Python 프로그래밍 2주차

유용한 내장 함수와 모듈 활용

📋 오늘의 학습 목표

이론

- **유용한 내장 함수들** 마스터하기
- f-string **포맷팅** 심화 학습
- **자주 쓰는 표준 라이브러리** 활용법

실습

- **실생활 문제 해결** 프로그램 만들기
- 성적 처리, 로또 번호 생성기, 파일 정리 자동화

🔢 유용한 내장 함수들

1. enumerate() - 인덱스와 값을 함께!

Before (개선 전)

```
fruits = ['apple', 'banana', 'cherry']
for i in range(len(fruits)):
    print(f"{i}: {fruits[i]}")
```

After (개선 후)

```
fruits = ['apple', 'banana', 'cherry']
for i, fruit in enumerate(fruits):
    print(f"{i}: {fruit}")
```

🔢 enumerate() 심화 활용

```
# 시작 번호 지정
students = ['Alice', 'Bob', 'Charlie']
for num, student in enumerate(students, 1):
    print(f"학번 {num}: {student}")
# 출력:
# 학번 1: Alice
# 학번 2: Bob
# 학번 3: Charlie

# 조건부 처리
scores = [85, 90, 78, 92, 88]
for i, score in enumerate(scores):
    if score >= 90:
        print(f"{i+1}번째 학생: 우수 ({score}점)")
```

🤐 zip() - 여러 리스트를 묶어서!

```
names = ['Alice', 'Bob', 'Charlie']
ages = [25, 30, 35]
cities = ['Seoul', 'Busan', 'Incheon']

# 세 리스트를 동시에 순회
for name, age, city in zip(names, ages, cities):
    print(f"{name}({age}세)는 {city}에 살아요")

# 출력:
# Alice(25세)는 Seoul에 살아요
# Bob(30세)는 Busan에 살아요
# Charlie(35세)는 Incheon에 살아요
```

🤐 zip() 실용 예제

```
# 딕셔너리 만들기
keys = ['name', 'age', 'city']
values = ['Alice', 25, 'Seoul']
person = dict(zip(keys, values))
print(person)  # {'name': 'Alice', 'age': 25, 'city': 'Seoul'}

# 리스트 쌍 만들기
x_coords = [1, 2, 3, 4]
y_coords = [10, 20, 30, 40]
points = list(zip(x_coords, y_coords))
print(points)  # [(1, 10), (2, 20), (3, 30), (4, 40)]

# 길이가 다른 리스트는?
a = [1, 2, 3, 4, 5]
b = ['a', 'b', 'c']
result = list(zip(a, b))
print(result)  # [(1, 'a'), (2, 'b'), (3, 'c')] - 짧은 쪽에 맞춤
```

🗺️ map() - 모든 요소에 함수 적용!

```python
# 문자열 숫자들을 정수로 변환
str_numbers = ['1', '2', '3', '4', '5']
numbers = list(map(int, str_numbers))
print(numbers)  # [1, 2, 3, 4, 5]

# 모든 요소를 제곱
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, numbers))
print(squared)  # [1, 4, 9, 16, 25]

# 여러 개의 iterable
def add(x, y):
    return x + y

a = [1, 2, 3]
b = [10, 20, 30]
result = list(map(add, a, b))
print(result)  # [11, 22, 33]
```

🔍 filter() - 조건에 맞는 요소만!

```
# 짝수만 필터링
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)  # [2, 4, 6, 8, 10]

# 빈 문자열 제거
words = ['hello', '', 'world', '', 'python']
non_empty = list(filter(lambda x: x != '', words))
print(non_empty)  # ['hello', 'world', 'python']

# 조건 함수 정의
def is_passing(score):
    return score >= 60

scores = [85, 45, 90, 55, 78, 92]
passing_scores = list(filter(is_passing, scores))
print(passing_scores)  # [85, 90, 78, 92]
```

📊 sorted()의 key 파라미터 활용

기본 정렬

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6]
print(sorted(numbers))  # [1, 1, 2, 3, 4, 5, 6, 9]
print(sorted(numbers, reverse=True))  # [9, 6, 5, 4, 3, 2, 1, 1]
```

key 파라미터로 정렬 기준 설정

