**Shell vs Editor**

**Shell 창**은 '명령어를 한 줄 단위로 입력하는 창'이야

ex)

>>> a = 10 (나: a는 이제 10이야)

>>> a (나: a가 뭐야?)

10 (컴퓨터: 10이야)

>>> b = 20 (나: b는 이제 20이야)

>>> c = 49 (나: c는 이제 49야)

>>> a+b (나: a+b가 뭐야?)

30 (컴퓨터: 30이야)

**Editor 창**은 '명령어를 모두 입력하고, F5 버튼으로 한번에 모든 명령어를 실행하는 창'이야

ex)

a = 10 (a는 이제 10이야)

b = 20 (b는 이제 20이야)

c = a + b (c는 이제 a+b(=30)이야)

print(c) (c의 값이 무엇인지 나한테 보여줄래(출력해줄래)?)

(윗 파일을 실행할 경우)

30 (c 값(=30)이 무엇인지 출력했어)

**파이썬의 자료형1**

자료형의 종류에는 **정수형(int), 실수형(float), 문자열(str), 불(bool), 리스트(list), 딕셔너리(dict), 튜플(tuple)** 등이 있어.

**정수형(int)**는 소숫점(.)이 없는 숫자를 의미해.

**실수형(float)**은 소숫점이 붙은 모든 숫자를 의미해. (6.0, 10.0 같은 것도 다 실수형이야.)

정수형과 실수형의 차이점은 소숫점의 여부 차이고, 나머지는 숫자라는 점에서 공통이야.

숫자니까 얘내들은 덧셈(+) 뺄셈(-) 곱셈(\*) 나눗셈(/) 같은 걸 할수 있겠지?

**문자열(str)**은 우리가 아는 모든 문자를 의미해.

' ', " "(작은따옴표, 큰따옴표 1쌍)이나 ''' ''', """ """(3쌍) 안에 들어간거는 다 문자열이야. 상황에 맞게 4개 중 아무거나 사용하면 돼. 그러니 1은 정수, 1.0은 실수지만 "1"은 문자열이라고 봐야겠지?

문자열은 +나 \*를 사용할 수 있는데, 이건 사칙연산을 하는게 아니야.

예를 들면, Shell창에

>>> "안녕"+"반가워" 라고 치면,

'안녕반가워' 라고 뜰거야. 문자열에서 +는 앞뒤 문자열을 붙인다는 의미야.

그리고 Shell창에

>>> "안녕"\*3 이라고 치면,

'안녕안녕안녕' 이라고 뜰꺼야. 문자열에서 \*는 같은 문자열을 ~개 붙인다는 의미야.

우리가 아는 사칙연산은 문자열에서는 불가능해.

하늘에서 땅을 빼라? 물통에 마우스를 곱해라? 스마트폰을 필통으로 나눠라? 이런게 불가능 한 것 처럼. 그래서 조심해야할 건, "1", "124" 이런 것도 컴퓨터에서는 아까의 물통, 마우스, 스마트폰처럼 하나의 문자열로 인식해서 계산을 못해.

그래서 Shell창에

>>> 1\*3 을 입력하면

3 이 나오지만

>>> "1"\*3 을 입력하면

'111' 이 나오는거야.

**불(bool)**은 참(True), 거짓(False) 두 값만을 가져. 참/거짓을 가르는 자료형이야.

예를 들면, Shell창에 bool(1>3)이라고 치면 False라 뜰거야. 1이 3보다 크다는 건 거짓말이니까.

**파이썬의 자료형2**

**리스트(list)**는 [] 대괄호를 사용해. 말 그대로 리스트, 목록이라고 생각하면 돼.

예를 들면 Shell창에서

>>> fruit = [‘apple’, ‘banana’, ‘grape’]

라고 한다면, fruit는 리스트의 이름이고, ‘apple’, ‘banana’, ‘grape’는 리스트의 원소야.

이러한 리스트에서 각 원소는 **인덱스(index)**, 쉽게 말하면 ‘순서’를 가지고 있어.

예를 들면, 아까 fruit 리스트에서 ‘apple’은 첫번째, ‘banana’는 두번째의 순서에 있지?

하지만 조심해야할 건, 파이썬에서는 0이 시작 숫자야. 그러니까 ‘apple’의 인덱스값은 0이야.

Shell창에 리스트이름[인덱스값]을 입력하면 그 리스트의 해당 인덱스에 해당하는 값을 불러.

