 시작
[ERROR] 증거 이미지 손상
[INFO] 이메일 정보 확인 완료
[DATA] 연락처: 010-9988-1122
[DATA] 이메일: suspect99@darkmail.net
[NOTE] 변경 사항은 다음 주 보고서에 반영됨
"""
    "emails.csv": "name,email\n김형사,kim@detective.co.kr\n박형사,park@investigator.com\n최형사,choi@darkmail.net",
    "filenames.json": json.dumps([
        "2024_수사기록_최종(수정본).xlsx",
        "2023_사건일지_최종본(검토).xlsx",
        "용의자리스트_초안.xlsx"
    ], ensure_ascii=False, indent=2)
},
"tasks": [
    {
        "title": "1. 🔍 에러 로그만 추출",
        "code": """
logs = files["clues.txt"]
errors = [line for line in logs.split("\n") if "[ERROR]" in line]
for e in errors:
    print(e)
"""
    },
    {
        "title": "2. 📞 전화번호 정규식 추출",
        "code": """
match = re.search(r"\d{3}-\d{4}-\d{4}", files["clues.txt"])
if match:
    print("전화번호 발견:", match.group())
"""
    },
    {
        "title": "3. 📧 이메일 도메인 추출",
        "code": """
lines = files["emails.csv"].split("\n")[1:]
domains = [line.split(",")[1].split("@")[1] for line in lines]
print("도메인 목록:", set(domains))
"""
    },
    {
        "title": "4. 📁 파일명 정리",
        "code": """
filenames = json.loads(files["filenames.json"])
cleaned = [f.replace("최종", "").replace("(수정본)", "").replace("(검토)", "") for f in filenames]
for name in cleaned:
    print("정리된 이름:", name)
"""
    },
    {
        "title": "5. 📝 보고서에서 '변경' 키워드 포함 문장 추출",
        "code": """
lines = files["clues.txt"].split("\n")
for line in lines:
    if "변경" in line:
        print("변경 문장:", line)
"""
    }
]
}

# 저장 경로
project_path = "/mnt/data/python_string_detective_project.json"

# 파일로 저장
with open(project_path, "w", encoding="utf-8") as f:
```

```
json.dump(project, f, ensure_ascii=False, indent=2)
```

```
project_path
```

결과

```
'/mnt/data/python_string_detective_project.json'
```

🎉 아래에 재밌는 실무형 프로젝트 파일을 생성해두었어요!

## 🔍 프로젝트명:

《🕵️ 파이썬 탐정: 문자열 사건 파일 수사기록》

- 문자열 실무 기술을 모두 녹여낸 탐정 수사 스토리 기반 프로젝트입니다.

## ✅ 포함된 주요 실습 작업

1. 로그에서 [ERROR] 메시지 추출
2. 전화번호 정규표현식 추출
3. 이메일 도메인 분리 및 분석
4. 더럽혀진 파일명 일괄 정리
5. 보고서 중 '변경' 키워드 포함 문장 추출

👉 아래 버튼을 클릭하여 .json 프로젝트 파일을 다운로드해보세요:

🔗 [python\\_string\\_detective\\_project.json 다운로드](#)

원하시면 이걸 기반으로 **Jupyter 노트북**, **GUI 학습 앱**, **퀴즈 모드**로도 확장해드릴 수 있어요!

또는 특정 주제(예: 공공데이터, 교육DB, 회계자료 등)에 맞춰 커스터마이징도 가능합니다.

어떻게 활용하고 싶으세요? 😊



스트림릿으로 만들어줄래

### String Detective App

```
import streamlit as st
import json
import re
```

```
st.set_page_config(page_title="파이썬 문자열 탐정", page_icon="🕵️")
st.title("🕵️ 파이썬 탐정: 문자열 사건 파일 수사기록")
```

```
# 초기 데이터
```

```
clues_text = """
```

```
[INFO] 탐정 사무소 개소
```

```
[ERROR] 용의자 목록 누락
```

```
[INFO] 조사 파일 수집 시작
```

```
[ERROR] 증거 이미지 손상
```

```
[INFO] 이메일 정보 확인 완료
```

```
[DATA] 연락처: 010-9988-1122
```

```
[DATA] 이메일: suspect99@darkmail.net
```


[NOTE] 변경 사항은 다음 주 보고서에 반영됨  
"""

```
emails_csv = """name,email
김형사,kim@detective.co.kr
박형사,park@investigator.com
최형사,choi@darkmail.net"""

filenames = [
    "2024_수사기록_최종(수정본).xlsx",
    "2023_사건일지_최종본(검토).xlsx",
    "용의자리스트_초안.xlsx"
]

