**프로그래밍 핵심 개념 in Python**

1. 자료형

자료형(Data type): 계산기가 숫자를 사용하듯, 프로그래밍에서 사용하는 것들.

(1) 코멘트

# 붙이고 내용을 쓰면, #이하의 내용은 코드로 인식하지 않음. = 주석을 다는 것

->복잡한 코드를 설명, 하다만 부분을 표시, 다른 개발자와의 소통 등을 위해 사용함.

(2)숫자열(Number)

정수(Integer): -2, -1, 0, 1, 2

소수(Floating Point): 1.7, 2.0 (정수부분 + 소수부분)

#파이썬은 일반적인 계산순서(사칙연산)를 따른다.  
print(2 + 3 \* 4)  
print(2 \* (2 + 3))  
  
#덧셈  
print(4+7)  
print(4.0+7.0)  
print(4.0+7)  
  
#뺄셈  
print(2 - 4)  
print(2.0 - 4.0)  
print(2.0 - 4)  
  
#곱셈  
print(5 \* 3)  
print(5.0 \* 3.0)  
print(5.0 \* 3)  
  
#나머지를 구하는 연산  
print(7 % 3)  
print(7.0 % 3.0)  
print(7.0 % 3)  
  
#거듭제곱  
print(2 \*\* 3)  
print(2.0 \*\* 3.0)  
print(2.0 \*\* 3)  
  
#파이썬에서는 integer보다 Floating Point가 쎄다, 하나라도 소수형이면 결과값은 소수형- 나눗셈은 다름  
  
#나눗셈: 매번 다 소수형으로 나옴  
print(7 / 2)  
print(7.0 / 2)  
print(6 / 2)  
print(6.0 / 2.0)  
  
#floor division(버림 나눗셈: 소수부분만 버리고 정수부분만 남겨둠)  
print(7 // 2)  
print(8.0 // 3) #단, 하나라도 소수형이면, 결과값도 소수형  
  
#round(빈올림)  
print(round(3.1415926335))  
print(round(3.1415926335, 2)) #소수점 아래 두자리까지 반올림

(3) 문자열(String)

따옴표 안에 키보드로 쓸 수 있는 문자들을 표현한 것 Ex. "Hello world!", "2"

문자열 연결=String Concatenation

덧셈) “Hello”+”World”=”HelloWorld”

곱셈) “Hello” \* 3 =”HelloHelloHello”

print("'I' am excited to learn Python!")  
print("I\'m \"excited\" to learn Python.") #문자열에 역슬래쉬+따옴표로 따욤표를 의도했다는 것을 표시하기

☞ 2, 2.0, "2" 이 세가지는 다 다른 type임. 2+5=7이지만,"2"+"5"="25"임

(4) 형 변환

Type Conversion: 값을 한 자료형에서 다른 자료형으로 바꾸는 것.

#형 변환  
print(int(3.8)) #소수형을 정수형으로 형변환  
print(float(3)) #정수형을 소수형으로 형변환  
print(int("2") +int("5")) #문자열을 정수형으로 변환  
print(float("1.1") + float("2.5")) #문자열을 소수형으로 변환  
print(str(2) + str(5)) #정수형을 문자형으로 변경  
  
age = 7  
print("제 나이는 " + str(age) + "살 입니다.")  
  
print(int("Hello world!")) #알파벳은 숫자로 바꾼다는게 논리적으로 말이 안되니 에러가 뜸

(5) 문자열 포맷팅

1) 포멧 메소드: 현재 가장 많이 사용하는 방식

rint("오늘은 {}년 {}월 {}일입니다.".format(year, month, day))  
  
date\_string = "오늘은 {}년 {}월 {}일입니다."  
print(date\_string.format(year, month, day))  
  
date\_string = "내일은 {}년 {}월 {}일입니다."  
print(date\_string.format(year, month, day + 1))  
  
print("저는 {}, {}, {}를 좋아합니다".format("박지성", "유재석", "빌 게이츠"))  
print("저는 {2}, {1}, {0}를 좋아합니다".format("박지성", "유재석", "빌 게이츠")) #컴퓨터는 0,1,2 순으로 카운트  
  
data\_string = "저는 {}, {}, {}를 좋아합니다."  
print(data\_string.format("박지성", "유재석", "빌게이츠"))  
  
num\_1 = 1  
num\_2 = 3  
print("{0} 나누기 {1}은 {2}입니다.".format(num\_1, num\_2, num\_1 / num\_2))  
print("{0} 나누기 {1}은 {2:.2f}입니다.".format(num\_1, num\_2, num\_1 / num\_2)) #소수점 둘째 자리까지 반올림

