# Python 프로그래밍 1주차

## OT 및 복습 + 코딩 스타일

# 👋 자기소개 & 학습 목표

## 자기소개 시간

- 이름, 프로그래밍 경험, Python을 배우는 이유
- 현재 Python 실력 수준은?

## 학습 목표 공유

- 각자의 Python 학습 목표는?
- 이 과정을 통해 얻고 싶은 것은?

# 📝 쪽지시험 리뷰 시작!

총 17문제를 함께 풀어보며

**헷갈리기 쉬운 개념들을 확실히 정리**해봅시다!

# 문제 1: 리스트 인덱싱

```
my_list = [1, 2, 3, 4, 5]
print(my_list[2])
```

**정답**: D. 3

## 🔍 해설

- Python 인덱스는 **0부터 시작**
- `my_list[2]`는 **세 번째 요소**를 의미
- 인덱스: 0→1, 1→2, 2→3, 3→4, 4→5

# 문제 2: 타입 변환과 type()

```
a = '123456789'
b = type(a)
c = type(int(a))
print(c)
```

**정답**: C. <class 'int'>

## 🔍 해설

- `a`는 문자열 → `type(a)`는 `<class 'str'>`
- `int(a)`는 정수로 변환 → `type(int(a))`는 `<class 'int'>`

# 문제 3: 리스트 슬라이싱 ⭐

```
info = ['머용', '010-1234-1010', '서울 강서구', '오즈코딩스쿨']
```

이름과 전화번호만 출력하려면?

**정답**: B. print(info[0:2])

## 🔍 해설

- `info[0:2]` → 인덱스 0부터 2 전까지 (0, 1)
- 결과: `['머용', '010-1234-1010']`

# 슬라이싱 완벽 이해! 🎯

```
my_list = [0, 1, 2, 3, 4, 5]
```

| 슬라이싱 | 결과 | 설명 |
|---|---|---|
| `my_list[:3]` | `[0, 1, 2]` | 처음부터 3 전까지 |
| `my_list[2:]` | `[2, 3, 4, 5]` | 2부터 끝까지 |
| `my_list[1:4]` | `[1, 2, 3]` | 1부터 4 전까지 |
| `my_list[::2]` | `[0, 2, 4]` | 2칸씩 건너뛰기 |
| `my_list[::-1]` | `[5, 4, 3, 2, 1, 0]` | 역순 |

# 문제 4: 리스트 내장 메서드

**정답: E. display()**

## 🔍 해설

실제 리스트 메서드들:

- `append()` : 끝에 요소 추가
- `pop()` : 요소 제거 후 반환
- `remove()` : 특정 값 제거
- `insert()` : 특정 위치에 삽입
- `display()` 는 **존재하지 않는 메서드**

# 문제 5: 조건문

```
x = 10
if x > 5:
    print("크다")
else:
    print("작다")
```

**정답: A. 크다**

## 🔍 해설

- `x = 10`이고 `10 > 5`는 참
- 따라서 "크다" 출력

# 문제 6: in 연산자와 논리 연산자

```
name_list = ['이용대', '오인용', '구자철', '박지성', '김연아']
print('용' in name_list and '왕' in name_list)
```

**정답**: B. False

## 🔍 해설

- `'용' in name_list` : False (문자열 '용'이 리스트에 없음)
- `'왕' in name_list` : False
- `False and False` = **False**

# 문제 7: while 반복문

```
a = 1
[빈칸] (a < 11):
    print(a)
    a += 1
```

**정답**: A. while

## 🔍 해설

- 조건이 참인 동안 반복하는 것은 `while` 문
- 1부터 10까지 출력하는 코드

# 문제 8: for 문의 특징

**정답: D. for 문은 항상 숫자 0에서 시작하여 지정된 숫자까지 반복합니다.**

## 🔍 해설

틀린 이유:

```
for name in ['Alice', 'Bob']:  # 문자열 리스트 순회
    print(name)

for i in range(5, 10):  # 5부터 9까지
    print(i)
```

for문은 0부터 시작할 필요가 없습니다!

