

Python Class 1

Hello World

어떤 프로그램 언어든 처음은 항상 "Hello World"를 맞이하는 방법부터 시작합니다. 파이썬(Python)은 매우 간단하게 `print()` 라는 함수를 통하여 출력하고 싶은 문자열을 출력할 수 있게 해줍니다. `()` 안에 들어갈 내용은 문자열이면 `' '` 나 `" "` 으로 감싸주고, 숫자일 경우 그냥 입력하시면 됩니다.

```
>>> print("Hello World")
      print('Hi')
      print(1234)
      print("This is Minsu's Note.")
```

```
Hello World
Hi
1234
This is Minsu's Note.
```

문자열 출력에 관해 파이썬은 한가지 더 살펴 볼 것이 있습니다. 아래 예시를 통해 살펴 보겠습니다...

```
>>> print("""Hello
Python
World""")
```

```
Hello
Python
World
```

여러줄 문자열을 표현하고 싶을때 다른 언어에서는 `escape(이스케이프)` 문자를 통해 `\n` 을 이용하여 여러줄을 표현하였는데 파이썬에서는 `""" 나 '''` 처럼 따옴표 세개를 연속으로 붙여서 쓰면 위의 예시처럼 여러줄 문자열을 한번에 표현할 수 있습니다.

Format Specifier

단순한 출력을 벗어나서 *문자열 포맷* 을 이용하여 출력 형식을 지정하여 출력 해봅시다. 먼저 `%s` , `%d` 에 대해 알아보면... `%s` 는 문자열을 지정받아 출력하고, `%d` 는 숫자를 지정받아 출력합니다.

```
>>> print("%s apples" % 'five')
```

```
five apples
```

```
>>> print("%d apples" %5)
```

```
5 apples
```

또한, 문자열이나 숫자를 특정 변수에 담아서 그 변수를 통해 값을 출력 할 수도 있습니다.

```
>>> number =3
      print("% apple" %number)
```

3apple

인덱싱(Indexing)

다음은 인덱싱(Indexing)에 관해 살펴보도록하겠습니다. 우선 인덱싱(Indexing)이란... 무엇인가를 "가리킨다" 는 의미이며 각 문자에 대해 숫자 0부터 시작하여 차례로 1씩 증가된 Index가 붙습니다.

간단히 살펴보면...

```
Life is too short, You need Python
0      1      2      3
0123456789012345678901234567890123
```

위의 예시처럼, 각 문자열에 대해 _Index_가 붙으며 그 시작은 0입니다.

문자열 인덱싱

이를 활용하여 **문자열 인덱싱** 에 대해 살펴보겠습니다. 아래와 같이 **a** 문자열에 대해 각 인덱스 번호를 조합하여 **hello** 가 출력되도록 할 수도 있습니다.

```
>>> a = "life is too short, You need Python"
      print(a[13] + a[3] + a[0] + a[0] + a[9])
```


hello

문자열 슬라이싱

문자열 슬라이싱은 위의 예시처럼 어떤 문자열에서 필요한 부분만 뽑아내기 위해 사용되는 것입니다. 다만 위의 예시와 차이점은 위의 예시는 각 문자 하나씩을 뽑아서 조합한 것이고, 슬라이싱은 문자열의 범위를 지정하여 문자 하나 이상을 뽑아낼 수 있다는 것입니다.

그럼 아래 예시를 살펴 보도록 하겠습니다...

```
>>> a = "20010331Rainy"
>>> year = a[:4]
>>> day = a[4:8]
>>> weather = a[8:]
>>> year
'2001'
>>> day
'0331'
>>> weather
'Rainy'
```

예시에서 **[:]** 이런 형태가 눈에 보일 텐데요... 이것이 슬라이싱입니다.  안에 **[시작번호:끝번호]** 형태로 문자열에서 뽑아내고 싶은 범위를 지정하는 것 입니다. 다만 주의할 점은 **시작번호** 는 범위에 포함되지만 **끝번호** 는 범위에 포함되지 않습니다.

