

Fastcampus Data Science Extension SCHOOL

Python

Index

- Let's Code Pythonic
- Syntax
- Data types, Operators
- input
- type casting
- string
- list, tuple

Let's Code PYTHONIC

Important Python Enhance Proposal

Layout

- 들여쓰기: 공백 4칸 or 탭(섞어쓰면 안됨)
- 한 줄은 79자(120자도 상관없음)
- 클래스정의와 최상위 함수는 두 줄을 띄움
- 클래스 내 메소드는 한 줄을 띄움

Important Python Enhance Proposal

Variables

- `_variable` : 내부적으로 사용되는 변수
- `print_` : 파이썬 키워드와 충돌 방지

Naming Convention

- 클래스 이름은 `CamelCase`
- 함수, 변수, 메소드 이름은 `snake_case`

파이썬에서 쓰이지 않는 네이밍 규칙

- `chHungarianNotation`
- `javaScriptStyleCamelCase`

Syntax

문법, 구조, 또는 언어 문장 내에 있는 구성요소의 순서

"나는 입니다 학생" (Syntax Error)

"나는 학생 입니다" (Syntactically Valid)

"Python"5 (Syntax Error)

3.6 * 12 (Syntactically Valid)

Data type

- int
- float
- long(0b, 0o, 0x)
- string
- boolean
- list, tuple, range
- set
- dictionary

operators

- arithmetic
 - `+, -, *, /, %, //, **`
- boolean
 - `and, or, not`
 - <https://docs.python.org/3/reference/expressions.html#boolean-operations>
- comparisons
 - `<, >, <=, >=`
 - `==, !=(value)`
 - `is, is not(object identity)`
 - `&, |, ~(bitwise) => 1000(8), 1010(10)`

Floating point

- 컴퓨터는 실수를 이진 부동소수점을 활용합니다.
- 0, 1!!
- $0.1 + 0.2 \neq 0.3$
-> `round(0.1+0.2, 1) == round(0.3, 1)`

type casting

`float(3) --> int to float`

`int(3.6) --> float to int`

`str(1) --> int to string`

`int("12") --> string to int`

input

```
name = input("What is your name? ")  
print("Hi, ", name)
```

input with evaluation

```
input("How old are you? ")  
eval(input("How old are you? "))
```

type casting with input

```
int(input("How old are you? "))
```

Small Project Again!

사용자의 입력을 받아 반지름(r)을 선언한 뒤, 이를 이용하여 원의 지름, 둘레, 넓이, 구의 겉넓이, 부피를 각각 출력하는 파이썬 파일을 만들어보세요.
($\pi=3.1415$)

Strings

Strings

```
some_string = "python"  
len(some_string)
```

- index

p	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

```
some_string[3:5] = "ho"  
some_string[1:5:2] = "yh"  
some_string[::] = some_string[0:len(some_string):1]  
some_string[::-1] = some_string[-1:-len(some_string):-1]  
some_string[::-1] = "nohtyp"
```


but, strings are immutable

```
>>> some_string = "python"

>>> some_string[0] = "c"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment

>>> some_string = "c" + some_string[1:]
```

String Functions

```
func = "python is easy programming language"  
func.count('p')
```

```
func.find('p')
```

```
comma = ","  
func = comma.join('python')
```

```
func.split(',')
```

```
python_is_easy = "python is easy"  
python_is_easy.split()
```

```
python_is_easy.replace("python", "golang")
```

String Functions

```
some_string = "    computer    "  
some_string.strip()
```

```
some_string = ",,,Fastcampus..."  
some_string.strip(",")  
some_string.strip(".")
```

String Formatting - old way

```
print("I have a %s, I have an %s." % ("pen", "apple"))
```

- %s - string
- %c - character
- %d - Integer(decimal)
- %f - floating-point
- %o - 8진수(Octal)
- %x - 16진수(hexadecimal)
- %% - %

String Formatting - New way

```
print("I have a {}, I have an {}".format("pen", "apple"))
```

```
print("I have a {0}, I have an {1}".format("pen", "apple"))
```

```
print("I have a {0}, I have an {0}".format("pen", "apple"))
```

padding and align

- `{:10}`
- `{:>10}`
- `{:^10}`
- `{:_^10}`

Toggl

<https://blog.toggl.com/wp-content/uploads/2016/12/toggl-it-jobs-explained-with-changing-lightbulb.jpg>

<https://assets.toggl.com/images/toggl-how-to-save-the-princess-in-8-programming-languages.jpg>

List, Tuple

List

```
animals = [' ', ' ', ' ']
```

Tuple

```
animals = (' ', ' ', ' ')
```


List

빈 list를 선언합니다. 선언과 동시에 값을 채워넣을 수 있습니다.

```
lang = ["python", "c", "java", "golang"]
```

```
lang = []
```

list에 요소를 추가합니다.

```
lang.append("python")
```

```
lang.append("java")
```

```
lang.append("golang")
```

```
print(lang)
```

혹은 특정한 위치에 원하는 값을 추가할 수 있습니다.

```
lang.insert(1, "c")  
print(lang)
```

특정 요소를 삭제할 수도 있습니다.

```
lang.remove("golang")  
print(lang)
```

혹은 리스트에 있던 값을 빼낼 수도 있습니다.

```
java = lang.pop(2)  
print(lang)  
print(java)
```

리스트를 정렬하는 법을 알아보니다.

```
numbers = [2, 1, 4, 3]
```

```
print(numbers)
```

```
numbers.sort()
```

```
print(numbers)
```

리스트를 역순으로 출력하고 싶을땐 이렇게 한답니다.

```
numbers = [2, 1, 4, 3]
```

```
numbers.reverse()
```

```
print(numbers)
```

리스트를 내림차순으로 정렬하려면??

1. sort -> reverse

```
numbers.sort()  
numbers.reverse()
```

2. sort(reverse=True)

```
numbers.sort(reverse=True)
```

특정 값의 위치를 출력할때 이렇게 합니다.

```
index_of_two = numbers.index(2)  
print(index_of_two)
```

리스트끼리 더할 때 extend를 활용합니다.

```
numbers += [5, 6]  
print(numbers)  
numbers.extend([7, 8])  
print(numbers)
```

Tuple

Tuple은 괄호를 이용해 선언할 수 있습니다.

```
tuple1 = (1, 2, 3, 4)
```

tuple은 삭제나 추가가 불가능합니다.

```
del tuple[1]  
tuple1[1] = 'c'
```

tuple끼리 더하거나 반복하는 것은 가능합니다.

```
tuple2 = (5, 6)  
print(tuple1 + tuple2)  
  
print(tuple1 * 3)
```

tuple은 값을 편하게 바꿀 수 있습니다.

```
x = y
y = x (x)

temp = x
x = y
y = temp

(x,y) = (y,x)
```

혹은 함수에서 하나 이상의 값을 반환할 때 사용합니다.

```
def quot_and_rem(x,y):
    quot = x // y
    rem = x % y
    return (quot, rem)

(quot, rem) = quot_and_rem(3,10)
```


List <-> Tuple

```
list((1,2))  
tuple([1,2])
```