

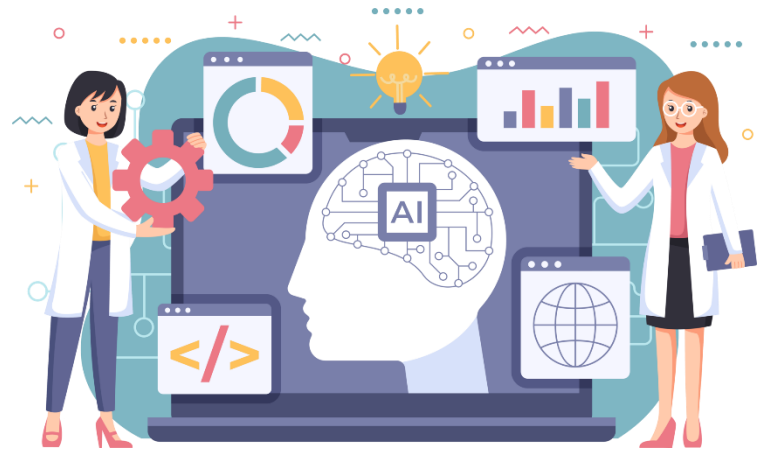


기술 통계 / Fast API

인공지능 RPA – 4주차

# 학습 내용

1. 기술통계 구하기
2. 함수
3. Fast API 기초



---

# 인공지능 이론 및 실습

---

## \_데이터프레임 실습

```
data = {'이름' : ['Kim', 'Park', 'Lee', 'Ho'],  
        '국어' : [90, 58, 88, 100],  
        '영어' : [100, 60, 80, 70],  
        '수학' : [55, 65, 76, 88] }
```

	이름	국어	영어	수학
0	Kim	90	100	55
1	Park	58	60	65
2	Lee	88	80	76
3	Ho	100	70	88

1. data를 데이터 프레임으로 만드시오
2. 1에서 만든 데이터 프레임을 출력하시오
3. 학생 이름만 추출해서 출력하시오 (열 추출)
4. 'Park' 성적만 출력하시오
5. 'Ho' 학생의 수학점수를 90점으로 수정하시오
6. 'Oh' 학생의 국어(100), 영어(70), 수학(80) 성적을 새로 추가하시오
7. 'Lee' 학생의 성적을 삭제하시오.



## \_데이터프레임 실습

1. data를 데이터 프레임으로 만드시오

	이름	국어	영어	수학
0	Kim	90	100	55
1	Park	58	60	65
2	Lee	88	80	76
3	Ho	100	70	88

2. 1에서 만든 데이터 프레임을 출력하시오

```
print(df, end="\n\n")
```

3. 학생 이름만 추출해서 출력하시오 (열 추출)

```
sr_name = df['이름']
```

```
print(sr_name, end="\n\n")
```



## \_데이터프레임 실습

### 4. 'Park' 성적만 출력하시오

```
park_data = df.loc[1]
```

```
park_data = df.loc[df['이름'] == 'Park']  
print(park_data, end="\n\n")
```

	이름	국어	영어	수학
0	Kim	90	100	55
1	Park	58	60	65
2	Lee	88	80	76
3	Ho	100	70	88

### 5. 'Ho' 학생의 수학점수를 90점으로 수정하시오

```
df.loc[df['이름'] == 'Ho', '수학'] = 90  
print(df, end="\n\n")
```

### 6. 'Oh' 학생의 국어(100), 영어(70), 수학(80) 성적을 새로 추가하시오

```
df.loc[3] = ['Oh', 100, 70, 80]  
print(df, end="\n\n")
```



## \_기술통계 구하기

- describe( ) : 기술통계를 자동으로 추출
  - count: 누락된 값을 제외한 데이터 개수
  - mean: 평균
  - std: 표준편차
  - min: 최솟값
  - 50%: 중앙값.
  - 25%와 75%: 순서대로 늘어 놓았을 때 25% 지점과 75% 지점에 놓인 값
  - max: 최댓값

	이름	국어	영어	수학
0	Kim	90	100	55
1	Park	58	60	65
2	Lee	88	80	76
3	Ho	100	70	88

	국어	영어	수학
count	4.00000	4.000000	4.00000
mean	84.00000	77.500000	71.00000
std	18.11077	17.078251	14.21267
min	58.00000	60.000000	55.00000
25%	80.50000	67.500000	62.50000
50%	89.00000	75.000000	70.50000
75%	92.50000	85.000000	79.00000
max	100.00000	100.000000	88.00000



## \_기술통계 구하기

### ■ 평균 : mean()

df[<열 범위>].mean()

### ■ 중앙값 : median()

df[<열 범위>].median()

### ■ 최대값, 최소값

df[<열 범위>].max()      df[<열 범위>].min()

### ■ 백분위수/사분위수 찾기 : %에 위치한 값 찾기

df[<열 범위>].quantile([0.25,0.5,0.75])

### ■ 분산/ 표준편차

df[<열 범위>].var()      df[<열 범위>].std()