리스트인덱싱

그럼 이번에는 **리스트 인덱싱** 에 대해 살펴보겠습니다. 아래와 같이 리스트 변수 **a, b, c, d** 를 선언하였습니다.

```
a=[]
b=[1,2,3]
c=['sunny', 'rainy']
d=[1,2,['sunny','rainy']]
```

리스트는 **a=[]** 처럼 아무런 요소를 갖고 있지 않은 **빈 리스트** 일 수도 있으며... 보통은 리스트 **b** 와 **c** 처럼 **문자열** 이나 **숫자형** 요소를 갖고 있습니다. 그리고 **d** 처럼 **리스트 자체를 요소로** 갖는 리스트도 있을 수 있습니다.

그럼 위에 선언 된 값을 바탕으로 아래 예시를 살펴보면 **b[-1]** 은 무엇을 뜻할까요?

```
>>> b[-1]
```

3

출력된 **3** 을 통해 알수 있듯이 리스트 요소를 거꾸로 카운팅하기 위해 **-** 기호를 사용하여 **-1** 은 뒤에서 부터 첫번째를 의미합니다. 그리고 리스트의 내부 리스트 요소는 아래와 같은 방식으로 출력이 가능합니다.

```
>>> print(d[-1])
      print(d[-1][0])
      print(d[-1][1])
```

```
['sunny', 'rainy'] sunny rainy
```

조건문

그럼 방금 배운 리스트를 활용하여 **조건문** 에 대해 알아 보겠습니다. 먼저 아래와 같이 리스트 변수 **pocket** 을 선언하겠습니다.

```
>>> pocket=['card', 'phone', 'money']
```

그럼 이야기를 붙여 보겠습니다. 우리 **pocket** (주머니)에 **'money'** (돈)이 있으면 **'taxi'** 를 타고, 아니면 집까지 **'walk'** 를 해야 하는 상황입니다. 우리 걸어가게 될까요? 택시를 타고 가게 될까요? **파이썬** 에게 물어보죠^^

```
>>> if 'money' in pocket:
      print('taxi')
    else:
      print('walk')
```

taxi

다행히 힘들이지 않고 택시를 타고 집에 갈 수 있게 되었네요^^ 그럼 다음 이야기를 한번 살펴 보죠 !

이번 이야기는 주머니에 **cash** 가 있으면 **taxi** 를 부르고, **card** 가 있으면 **card taxi** 를 부르는 겁니다. 만약 둘다 없으면? **walk** 하게 되겠죠...ㅠㅠ 이번에도 **파이썬** 에게 물어보겠습니다.

```
>>> pocket=['card', 'phone', 'money']
      if 'cash' in pocket:
        print('taxi')
```

```
elif 'card' in pocket:
    print('card taxi')
else:
    print('walk')
```

card taxi

다행히 이번엔 card 가 있었네요^^ 덕분에 card taxi 를 타고 집에 잘 갔습니다.

* __반복문__ *

이번엔 여태 배운 문자열 포맷 과 조건문 을 포함한 채로 반복문 을 새롭게 배워 보겠습니다. 반복문 은 for A in B 이라는 문법을 통해, 배열이나 리스트형 변수 B 로 부터 차례로 각 요소를 A 라는 변수에 담아서 처리를 합니다.

아래 예시를 통해 추가 설명을 진행하겠습니다...

```
>>> scores=[99,25,67,75,80]
>>> number=0
    for score in scores:
        number=number+1
        if score >= 60 :
            print("%d 번 학생 합격" %number)
        else:
            print("%d 번 학생 불합격" %number)
```

```
1 번 학생 합격
2 번 학생 불합격
3 번 학생 합격
4 번 학생 합격
5 번 학생 합격
```

예시를 보시면 우선 scores 라는 리스트형 변수에 숫자형 요소들을 포함한 채로 선언하였습니다. 그 다음, number 라는 변수를 0으로 초기화 시키고 위에 선언하였던 scores 로 부터 각 숫자형 요소를 score 라는 변수에 담아서 조건문 을 수행하고 있습니다.

요약하자면 점수가 60점 이상인 학생들은 합격이라는 문구를 출력하고 60점 미만인 학생은 불합격을 출력하는 내용입니다. number 는 각 학생들을 구분하기 위해 반복문 이 진행되는 동안 값을 1씩 증가시켜 번호를 매겨주기 위한 변수였습니다.

함수

다음은 함수에 대해 알아 보겠습니다. 먼저 아래 예시를 보겠습니다.

```
>>> def sum(a,b):
    return a+b
```

예시를 풀이하자면 아래와 같습니다.