아까 fruit를 이어서 사용하면

>>> fruit[0] 이라고 하면(fruit 리스트에서 0의 인덱스를 가진 값은 뭐야? 라는 의미)

apple 이라고 뜰거야.(사과야. 라는 의미)

한편, 우리는 ‘슬라이스’라는 개념을 통해 리스트에서 여러 원소를 동시에 꺼낼 수도 있어.

슬라이스라는 이름은 잊어도 되고, 그냥 :(콜론)을 이용해서 인덱스 범위를 설정해서 여러 개를 동시에 가져온다고 생각하면 돼.

형태는 **리스트이름[인덱스:인덱스]**야. 앞에서 입력한 인덱스의 값부터, 뒤에 입력한 인덱스의 전 값까지 불러온다는 의미야.(앞 숫자 이상 뒷 숫자 미만의 인덱스에 해당하는 값 전부랑 똑같은 의미야) 리스트이름[0:3] 이라고 하면 리스트의 인덱스 0, 1, 2에 해당하는 값을 불러오는 거야.

만약에 콜론 앞에 숫자를 안 쓰고 비우면, 리스트 처음부터~, 뒤에 숫자를 안 쓰면 ~끝까지 라는 의미야. 리스트[:2]면 해당 인덱스는 0,1,2겠지? 그리고 리스트[3:]면 인덱스 3부터 끝까지겠지?

예를 들면 Shell창에서

>>> drink = ['coke', 'soda', 'juice', 'coffee', 'water'] 라고 하고

>>> drink[:3] 라고 하면

['coke', 'soda', 'juice'] 라고 뜰거야. 다른 것도 꼭 실험해 봐.

**튜플(tuple)** () 소괄호, 혹은 아예 괄호를 안 쓰기도 해.

리스트와 거의 동일하지만, 내부의 값을 변경할 수 없다는 차이점이 있어.

리스트에서 설명하지 않았던 건데, 리스트 설명에서 쓴 drink 리스트를 사용해보자면 drink[0]는 ‘coke’지? 그런데 Shell창에 다음처럼 입력하면

>>> drink[0] = ‘coca’ (drink 리스트의 인덱스 0에 해당하는 값을 ‘coca’로 변경)

>>> drink (drink 리스트를 보여달라는 의미)

['coca', 'soda', 'juice', 'coffee', 'water'] (drink 리스트 출력)

이렇게 리스트 내 해당 인덱스의 값을 변경할 수도 있어. 하지만 튜플에서는 이런 수정이 안 돼.

**딕셔너리(dict)** {} 중괄호를 사용해. { 키(key) : 값(value) }의 형태를 가지고 있어.

키(key)는 리스트의 인덱스 같은 개념이야. 예를 들면 age = {‘철수’ : ‘20살’} 이라는 딕셔너리가 있다고 하면, age[‘철수’]는 ‘20살’이라는 값인거지.

이런 딕셔너리는, ‘인덱스’를 사용하는 리스트, 튜플과 다르게 ‘키’를 사용해서 순서가 없다는 차이점이 있어.

변수, print, input

**변수**는 ‘일정한 자료를 담는 주머니’ 같은 거야. 바로 예를 들자면, Shell 창에서

>>> a = [‘사과’, ‘바나나’, ‘음료수’] (a라는 변수는 이제 [‘사과’, ‘바나나’, ‘음료수’]라는 리스트이다.)

>>> a (a는 무엇인가요? 라는 의미)

[‘사과’, ‘바나나’, ‘음료수’] (a가 무엇인지 출력한 것)

앞에서 소개한 자료형에서도 이런 ‘=’을 많이 사용한 걸 봤을거야. 여기서 =는 대입의 의미야. 우리가 생각하는 ‘똑같다는 의미’의 등식은 ==를 사용하는 데, 나중에 배울거니 이번엔 패스하자.

그러니까 (좌항) = (우항)은 ‘이제 (좌항)은 (우항)이라고 하자~’라는 의미인거야.

헷갈리는 예시를 하나 들어볼게. Shell 창에

>>> a = 2 (a는 이제 2라는 정수를 의미하는 변수다.)

>>> b = 3 (b는 이제 3이라는 정수를 의미하는 변수다.)

>>> x = a (x는 이제 a의 값(=2)을 갖는 변수다.)

>>> a = b (a는 이제 b의 값(=3)을 갖는 변수다.)

>>> b = x (b는 이제 x의 값(=2)을 갖는 변수다.)

라고 하면,

>>> a (a의 값은? 이라는 의미)

3 (a라는 변수는 3이라는 정수값을 가진다는 의미)

>>> b (b의 값은? 이라는 의미)

2 (b라는 변수는 2라는 정수값을 가진다는 의미)

이 예시가 이해가 안 되면 다시 물어 봐.