2) 포멧 스트링: %s, %d를 사용하는 옛날 방식

name = "최지웅"  
age = 32  
print("제 이름은 %s이고 %d살입니다." % (name, age))

3) f-string: 새로운 방식, 미래에 더 많이 쓰일 수도?

name = "최지웅"  
age = 32  
  
print(f"제 이름은 {name}이고 {age}살입니다.")

(6) 불 대수

일상적인 논리를 수학적으로 표현한 것.

\*명제: 참 또는 거짓이 확실한 문장(질문이나 주관적인 것들은 명제가 아님.)

값: 진리값(참/거짓)을 값으로 사용, 그래서 진리 값은 True, 또는 False만 존재

연산: AND, OR, NOT

AND: 명제 X와 Y가 모두 참일 때만 x AND y가 참

OR: 명제 X와 Y 중 하나라도 참이면, x OR y가 참

NOT: 반대로 뒤집어 주는 역할, 참->거짓, 거짓->참

(7) 불린형(Boolean)

True, False (참과 거짓, 주로 값을 비교할 때 사용)

print(2 > 1 and "Hello" == "Hello") #True and True  
print(7 == 7 or (4 < 3 and 12 > 10)) # True or False  
  
x = 3  
print(x > 4 or not (x < 2 or x == 3)) #false or not (false or true) = false or false= false

(8) Type 함수

해당 코드가 무슨 자료형인지 알려줌.

print(type(3)) # class ‘int’  
print(type(3.0)) # class ‘float’  
print(type("3")) # class ‘str’

print(type("True"))# class ‘str’  
print(type(True))# class ‘bool’

print(type(print)) #class ‘builtin\_function\_or\_method’

print(type(4 / 2)) #class ‘float’ (2.0이 나오니까)

2.추상화

추상화=복잡한 내용은 숨기고 주요 기능만 -> 변수(Variable), 함수(Function), 객체(Object)

(1) 변수(Variable): 값을 저장하는 것

Ex. X=25 -> 25를 x에 저장

X=X+1

☞ 프로그래밍에서 =는 오른쪽의 값을 왼쪽의 변수에 지정해주는 지정연산자(assignment operator)

(2) 함수(Function): 명령을 저장하는 것

Ex. Print(x+y) -> x+y를 콘솔에 나타나게

내장함수(파이썬 개발자가 미리 만들어 둔 것)

사용자 지정 함수: 새롭게 사용자가 정의하는 것.

def hello():  
 print("Hello!")  
 print("Welcome to Codeit!") #hello 함수는 이 두줄을 저장하는 함수

☞ 이때 def hello():를 해더라고 함.

☞ 해더의 hello를 파라미터(함수에 넘겨주는 값)이라고 함.

한 함수에 여러 개의 파라미터가 올 수 있음.

def print\_sum(a, b): #파라미터 두개  
 print(a + b)  
print\_sum(7, 3)  
def print\_sum(a, b, c): #파라미터 세개  
 print(a + b + c)  
print\_sum(7, 3, 2)  
def print\_sum(num\_1, num\_2, num\_3, num\_4): #파라미터 네개  
 print(num\_1 + num\_2 + num\_3 + num\_4)  
print\_sum(7, 3, 2, 1)

(3) Return문

Return: 함수에게 어떤 정보를 주면 다른 정보로 돌려준다.

Return문의 역할: 값을 돌려주기, 함수 즉시 종료하기

def square(x):  
 print("함수 시작")  
 return x \* x  
 print("함수 끝") # return문은 함수를 종료시켜, 이 부분이 나오지 않음(데드코드)

python에서 return문이 없으면 none이 출력

def print\_square(x):  
 print(x \* x)  
  
def get\_square(x):  
 return(x \* x)  
  
print\_square(3) # 9를 프린트  
get\_square(3) #그냥 이 자체가 9로 대체되었지만, print가 없으니 그냥 아무것도 출력 안됨  
print(get\_square(3))# 9를 프린트  
print(print\_square(3)) # print\_square(3)은 9이지만, return문이 없으니 none이 return됨.