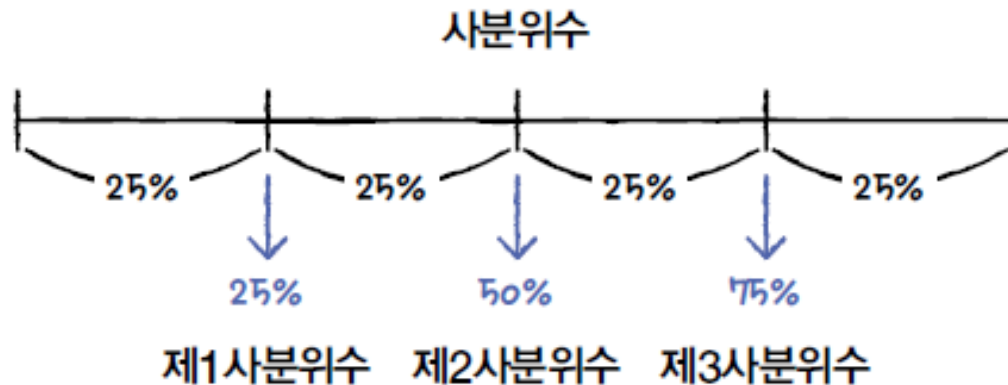




## \_기술통계 구하기

### 분위수

- 사분위수(quartile)는 순서대로 정렬된 데이터를 네 구간으로 나눔
  - 사분위수는 3개가 나오고 각각 25%, 50%, 75%에 해당
  - 제1사분위수 - 25%에 해당하는 값
  - 제2사분위수 - 중앙값
  - 제3사분위수 - 75%에 해당하는 값



# \_파이썬 함수

## 함수

- 프로그래밍에서 함수(Function)는 특정 기능을 하는 로직을 이름을 붙여 묶은 것
- 이름을 불러주어야 실행된다 (호출)

## 함수 호출 (사용)

- 함수이름 (전달데이터)  
ex) func(100)
- **함수를 호출하려면 함수가 있어야 함**
  - 함수가 정의되어 있는 가?
  - 함수를 담은 라이브러리를 import 했는 가?



# \_파이썬 함수

## 함수 정의

```
def 함수이름(형식인수1, 형식인수2, ... ) :  
    ← 문장1  
    문장2  
    ...  
    문장n (return문 포함)
```

함수코드

- 함수 이름 뒤에 반드시 ‘:’
- 함수정의는 그 자체로는 실행되지 않음. **호출해야 실행됨**
- **리턴값** : 함수가 넘겨주는 값
  - 함수가 종료될 때 호출부분으로 반환하는 값
  - 함수코드 내에서 **return 문**으로 함수를 종료하고 값을 반환함

```
return (값 또는 수식) # 반환값이 있는 경우  
return                # 반환값이 없는 경우
```



## 파이썬 함수

```
def sum_int(a, b):  
    return a+b
```

```
num1 = 10  
num2 = 20
```

```
result = sum_int(num1, num2)  
print(f'{num1} + {num2} = {result}')
```

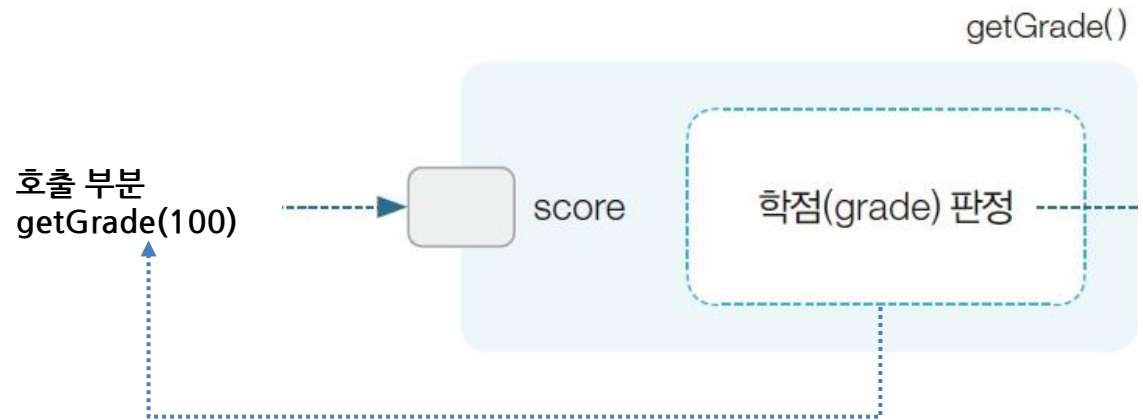


# \_파이썬 함수

## 함수 인수의 특성

- 함수의 인수는 호출부분에서 전달하는 데이터를 받는다
- 함수의 인수는 함수내부에서 사용하는 일종의 지역변수다

```
def getGrade(score) :  
    if score >= 90 : grade = 'A'  
    elif score >= 80 : grade = 'B'  
    elif score >= 70 : grade = 'C'  
    elif score >= 60 : grade = 'D'  
    else : grade = 'F'  
    return (grade)
```