# 문제 9: 함수 정의

**정답: B. def myFunction()**

## 🔍 해설

```
# 올바른 함수 정의
def myFunction():
    pass

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

함수 정의는 `def 함수명():` 형태!

# 문제 10: 함수 매개변수

**정답: D. 함수의 매개변수의 데이터 타입으로 클래스를 사용할 수 있다.**

## 🔍 해설

```
class Person:
    def __init__(self, name):
        self.name = name

def greet(person: Person):  # 클래스를 매개변수로 사용
    print(f"Hello, {person.name}")

p = Person("Alice")
greet(p)
```

# 함수와 메서드의 차이 🤔

## 함수 (Function)

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

result = add(3, 5)  # 독립적으로 호출
```

## 메서드 (Method)

```
my_list = [1, 2, 3]
my_list.append(4)  # 객체에 속한 함수

"hello".upper()  # 문자열 객체의 메서드
```

**메서드는 특정 객체에 속한 함수입니다!**

# 문제 11: 클래스 선언

**정답**: A. class MyClass:

## 🔍 해설

```
# 올바른 클래스 정의
class MyClass:
    def __init__(self):
        pass

# 인스턴스 생성
obj = MyClass()
```

# 문제 12: 클래스 키워드들

1. **self** - 인스턴스 자신을 참조
2. **@classmethod** - 클래스 메서드 데코레이터
3. **init** - 초기화 메서드
4. **@property** - 프로퍼티 데코레이터

**정답: A. self, @classmethod, init, @property**

# 클래스 기초 개념 복습 📚

```python
class Rectangle:
    def __init__(self, width, height):  # 초기화 메서드
        self.width = width    # self: 인스턴스 자신
        self.height = height

    @property
    def area(self):  # 프로퍼티: 메서드를 속성처럼 사용
        return self.width * self.height

    @classmethod
    def square(cls, side):  # 클래스 메서드
        return cls(side, side)

# 사용법
rect = Rectangle(3, 4)
print(rect.area)  # 12 (메서드지만 속성처럼 접근)
square = Rectangle.square(5)  # 클래스 메서드로 정사각형 생성
```

# 문제 13: math 모듈

**정답**: A. sqrt()

```
import math

result = math.sqrt(16)  # 4.0
```

# 문제 14: 파일 입출력

**정답: B. open() 함수를 사용하여 파일을 열 때, 반드시 close() 메서드를 호출하여 파일 핸들을 닫아야 합니다.**

## 🔍 해설

```
# with문을 사용하면 자동으로 닫힘
with open('file.txt', 'r') as f:
    content = f.read()
# 여기서 자동으로 파일이 닫힘
```

# 문제 15: 예외 처리

**정답**: A. try-except

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("0으로 나눌 수 없습니다!")
```

# 문제 16: 예외 발생 키워드

**정답**: raise

```
def divide(a, b):
    if b == 0:
        raise ValueError("0으로 나눌 수 없습니다!")
    return a / b
```

# 문제 17: 클래스 상속

```
class A:
    def __init__(self, x):
        self.x = x

class B(A):
    def __init__(self, x, y):
        super().__init__(x)  # 부모 클래스 초기화
        self.y = y

class C(B):
    def __init__(self, x, y, z):
        super().__init__(x, y)  # 부모 클래스 초기화
        self.z = z

obj = C(1, 2, 3)
print(obj.x, obj.y, obj.z)  # 1 2 3
```

**정답: A. 1 2 3**

# 🎯 자주 틀리는 개념 정리

1. **인덱스는 0부터!**
2. **슬라이싱: [시작:끝:간격]**
3. **함수 vs 메서드 구분하기**
4. **클래스에서 self의 역할**
5. **상속에서 super() 사용법**

# 💡 새로운 내용: PEP 8 스타일 가이드

## PEP 8이란?

- **Python Enhancement Proposal 8**
- Python 코드 작성 스타일 가이드
- 가독성 있는 코드를 위한 규칙

# 📏 PEP 8 주요 규칙

## 1. 들여쓰기

```
# Good
if True:
    print("Hello")

# Bad
if True:
        print("Hello")  # 너무 많은 공백
```

## 2. 한 줄 길이

- **최대 79자**
- 긴 줄은 백슬래시()나 괄호로 나누기

# 📏 PEP 8 주요 규칙

## 3. 빈 줄