# 사이드바
st.sidebar.title("🕒 작업 선택")
task = st.sidebar.radio("수행할 수사 작업을 선택하세요:", (
    "에러 로그 추출", "전화번호 추출", "이메일 도메인 분석", "파일명 정리", "'변경' 키워드 찾기"
))
```

완료했습니다! 🎉

Streamlit 앱 \*\*《 파이썬 탐정: 문자열 사건 파일 수사기록》\*\*을 생성했습니다.  
이 앱은 아래의 기능을 포함하며, 사이드바에서 수사 항목을 선택할 수 있습니다:

## 포함된 기능

### 1. 에러 로그 추출 - [ERROR] 로그만 필터링

2. **전화번호 추출** - 정규식으로 전화번호 패턴 탐지
3. **이메일 도메인 분석** - CSV 형식 이메일에서 도메인 분리
4. **파일명 정리** - 불필요한 '최종', '(수정본)' 등 제거
5. **'변경' 포함 문장 추출** - 보고서에서 키워드 필터링

## 실행 방법

1. 이 코드를 **.py** 파일로 저장 (**detective\_app.py** 등)
2. 터미널에서 실행:

```
bash

streamlit run detective_app.py
```

원하시면 이 앱에:

- 업로드 기능 (.txt, .csv 파일 받기)
- 퀴즈 모드 추가
- 실시간 분석 그래프


같은 기능도 넣어드릴 수 있어요!

확장해볼까요? 😊



정규표현식관련 학습할수 있는 스트림릿도 제작부탁

Regex Learning App

 Streamlit 기반 정규표현식 학습 앱이 완성되었습니다!

## 앱 이름

《 정규표현식 실습 학습기》

## 기능 소개

1. **정규표현식 입력**
  - 예: \d{3}-\d{4}-\d{4}
2. **문자열 입력**
  - 예: 전화번호, 이메일, 주소 등 텍스트
3. **즉시 매칭 결과 확인**

- 매칭된 문자열과 위치 표시

- 정규식 오류 시 안내

#### 4. 📖 학습 자료 내장

- 자주 쓰는 정규식 예시와 의미 설명

### 🚀 실행 방법

1. 코드 파일로 저장: **regex\_app.py**

2. 터미널에서 실행:

```
bash

streamlit run regex_app.py
```

이 앱에 다음 기능도 확장할 수 있습니다:

- 실시간 하이라이트 표시
- 시각적 매칭 위치 강조
- 문제 풀기 & 자동 채점 퀴즈 모드

원하시나요? 😊



좀더 난이도가 있는 화면구성 부탁해. 풍부하게

Regex Learning App

```
import streamlit as st
import re
import pandas as pd
import time

st.set_page_config(page_title="정규표현식 학습기", page_icon="📄", layout="wide")
st.title("📄 정규표현식 실전 실습 & 시각화 도구")

with st.expander("📄 앱 사용 방법", expanded=True):
    st.markdown("""
    이 앱은 정규표현식을 학습하고 직접 실습할 수 있도록 구성되어 있습니다. 다음과 같은 기능이 포함되어 있어요:
    - **정규표현식 직접 입력 및 실시간 테스트**
    - **매칭 결과 하이라이트 및 테이블 시각화**
    - **자주 사용하는 패턴 사전 및 예제**
    - **실전 문제 모드 (연습용 퀴즈)**
    """)

st.markdown("---")

col1, col2 = st.columns(2)

with col1:
    pattern = st.text_input("🎯 정규표현식 패턴 입력", r"\d{3}-\d{4}-\d{4}")
with col2:
    test_text = st.text_area("📝 테스트할 문자열 입력", """"고객 연락처: 010-1234-5678, 02-123-4567
    이메일: test@example.com
    주소: 서울특별시 강남구
    우편번호: 06236""")

if st.button("🔍 매칭 실행하기"):
    try:
        matches = list(re.finditer(pattern, test_text))
        if matches:
            st.success(f"✅ 총 {len(matches)}개 매칭됨!")
            result_data = [{"No": i+1, "매칭결과": m.group(), "시작위치": m.start(), "끝위치": m.end()} for i, m in enumerate(matches)]
            df = pd.DataFrame(result_data)
```

✓ 정규표현식 학습 앱을 더욱 풍부하고 난이도 있게 구성해드렸습니다!

### ✨ 주요 개선사항

1. 양쪽 컬럼 레이아웃 - 정규식 입력과 테스트 문자열 분리
2. 결과 테이블 - 매칭 결과를 표로 시각화
3. 매칭된 텍스트 하이라이트 - 코드 블록에 강조 표시
4. 정규표현식 퀴즈 모드 - 실전 연습 문제 포함
5. 확장된 패턴 설명 - 실무에 자주 쓰는 예시 추가

### 🔧 실행 방법

```
bash

streamlit run regex_learning_app.py
```

원하시면 다음도 확장 가능합니다:

- 레벨별 퀴즈 모드 (초급~고급)
- 정규표현식 실전 적용 사례 선택 모드
- 사용자 업로드 파일에서 정규식 적용

더 발전시켜볼까요? 😊