```
# 절댓값으로 정렬
numbers = [-3, 1, -4, 2, 5]
print(sorted(numbers, key=abs))  # [1, 2, -3, -4, 5]

# 문자열 길이로 정렬
words = ['python', 'java', 'c', 'javascript']
print(sorted(words, key=len))  # ['c', 'java', 'python', 'javascript']
```

📊 sorted() 심화 활용

```
# 학생 정보를 성적으로 정렬
students = [
    {'name': 'Alice', 'score': 85},
    {'name': 'Bob', 'score': 90},
    {'name': 'Charlie', 'score': 78}
]

# 성적 기준 정렬
by_score = sorted(students, key=lambda x: x['score'], reverse=True)
print(by_score)
# [{'name': 'Bob', 'score': 90}, {'name': 'Alice', 'score': 85}, {'name': 'Charlie', 'score': 78}]

# 이름 기준 정렬
by_name = sorted(students, key=lambda x: x['name'])
print(by_name)
# [{'name': 'Alice', 'score': 85}, {'name': 'Bob', 'score': 90}, {'name': 'Charlie', 'score': 78}]
```

🎨 f-string 포맷팅 심화

기본 사용법 복습

```
name = "Alice"
age = 25
print(f"이름: {name}, 나이: {age}")  # 이름: Alice, 나이: 25
```

숫자 포맷팅

```
price = 1234567.89
print(f"가격: {price:,}원")          # 가격: 1,234,567.89원
print(f"가격: {price:.0f}원")        # 가격: 1234568원
print(f"가격: {price:,.0f}원")       # 가격: 1,234,568원

# 퍼센트 표시
rate = 0.1234
print(f"할인율: {rate:.1%}")         # 할인율: 12.3%
```

🎨 f-string 심화 기능

정렬과 패딩

```
items = ['Apple', 'Banana', 'Cherry']
for item in items:
    print(f"|{item:<10}|")  # 왼쪽 정렬
    print(f"|{item:>10}|")  # 오른쪽 정렬
    print(f"|{item:^10}|")  # 가운데 정렬

# 출력:
# |Apple     |
# |     Apple|
# |  Apple   |
```

날짜 포맷팅

```
from datetime import datetime

now = datetime.now()
print(f"현재 시간: {now:%Y-%m-%d %H:%M:%S}")
print(f"오늘 날짜: {now:%Y년 %m월 %d일}")
```

📅 datetime으로 날짜/시간 다루기

기본 사용법

```
from datetime import datetime, date, timedelta

# 현재 시간
now = datetime.now()
print(f"현재: {now}")

# 특정 날짜 생성
birthday = datetime(1995, 5, 15)
print(f"생일: {birthday}")

# 오늘 날짜
today = date.today()
print(f"오늘: {today}")
```

datetime 심화 활용

날짜 계산

```python
from datetime import datetime, timedelta

# 7일 후
now = datetime.now()
after_week = now + timedelta(days=7)
print(f"일주일 후: {after_week}")

# 나이 계산
def calculate_age(birth_date):
    today = date.today()
    age = today.year - birth_date.year
    if today.month < birth_date.month or \
       (today.month == birth_date.month and today.day < birth_date.day):
        age -= 1
    return age

birthday = date(1995, 5, 15)
age = calculate_age(birthday)
print(f"나이: {age}세")
```

문자열과 datetime 변환

```python
from datetime import datetime

# 문자열 → datetime
date_str = "2024-03-15"
date_obj = datetime.strptime(date_str, "%Y-%m-%d")
print(date_obj)

# datetime → 문자열
now = datetime.now()
formatted = now.strftime("%Y년 %m월 %d일 %H시 %M분")
print(formatted)

# 다양한 포맷
formats = [
    "%Y-%m-%d",           # 2024-03-15
    "%Y/%m/%d",           # 2024/03/15
    "%Y년 %m월 %d일",      # 2024년 03월 15일
    "%Y-%m-%d %H:%M:%S"   # 2024-03-15 14:30:25
]
```

🎲 random 모듈 활용

기본 난수 생성