```
# 클래스와 함수 사이: 2줄
class MyClass:
    pass


def my_function():
    pass


# 메서드 사이: 1줄
class MyClass:
    def method1(self):
        pass

    def method2(self):
        pass
```

# 🏷️ 변수명, 함수명 잘 짓는 법

## 1. snake_case 사용

```
# Good
user_name = "Alice"
total_price = 1000

def calculate_total_price():
    pass

# Bad
userName = "Alice"
TotalPrice = 1000

def CalculateTotalPrice():
    pass
```

# 🏷️ 변수명, 함수명 잘 짓는 법

## 2. 의미 있는 이름 사용

```
# Good
students = ["Alice", "Bob", "Charlie"]
for student in students:
    print(student)

# Bad
s = ["Alice", "Bob", "Charlie"]
for x in s:
    print(x)
```

## 3. 클래스는 PascalCase

```
class StudentManager:
    pass

class DatabaseConnection:
    pass
```

# 📝 주석과 docstring 작성법

## 1. 주석 (#)

```
# 사용자의 나이를 계산하는 함수
def calculate_age(birth_year):
    current_year = 2024  # 현재 년도
    return current_year - birth_year
```

## 2. Docstring (""")

```
def calculate_bmi(weight, height):
    """
    BMI를 계산하는 함수

    Args:
        weight (float): 몸무게 (kg)
        height (float): 키 (m)

    Returns:
        float: BMI 값
    """
    return weight / (height ** 2)
```

# 🐛 간단한 디버깅 기법

## 1. Print 디버깅

```
def calculate_average(numbers):
    print(f"Input: {numbers}")  # 입력값 확인

    total = sum(numbers)
    print(f"Total: {total}")  # 중간값 확인

    count = len(numbers)
    print(f"Count: {count}")  # 중간값 확인

    average = total / count
    print(f"Average: {average}")  # 최종값 확인

    return average
```

# 🐛 간단한 디버깅 기법 (계속)

## 2. IDE 디버거 사용

- **중단점(Breakpoint)** 설정
- **단계별 실행** (Step Over, Step Into)
- **변수 값 실시간 확인**

## 3. 예외 처리로 디버깅

```
try:
    result = risky_operation()
except Exception as e:
    print(f"Error occurred: {e}")
    print(f"Error type: {type(e)}")
```

# 🔧 실습: 코드 리팩토링

## Before (개선 전)

```
def f(l):
    s=0
    for i in range(len(l)):
        s=s+l[i]
    return s

def g(l):
    m=l[0]
    for i in range(1,len(l)):
        if l[i]>m:
            m=l[i]
    return m
```

## 🔧 실습: 코드 리팩토링

### After (개선 후)

```
def calculate_sum(numbers):
    """숫자 리스트의 합을 계산합니다."""
    total = 0
    for number in numbers:
        total += number
    return total

def find_maximum(numbers):
    """숫자 리스트에서 최대값을 찾습니다."""
    if not numbers:
        return None

    maximum = numbers[0]
    for number in numbers[1:]:
        if number > maximum:
            maximum = number
    return maximum
```

# 👥 팀별 코드 리뷰 체험

## 코드 리뷰 체크리스트

1. **가독성**: 변수명이 명확한가?
2. **효율성**: 더 간단한 방법이 있는가?
3. **에러 처리**: 예외 상황을 고려했는가?
4. **PEP 8**: 스타일 가이드를 준수했는가?

## 실습 과제

- 각자 간단한 함수 작성
- 팀원끼리 코드 리뷰
- 개선점 찾아서 함께 토론

# 📚 1주차 정리

## 오늘 배운 것들

1. **쪽지시험 리뷰**: 헷갈리는 개념들 정리
2. **PEP 8**: Python 코딩 스타일 가이드
3. **좋은 변수명/함수명** 짓는 법
4. **주석과 docstring** 작성법
5. **디버깅 기법**: print 디버깅 → IDE 디버거
6. **코드 리팩토링** 실습

# 🎯 과제

1. **개인 과제**: 기존에 작성한 Python 코드를 PEP 8 스타일로 리팩토링하기
2. **복습**: 오늘 리뷰한 쪽지시험 문제 다시 풀어보기
3. **예습**: Python 내장 함수 enumerate(), zip() 찾아보기

**질문이 있으면 언제든 편하게 물어보세요!** 🙋‍♀️🙋‍♂️