---

# RPA 실습

---

# FastAPI

Rest API를 만들기 위한 Python 웹 프레임워크

<https://fastapi.tiangolo.com/ko/>



*FastAPI 프레임워크, 고성능, 간편한 학습, 빠른 코드 작성, 준비된 프로덕션*

Test **passing** coverage **100%** pypi package **v0.111.1**



# \_FastAPI

## 1. Fast API 설치

- 라이브러리 설치 (처음 1회)

```
pip install fastapi
```

```
pip install "uvicorn"
```

- 모듈 import (파일 마다)

```
from fastapi import FastAPI
```

## 2. 코드 입력

```
fapi1.py
```

## 3. 실행

```
uvicorn fapi1:app --reload
```

## 4. 브라우저에서 확인

```
http://127.0.0.1:8000
```

```
from fastapi import FastAPI
```

```
app = FastAPI()
```

```
1 @app.get("/")
```

```
2 def read_root():  
    return {"Hello": "World"}
```





## \_데이터 형식 : JSON과 CSV

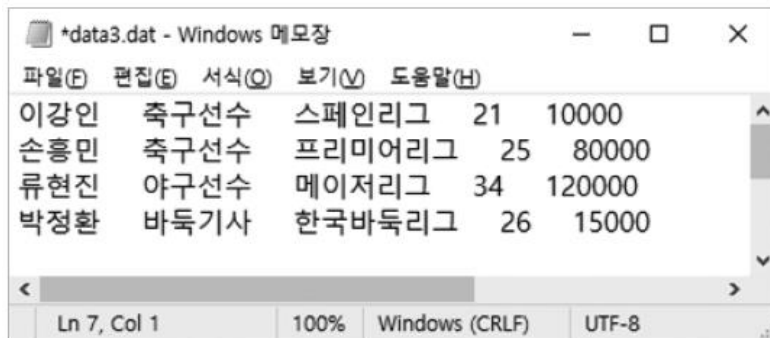
JSON(JavaScript Object Notation) : 'key : value' 쌍으로 구성된 텍스트 데이터

→ 모든 키는 큰 따옴표로 감싸야 한다

```
{  
  "name": "혼자 공부하는 데이터 분석",  
  "author": ["박해선", "홍길동"],  
  "year": 2022  
}
```

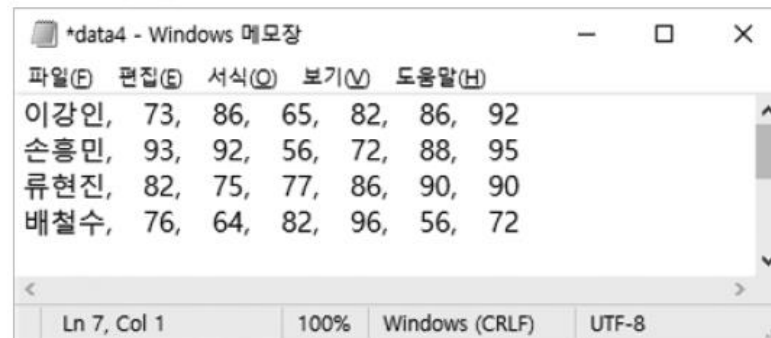
CSV(Comma-Separated Values) : coma(,)로 구분된 텍스트 데이터

- 한 줄이 하나의 레코드(record)이며 레코드는 콤마로 구분된 여러 필드(field)로 구성



이강인	축구선수	스페인리그	21	10000
손흥민	축구선수	프리미어리그	25	80000
류현진	야구선수	메이저리그	34	120000
박정환	바둑기사	한국바둑리그	26	15000

(a) 공백으로 구분



이강인,	73,	86,	65,	82,	86,	92
손흥민,	93,	92,	56,	72,	88,	95
류현진,	82,	75,	77,	86,	90,	90
배철수,	76,	64,	82,	96,	56,	72

(b) 콤마로 구분(CSV 파일)



# \_FastAPI

## 파라미터 받기

`http://127.0.0.1:8000/item?item_id=20000&name=홍길동&age=20`

- 함수의 인자를 통해서 파라미터를 받음 (타입 힌트를 이용)

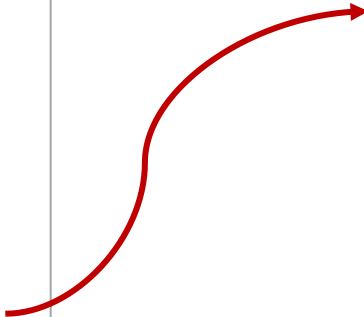
이름 : 형식 = 디폴트값

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/item")
def read_item(item_id: int, name: str = None, age: int = 0):
    return {"item_id": item_id, "name": name, "age": age}
```



- int (정수)
- float (실수)
- bool (불리언)
- str (문자열)
- None (null, void)
- List (리스트)
- Dict (딕셔너리)

