Python 독학 노트

**변수 선언:**

num = 5

s = "Hello"

**String 합치기:**

str = “Hello”

print str + “World!”

**if 문:**

if num == 5 or str != “H”:

print(“Hi!”)

elif num == 4:

print(“Hello!”)

else:

print(“No”)

**함수:**

def func\_name(par):

쏼라쏼라

func\_name(arg)

Import:

**Generic Import (일단 모듈 넣기):** import math -> 이럴 경우 math.sqrt() 이렇게 해야한다

**Function Import(모듈 안에 있는 특정 함수 넣기):** from math import sqrt -> 아럴 경우 그냥 sqrt()만 쓰면 된다

**Universal Import(math.sqrt()와 같이 . 없이 바로 sqrt만 사용하기):**

from math import \*

주의사항: 유니버설 임포트(Universal Imports)는 겉으로 보기엔 대단해 보이지만, 한 가지 정말 중요한 이유로 그리 좋은 아이디어가 아니라고 할 수 있습니다: 유니버설 임포트를 사용하면 어마어마한 양의 변수와 함수를 불러올 수 있지만, 이들이 여러분이 작성한 변수나 함수와 충돌을 일으킬 수 있기 때문입니다.

만약 여러분이 직접 만든 sqrt라는 이름의 함수가 있는 상태에서 math 모듈을 제네릭 임포트(generic import)를 통해 불러온다면, 여러분의 함수는 안전합니다: 여러분이 만든 함수는 sqrt가 될 것이고, 모듈로 부터 불러온 함수는 math.sqrt가 될 것이기 때문입니다. 하지만 만약 유니버설 임포트(universal import)를 통해 from math import \*라고 모듈을 불러오면, 문제가 생깁니다: 두 가지 다른 함수가 똑같이 sqrt라는 이름을 사용해서 충돌을 일으키기 때문이죠.

심지어 여러분이 작성한 정의가 모듈로 부터 불러들인 이름들과 직접적으로 충돌하지 않는다고 해도 import \*를 통해 여러 모듈들을 한꺼번에 불러오면 어떤 변수와 함수가 어떤 모듈로부터 왔는지 파악하기 어려울 겁니다. 이는 마치 제각기 다른 여러 박스로부터 수 없이 많은 도구들을 여러분의 방 안에 무작위로 쏟아부어 섞은 다음, 박스는 치워버려서 각각의 도구가 어느 박스로부터 왔는지 찾을 수 없게 만드는 것과 같습니다.

이러한 이유로, 필요에 따라 제네릭 임포트를 사용하거나 특정 변수 및 함수를 모듈로부터 불러오는 것이 가장 좋은 방법입니다.

파이썬의 함수들:

var.upper() -> 문자열 대문자로

var.lower() -> 문자열 소문자로

length = len(string) -> string의 길이 반환

toStr = str(num) -> num이 숫자일 시 이를 문자열 형식으로 반환

num = max(num1, num2…) -> 최대값 (min(), abs()도 있음)

print type(23) -> 괄호 안에 있는 변수의 형식을 반환

그리고 주의할점!!! type으로 변수 형식 반환 시 string형식이 아니라 그냥 자료형 형식이다. 예를 들어, type(23)으로 int가 반환됐는지 확인할 경우, if type(23) == “int가 아니라 if type(23) == int 이다.