이런 변수는 파이썬의 자료형을 담는 주머니 역할을 해서, 매번 번거롭게 자료형을 적을 필요 없이 쉽게 그 자료를 불러올 수 있다는 장점이 있어. 예를 들면 a = [‘1’,’2’,’3’,’4’,’5’]라는 리스트가 있다고 하고 이 리스트를 여러 번 사용하고 싶으면, 매번 [‘1’,’2’,’3’,’4’,’5’]를 입력할 번거로움 없이 a 한 글자만 입력하면 그 리스트를 불러올 수 있어.

**print**는 ‘괄호 안의 값을 출력하는(직접 눈에 보여주는) 함수’야. 이미 실습에서도 몇 번 접해봐서 알겠지만, 매우 기본적이면서도 자주 쓰이는 함수야.

사용방법은 **print(출력하고 싶은 값)**이야. 이 때, 출력하고 싶은 값은 파이썬의 모든 자료형, 그리고 해당 코드에 이미 입력된 변수 등을 입력할 수 있어.

만약 ‘Hello World!’라는 문자열을 출력하고 싶다면, 두 가지 방법이 있어. Shell창 켜서 직접 해 봐.

1. >>> print(‘Hello World!’) (혹은 >>> print(“Hello World!”))

2. >>> a(혹은 아무 이름) = ‘Hello World!’

>>> print(a)

1번의 경우, 프린트 함수 안에 직접 문자열을 넣은 거고, 2번의 경우, 변수를 활용해서 프린트 함수 안에 변수를 넣어 출력한 함수야. 둘다 실행하면 Hello World! 라고 잘 뜰거야.

이런 프린트 함수는 Editor에서 더 빛이 나. 이번엔 Editor 창을 켜서 이걸 입력하고 실행해 봐.

a = 10

b = 23

print(a+b)

Shell창에서는 그냥 변수만 입력해도 그 값이 떴지만(출력됐지만), Editor 창에서는 출력을 하고 싶으면 꼭 print를 사용해야하기 때문이야. 만약에 저기서

a = 10

b = 23

a+b

라고 했으면 Shell창에서라면 몰라도 Editor 파일 저장 후 실행을 해도 아무 것도 안 떴을거야.

여기서 조심해야할 것은, 문자열을 입력할 때, ‘’나 “”를 사용하지 않으면 파이썬은 그것을 변수라고 인식한다는 거야. 예를 들면

print(Hello!)라고 하면 파이썬은 ‘Hello!’라고 출력하는 게 아니라, Hello!라는 변수가 없다고 오류를 일으킬거야. 물론 Hello! = ‘Hello!’ 같이 변수를 미리 설정했다면, 그대로 출력을 하겠지?

**input**은 프로그램(코딩 파일)을 실행한 사람에게 값을 **입력**할 기회를 주는거야.

만약 가위바위보를 코딩으로 만든다고 한다면, 게임을 킨 사람이 ‘가위’나 ‘바위’나 ‘보’를 정해서 입력해줘야겠지? 하지만 만약 이 input이 없다면, 이 사람은 일일히 코딩 파일을 열고, 거기서 자기가 값을 수정하고, 저장하고, 실행해야겠지? 엄청 불편할거야. 하지만 input 함수로 입력자의 값을 받는다면, 훨씬 수월해질거야.

사용방법은 **input(출력하고 싶은 값)** 으로 사용하면 돼. 여기서 ‘출력하고 싶은 값’은 보통 입력 받고 싶은 값에 대한 설명을 적어 놔. 가위바위보를 예로 들자면, ‘가위, 바위, 보 중 하나를 입력하세요.’ 등을 설명해줘야 플레이어가 이 셋 중 하나를 입력해주겠지? 실제로 나도 가위바위보 프로그램을 만들 때, 플레이어의 값을 받기 위해

mine = input("'가위', '바위', '보' 중 하나를 선택하세요! >> ")

라고 했어. 여기서 플레이어가 ‘가위’를 입력하면, mine이라는 변수는 이제 ‘가위’라는 문자열이 되겠지?

참고로 input 함수로 받는 모든 입력값은 ‘문자열’로 받아. 그러니까 1을 입력해도 파이썬은 ‘1’로 받아들이는거지. 그래서 만약 입력값을 숫자처럼 사용하고 싶으면, int(input(~)), float(input()) 처럼 정수형이나 실수형으로 바꿔주는 int 함수, float 함수를 추가로 사용해야해.