"이 함수의 이름(함수명)은 sum이고 입력 인수로 2개의 값을 받으며 결과값은 2개의 입력값을 더한 값이다."

여기서 `def` 는 함수를 정의하기 위한 예약어이고, `return` 은 반환값을 위한 예약어입니다. 그럼 위의 예시를 일반화 시키면 아래와 같습니다.

```
def 함수이름(입력인수):
    <수행할 문장>
    ...
    return 결과값
```

앞으로 사용자 정의 함수를 사용할때 위와 같은 형태로 선언하고 사용하게 될 것입니다. 여기서 또 한가지...! **주목할 점** 이 있습니다. 파이썬은 *입력인수값*이 꼭 필요하지도, *반환값*이 꼭 필요하지도 않습니다. 따라서, *입력인수*가 아예 없을 수도 있고, *반환값*이 없는 void형일 수도 있습니다.

위의 `sum` 함수를 사용해보면 풀이대로 입력받은 두 인수의 값을 합을 반환합니다.

```
>>> a=3
      b=4
      c=sum(a,b)
      print(c)
```

7

또다른 함수 정의 방법은 `lambda` 를 이용하는 것 입니다. `lambda` 는 함수를 생성할 때 사용하는 예약어로, `def`와 동일한 역할을 합니다. 보통 함수를 한줄로 간결하게 만들 때 사용하는데... 우리말로는 "람다"라고 읽고, `def` 를 사용해야 할 정도로 복잡하지 않거나 `def`를 사용할 수 없는 곳에 주로 씁니다. 사용법은 아래 예시와 같습니다.

```
>>> myfunction=lambda a,b: a+b
>>> myfunction (2,3)
```

5

파일 읽고 쓰기

파일 읽기(Read)

이번에는 파일을 읽어와서 출력하는 작업을 해보겠습니다. 아래 예시를 통해 설명을 이어 나가겠습니다...

```
>>> f=open("/파일경로../파일명", "r")
lines = f.readlines()
for line in lines:
    print(line)
f.close
```

우선, 위의 예시를 풀이하면 아래 내용과 같습니다.

"/파일경로"에 있는 파일을 "r" (읽기) 하여 f 라는 파일 변수에 할당하였고, f 에 담겨 있는 파일을 한줄씩 읽어서 lines 라는 변수에 담았다. 반복문을 통하여 lines 에 담겨 있는 파일의 각 라인을 한줄씩 출력하고 반복문이 끝나면 사용한 f 라는 파일 변수는 닫겠다.

따라서 어떤 파일을 사용하기 위해 `open` 이라는 함수를 통해 파일을 열고, `readlines()` 라는 함수를 통해 파일의 내용을 한줄씩

읽어서 리스트화 시킨 뒤에 리스트 변수에 담긴 내용을 반복문을 통해 출력하면 간단히 파일을 읽어 볼 수 있는 작업을 할 수 있게 됩니다.

위의 예시처럼 파일작업을 하고 나면 항상 `.close` 를 해주어야 합니다. 하지만 매번 이렇게 선언해주는 것도 귀찮고 불편하다고 생각될 수 있습니다. 이럴때 자동으로 파일작업이 끝나면 `.close` 를 호출해주는 `with` 문을 이용하면 됩니다. 위의 예시를 `with` 문을 활용한 코드로 변환하면 아래와 같습니다.

```
>>> with open("/파일경로.../파일명", "r") as f:
    lines= f.readlines()
    for line in lines:
        print(line)
```

파일 쓰기

이번에는 파일쓰기를 해보겠습니다. 파일쓰기도 읽기와 마찬가지로 `with` 문을 활용하여 아래 예시를 작성해보았습니다.

```
>>> with open("/파일경로.../파일명", "w") as f:
    f.write("You are happy")
```

위의 예시대로라면 `"/파일경로.../"` 에 `"파일명"` 을 가진 파일이 생성 되었을 겁니다. 그리고 그 파일의 내용은 `You are happy` 가 있을 겁니다. 파일 쓰기와 읽기 작업과 차이점은 `"r"` 대신 `"w"` 가 쓰였다는 점과 쓰기 작업을 하려는 문장을 `.write()` 를 통해 처리하였다는 것입니다.