2023/02/27

C, C++과 Python의 차이점

1. Python은 주석을 '#'로 사용하며, 여러줄 주석처리를 할땐 """을 사용한다

```
# 한 줄 주석

"""
여러 줄 주석입니다.
여러 줄을 주석 처리하려면
따옴표 세 개를 연속해서 사용
"""
```

2. 코드를 한번에 입력하려면 [New File]-[Save As]-[Run Module F5] 기능을 사용해야 한다

```
>>> print("Hello World")
Hello World
>>> x=10
>>> y=20
>>> print(x+y)
30
```

바로 바로 print 돼서 놀랬다

[New File]-[Save As]-[Run Module F5] 기능을 사용

```
#my first Python program

print("This is Python program written in a file.")
x=1
y=3
print("x:", x)
print("x:", y)
print("x+y:", x+y)
print("x-y:", x-y)
print("x*y:", x*y)
print("x/y:", x/y)
print("x//y:", x//y)
```

```
This is Python program written in a file.
x: 1
x: 3
x+y: 4
x-y: -2
x*y: 3
x/y: 0.3333333333333333
x//y: 0
```

/ : 주어진 숫자 데이터에 대한 실수형 나눗셈 (소수점 이하 값 유지)
// : 주어진 숫자 데이터에 대한 정수형 나눗셈 (소수점 이하 값 생략)
% : 주어진 숫자 데이터에 대한 모듈로 계산

한번에 주르륵 출력 됐다
이렇게 코드를 한번에 작성하면 디버깅은 어떻게 하는거지??

2023/02/27

공부 하다보면 알겠지 ..

3. 별도로 변수의 자료형을 지정하지 않으면 변수 값이 대입될 때 그 값의 자료형에 따라 자동적으로 결정된다
위 코드에서 x, y 자료형을 선언하지 않았는데 자동적으로 변수값이 결정됨

4. Python에서는 다양한 자료형이 많다. (list, tuple, set, dict)

- type() 함수
: 데이터 타입을 확인하는 용도로 사용

type() 함수로 확인할 수 있는 자료형

- int: 정수형
- float: 실수형
- str: 문자열
- bool: 불리언 값
- list: 리스트
- tuple: 튜플
- set: 집합
- dict: 딕셔너리

```
a = input("input a data: ")
print("input data a=", a)
print("type of a is", type(a))

b = input("input an integer data :")
print("input data b=", b)
print("type of b is", type(b))
sum = a+b #string concatenation
print("sum is:", sum)
```

```
input a data : 1
input data a = 1
type of a is <class 'str'>
input an integer data :2
input data b= 2
type of b is <class 'str'>
sum is: 12
>>>
```

```
#Python program variables

x = 1 #integer
print("x=", x)
print("type(x) =", type(x))

x = 2.345 #float
```

2023/02/27

```
print("x=", x)
print("type(x) =", type(x))

x = [1, 2, 3, 4] #list
print("x=", x)
print("type(x) =", type(x))

x = (7, 8, 9, 10) #tuple
print("x=", x)
print("type(x) =", type(x))

x = {'A':1, 'B':2, 'C':3} #dict
print("x=", x)
print("type(x) =", type(x))

x = {1, 2, 3, 4} #set
print("x=", x)
print("type(x) =", type(x))
```

```
x= 1
type(x) = <class 'int'>
x= 2.345
type(x) = <class 'float'>
x= [1, 2, 3, 4]
type(x) = <class 'list'>
x= (7, 8, 9, 10)
type(x) = <class 'tuple'>
x= {'A': 1, 'B': 2, 'C': 3}
type(x) = <class 'dict'>
x= {1, 2, 3, 4}
type(x) = <class 'set'>
>>>
```

list
list는 여러 개의 데이터를 순서대로 저장하는 자료형이다. (수정 가능한 자료형)
[]를 사용하여 생성하며, ,로 각 요소를 구분.
인덱스를 사용하여 요소에 접근할 수 있으며, 슬라이싱을 통해 일부 요소를 추출할 수 있다.
append(), extend(), insert(), remove(), pop() 등 다양한 메소드를 제공

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])  # "apple"
print(fruits[1:])  # ["banana", "cherry"]
fruits.append("orange")
print(fruits)  # ["apple", "banana", "cherry", "orange"]
```

tuple
tuple은 list와 비슷하지만, 수정이 불가능한 자료형
()를 사용하여 생성하며, ,로 각 요소를 구분
인덱스를 사용하여 요소에 접근할 수 있으며, 슬라이싱을 통해 일부 요소를 추출할 수 있다.

2023/02/27

```
fruits = ("apple", "banana", "cherry")
print(fruits[0])  # "apple"
print(fruits[1:])  # ("banana", "cherry")
```

dict

dict는 키(key)와 값(value)의 쌍으로 이루어진 자료형

{}를 사용하여 생성하며, key:value 형식으로 각 요소를 구분

키를 사용하여 값에 접근할 수 있으며, keys(), values(), items() 등 다양한 메소드를 제공

```
fruits = {"apple": 1, "banana": 2, "cherry": 3}
print(fruits["apple"])  # 1
print(fruits.keys())  # ["apple", "banana", "cherry"]
```

set

set은 중복을 허용하지 않는 자료형입니다.

{}를 사용하여 생성하며, ,로 각 요소를 구분

인덱스를 사용할 수 없으며, add(), update(), remove(), pop() 등 다양한 메소드를 제공

```
fruits = {"apple", "banana", "cherry"}
print("apple" in fruits)  # True
fruits.add("orange")
print(fruits)  # {"apple", "banana", "cherry", "orange"}
```