```
import random

# 0.0 이상 1.0 미만의 실수
print(random.random())

# 1 이상 10 이하의 정수
print(random.randint(1, 10))

# 1 이상 10 미만의 정수
print(random.randrange(1, 10))

# 리스트에서 무작위 선택
fruits = ['apple', 'banana', 'cherry']
print(random.choice(fruits))
```

🎲 random 심화 활용

```python
import random

# 여러 개 무작위 선택 (중복 허용)
colors = ['red', 'blue', 'green', 'yellow']
selected = random.choices(colors, k=3)
print(selected)

# 여러 개 무작위 선택 (중복 없음)
numbers = list(range(1, 46))  # 1~45
lotto = random.sample(numbers, 6)
print(sorted(lotto))

# 리스트 섞기
deck = ['A', 'K', 'Q', 'J'] + list(range(10, 0, -1))
random.shuffle(deck)
print(deck)

# 시드 설정 (재현 가능한 난수)
random.seed(42)
print(random.randint(1, 100))  # 항상 같은 결과
```

💻 os, sys 기초

os 모듈 - 운영체제 관련

```
import os

# 현재 작업 디렉토리
print(os.getcwd())

# 디렉토리 목록
print(os.listdir('.'))

# 파일/디렉토리 존재 확인
print(os.path.exists('test.txt'))

# 디렉토리 생성
os.makedirs('new_folder', exist_ok=True)

# 파일 경로 조작
path = os.path.join('folder', 'subfolder', 'file.txt')
```

💻 sys 모듈 - 시스템 관련

```python
import sys

# Python 버전 정보
print(sys.version)

# 명령행 인수
print(sys.argv)  # 스크립트 실행 시 전달된 인수들

# 모듈 검색 경로
print(sys.path)

# 프로그램 종료
# sys.exit()  # 프로그램 종료 (주의해서 사용!)

# 플랫폼 정보
print(sys.platform)  # 'win32', 'darwin', 'linux' 등
```

🏆 실습 1: 성적 처리 프로그램

요구사항

1. 학생 이름과 점수를 입력받기
2. 평균, 최고점, 최저점 계산
3. 등급 매기기 (A: 90~100, B: 80~89, C: 70~79, D: 60~69, F: 0~59)
4. 결과를 파일로 저장

```
def grade_calculator():
    """성적 처리 프로그램"""
    students = []

    # 학생 정보 입력
    while True:
        name = input("학생 이름 (종료: q): ")
        if name.lower() == 'q':
            break

        try:
            score = int(input("점수: "))
            students.append({'name': name, 'score': score})
        except ValueError:
            print("올바른 숫자를 입력하세요.")

    # 계속...
```

🏆 실습 1: 성적 처리 프로그램

```python
def get_grade(score):
    """점수에 따른 등급 반환"""
    if score >= 90:
        return 'A'
    elif score >= 80:
        return 'B'
    elif score >= 70:
        return 'C'
    elif score >= 60:
        return 'D'
    else:
        return 'F'

def analyze_scores(students):
    """성적 분석"""
    if not students:
        return None

    scores = [s['score'] for s in students]

    return {
        'average': sum(scores) / len(scores),
        'max_score': max(scores),
        'min_score': min(scores),
        'total_students': len(students)
    }
```

🎰 실습 2: 로또 번호 생성기

요구사항

1. 1~45 중에서 6개 번호 무작위 선택
2. 중복 없이 정렬된 결과 출력
3. 여러 게임 생성 가능
4. 과거 당첨 번호와 비교 기능

```python
import random
from datetime import datetime

def generate_lotto_numbers():
    """로또 번호 6개를 생성합니다."""
    numbers = random.sample(range(1, 46), 6)
    return sorted(numbers)

def generate_multiple_games(count):
    """여러 게임의 로또 번호를 생성합니다."""
    games = []
    for i in range(count):
        numbers = generate_lotto_numbers()
        games.append(f"게임 {i+1}: {numbers}")
    return games
```

🎰 실습 2: 로또 번호 생성기