(4) 기본값 설정

파라미터에 default value(기본값)을 설정할 수 있음. 기본값을 설정하면 함수 호출시에 꼭 파라미터에 값을 안넘겨 주어도 됨(optional parameter).

def myself(name, age, nationality="한국"):  
 print("내 이름은 {}".format(name))  
 print("나이는 {}살".format(age))  
 print("국적은 {}".format(nationality))  
  
myself("코드잇", 1, "미국") #옵서녈 파라미터 제공함.  
print()  
myself("코드잇", 1) #옵셔널 파라미터 제공 안함.

단, optional parameter는 꼭 마지막에 있어야 함.

def yourself(name, age="22", nationality="한국"):  
 print("내 이름은 {}".format(name))  
 print("나이는 {}살".format(age))  
 print("국적은 {}".format(nationality))  
  
  
yourself("강윤경")

일반 파라미터가 앞에만 있다면, optional parameter는 두개 이상도 가능

(5) Syntatic Sugar

자주 쓰이는 표현을 더 간략하게 쓸 수 있게 해줌.

Ex. x=x+1 (x+=1), x=x-3 (x-=3)

x=x\*2(x\*=2)

x=x/2 (x/=2)

x=x%7 (x%=7)

(6) Scope (범위)

Scope: 변수가 사용 가능한 범위

로컬 변수: 함수 내에서 정의한 변수, 즉 변수x가 유효한 범위는 변수를 정의한 함수 안. (파라미터도 로컬 변수임.)

글로벌 변수: 함수 밖에서 정의한 변수, 모든 곳에서 사용 가능함.

x = 2 #글로벌 변수  
def my\_function():  
 x = 3 #로컬 변수  
 print(x)  
  
my\_function() #3이 출력됨.  
print(x) #2가 출력됨.

함수에서 변수를 사용하면, 로컬 변수를 먼저 찾고 나서 글로벌 변수를 찾음.

(7) Constant(상수)

값이 절대 변하지 않는 것, 단, 모든 글자를 대문자로 해야함(일반 변수와 상수를 구분하고, 실수를 줄이기 위해-상수는 절대로 바꾸지 않겠다는 약속)

PI = 3.14 #원주율 (값이 절대 바뀌지 않는 상수)  
  
def calculate\_area(r): #반지름을 받아서 원의 넓이 게산  
 return PI \* r \* r  
  
radius = 4  
print("반지름이 {}면, 넓이는 {}".format(radius, calculate\_area(radius)))

(8) 스타일

이해하기 쉬운 코드 = 좋은 스타일을 가진 좋은 코드

텍스트이(가) 표시된 사진

자동 생성된 설명

\*파이썬 PET8 스타일 가이드

-이름

모든 변수와 함수 이름은 소문자로 쓰고, 여러 단어일 경우 \_로 나누기.

모든 상수 이름은 대문자로 쓰고, 여러 단어일 경우 \_로 나누기.

변수와 함수 이름은 의미 있게 짓기.

-화이트 스페이스

들여쓰기: 무조건 스페이스 4개

함수 정의: 함수 정의 위아래로 빈 줄이 두 개씩 있어야 하고, 첫 줄이 함수 정의인 경우, 함수 위에는 빈 줄이 없어도 됨.

괄호: 괄호 바로 안에는 띄어쓰기 하지 말기

함수 괄호: 함수를 정의하거나 호출할 때 함수 이름과 괄호 사이에 띄어쓰기 하지 말기.

쉼표: 쉼표 앞에는 띄어쓰기 하지 말기.

지정 연산자: 지정 연산자 앞뒤로 띄어쓰기 하나씩 하기.

연산자: 연산자 앞뒤로 띄어쓰기 하나씩 하기. 하지만, 연산의 우선 순위를 강조하기 위해서는 연산자 앞뒤로 띄어쓰기를 붙이는 것을 권장함.

코멘트: 일반 코드와 같은 줄에 코멘트를 쓰려면, 코멘트 앞에 띄어쓰기 최소 두개 하기.

3. 제어문

**프로그래밍과 데이터 in Python**

**Python 응용하기**