```
def check_winning(my_numbers, winning_numbers):
    """당첨 번호와 내 번호를 비교합니다."""
    matches = set(my_numbers) & set(winning_numbers)
    match_count = len(matches)

    prizes = {
        6: "1등 (잭팟!)",
        5: "3등",
        4: "4등",
        3: "5등"
    }

    if match_count in prizes:
        return f"{match_count}개 일치 - {prizes[match_count]}"
    else:
        return f"{match_count}개 일치 - 낙첨"

def save_numbers_to_file(games):
    """생성된 번호를 파일로 저장합니다."""
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"lotto_numbers_{timestamp}.txt"

    with open(filename, 'w', encoding='utf-8') as f:
        f.write(f"로또 번호 생성 일시: {datetime.now()}\n")
        f.write("=" * 40 + "\n")
        for game in games:
            f.write(game + "\n")

    print(f"번호가 {filename}에 저장되었습니다.")
```

📁 실습 3: 파일 정리 자동화 스크립트

요구사항

1. 지정된 폴더의 파일들을 확장자별로 분류
2. 새 폴더 생성하여 파일 이동
3. 처리 결과 로그 출력

```python
import os
import shutil
from datetime import datetime

def organize_files(source_folder):
    """파일을 확장자별로 정리합니다."""

    # 확장자별 폴더 매핑
    file_types = {
        'images': ['.jpg', '.jpeg', '.png', '.gif', '.bmp'],
        'documents': ['.pdf', '.doc', '.docx', '.txt', '.xlsx'],
        'videos': ['.mp4', '.avi', '.mkv', '.mov'],
        'audio': ['.mp3', '.wav', '.flac'],
        'archives': ['.zip', '.rar', '.7z']
    }

    # 정리 결과 저장
    results = {'moved': 0, 'created_folders': [], 'errors': []}

    # 계속...
```

📁 실습 3: 파일 정리 자동화

```
def get_file_category(filename, file_types):
    """파일의 카테고리를 반환합니다."""
    _, ext = os.path.splitext(filename.lower())

    for category, extensions in file_types.items():
        if ext in extensions:
            return category

    return 'others'  # 기타 파일

def move_file_safely(source, destination):
    """파일을 안전하게 이동합니다."""
    try:
        # 대상 폴더가 없으면 생성
        os.makedirs(os.path.dirname(destination), exist_ok=True)

        # 같은 이름의 파일이 있으면 번호 추가
        if os.path.exists(destination):
            base, ext = os.path.splitext(destination)
            counter = 1
            while os.path.exists(f"{base}_{counter}{ext}"):
                counter += 1
            destination = f"{base}_{counter}{ext}"

        shutil.move(source, destination)
        return True, destination
    except Exception as e:
        return False, str(e)
```

📊 실습 결과 종합

각 실습에서 사용한 개념들

성적 처리 프로그램

- `enumerate()` : 순번과 함께 학생 정보 처리
- `map()` : 점수를 등급으로 변환
- `sorted()` : 성적순 정렬
- f-string: 결과 포맷팅

로또 번호 생성기

- `random.sample()` : 중복 없는 번호 선택
- `datetime` : 타임스탬프 생성
- 파일 입출력: 결과 저장

파일 정리 자동화

- `os` 모듈: 파일 시스템 조작
- `enumerate()` : 진행 상황 표시
- 예외 처리: 안전한 파일 이동

🎯 2주차 정리

오늘 배운 내장 함수들

- **enumerate()**: 인덱스와 값을 함께
- **zip()**: 여러 리스트를 묶어서
- **map()**: 모든 요소에 함수 적용
- **filter()**: 조건에 맞는 요소만
- **sorted()**: 다양한 기준으로 정렬

강력한 도구들

- **f-string 심화**: 숫자 포맷팅, 정렬
- **datetime**: 날짜/시간 처리
- **random**: 난수 생성
- **os/sys**: 시스템 조작

🎯 과제

개인 프로젝트

1. **성적 처리 프로그램** 완성하기
2. **로또 번호 생성기**에 통계 기능 추가
3. **파일 정리 스크립트**를 자신의 폴더에 적용

추가 학습

- Python 공식 문서의 내장 함수 섹션 읽어보기
- `collections` 모듈 알아보기 (Counter, defaultdict)

다음 주도 열심히 함께 배워봅시다! 🚀