

# 1. Python 시작하기

Spring 2018

## Python 기본 자료형

- 숫자형

1. 정수 int
2. 부동소수점 float
3. 복소수, 8진수, 16진수

변수의 타입을 확인하고 싶을 때는?  
: `type(변수명)` 으로 확인 가능

- 문자열 str

1. “...” 큰따옴표로 둘러싸거나, ‘...’ 작은따옴표로 둘러싸인 문자들의 집합
2. 큰따옴표 3개 또는 작은따옴표 3개로 둘러싸인 문자열은? Multi-line
3. 문자열은 문자 하나하나가 연결된 집합이라고 생각할 수 있음

- 이스케이프 코드

1. 줄바꿈 \n
2. 탭 \t
3. 역슬래시 \
4. 작은따옴표 \' 큰따옴표 \"

주석 (comments) 을 쓸 때는?  
: single-line은 줄앞에 #  
multi-line은 앞뒤에 따옴표 3개를

## Python 기본 자료형

- Boolean (bool) : True or False

변수의 타입을 바꾸고 싶을 때는? (casting)

int(x), float(x), str(x), bool(x) 타입을 앞에 써준다

- 리스트 list

1. [a,b,c...]
2. Mutable: 리스트의 원소를 추가하거나 바꾸거나 제거할 수 있음

- 튜플 tuple

1. (a,b,c,...)
2. Immutable: 튜플의 원소를 추가, 교체, 제거할 수 없음

- 딕셔너리 dict

1. {key1: value1, key2: value2, ... }

- 세트 set

1. {a,b,c,...}

## Python 기본 연산자

### ■ 숫자형 연산

1. 사칙연산자  $+$ ,  $-$ ,  $*$ ,  $/$
2. Power 연산자  $**$
3. 나머지 연산자  $\%$
4. 몫 연산자  $//$

0	1	2	3	4	5
H	E	L	L	O	!

### ■ 문자열 연산

1. 문자열 연결  $+$
2. 문자열 반복  $*$

0      1      2      3      4      5      6  
-6    -5    -4    -3    -2    -1

0	1	2	3	4	5
H	E	L	L	O	!

string[2:4]

0	1	2	3	4	5
H	E	L	L	O	!

string[-5:-2]

### ■ 문자열 인덱싱과 슬라이싱

1. 문자열 중의 문자 하나 또는 일부를 선택하는 방법
2. 문자열 뿐만 아니라 리스트(list), 튜플(tuple) 등의 sequence 데이터형에서 동일

## 문자열 formatting

- 문자열 중에 경우에 따라 바뀌는 값이 있을 때  
“사람 다리는 2개” “오징어 다리는 10개” “고양이 다리는 4개”  
이런 문자열을 손쉽게 만드는 방법은?

x = “%s 다리는 %d개”

x % (“사람”, 2)

x % (“오징어”, 10)

x % (“고양이”, 4)

- 문자열 포맷 코드
  1. %s: 문자열
  2. %d: 정수
  3. %f: 부동소수점 (%0.4f : 소수점 4자리까지 표시)
  4. %%: “%”라는 문자

## 고급 formatting

- format 함수

x = "{0} 다리는 {1}개"

x.format("사람", 2)

```
r = "{0} {1} {2}".format('GOOG', 100, 490.10)
r = "{name} {shares} {price}".format(name='GOOG', shares=100, price=490.10)
r = "Hello {0}, your age is {age}".format("Elwood", age=47)
r = "Use {{ and }} to output single curly braces".format()
```

```
name = "Elwood"
r = "{0:<10}".format(name)      # r = 'Elwood      '
r = "{0:>10}".format(name)      # r = '      Elwood'
r = "{0:^10}".format(name)     # r = '   Elwood   '
r = "{0:=^10}".format(name)    # r = '==Elwood=='
```

```
y = 3.1415926
r = '{0:10.2f}'.format(y)      # r = '          3.14'
r = '{0:10.2e}'.format(y)      # r = '    3.14e+00'
r = '{0:+10.2f}'.format(y)     # r = '        +3.14'
r = '{0:+010.2f}'.format(y)    # r = '+000003.14'
r = '{0:+10.2%}'.format(y)     # r = '    +314.16%'
```

## 문자열 관련 함수

- `count(x)`: 해당 문자 `x`의 개수
- `find(x)` 또는 `index(x)` : 차이점은 `x`가 없을 때 `-1`을 반환하는지, 에러가 나는지.
- `upper(x)`, `lower(x)` : 문자열을 대문자로 또는 소문자로 변환
- `strip(x)`, `lstrip(x)`, `rstrip(x)`: 문자열 앞뒤의 공백 제거
- `split(x)` : `x`를 기준으로 문자열을 나눠서 리스트로 만들

## 리스트 관련

- 다중리스트: 리스트 안에 다시 리스트
- `append(x)` : 리스트에 원소를 추가
- `extend(x)` : 리스트에 리스트를 연결 (x는 반드시 리스트)
- `sort()` : 정렬
- `reverse()` : 역순으로 뒤집기
- `index(x)` : x의 위치를 반환
- `insert(x, y)` : 인덱스 x 의 위치에 y를 삽입
- `remove(x)` : 원소 x를 제거
- `pop()` : 마지막 원소 제거



## 딕셔너리 관련

- `keys()` : key값들을 반환
- `values()` : value값들을 반환
- `items()` : (key,value)의 순서쌍을 반환

## 세트 관련

- `a & b` 또는 `a.intersection(b)`
- `a | b` 또는 `a.union(b)`
- `a - b` 또는 `a.difference(b)`
- `add(x)` : 원소 x 추가
- `update(x)` : set x의 원소들을 한번에 추가
- `remove(x)` : x 제거

특정원소가 list, tuple, dict, set에 있는지 확인할 때  
: `in` 을 사용함 (`x in a` : 원소 x가 a에 있으면 True)

## Indexing / Slicing

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

b = a[::2]           # b = [0, 2, 4, 6, 8 ]
c = a[::-2]          # c = [9, 7, 5, 3, 1 ]
d = a[0:4:2]          # d = [0, 2]
e = a[5:0:-2]         # e = [5, 3, 1]
f = a[:5:1]           # f = [0, 1, 2, 3, 4]
g = a[:5:-1]          # g = [9, 8, 7, 6]
h = a[5::1]           # h = [5, 6, 7, 8, 9]
i = a[5::-1]          # i = [5, 4, 3, 2, 1, 0]
j = a[5:0:-1]         # j = [5, 4, 3, 2, 1]
```

## 연산자

Operation	Description
<code>a + b</code>	Add a and b
<code>a - b</code>	Subtract b from a
<code>a * b</code>	Multiply a by b
<code>a / b</code>	Divide a by b
<code>a // b</code>	Floor-divide a by b, dropping any fractional remainder
<code>a ** b</code>	Raise a to the b power
<code>a &amp; b</code>	True if both a and b are True. For integers, take the bitwise AND.
<code>a   b</code>	True if either a or b is True. For integers, take the bitwise OR.
<code>a ^ b</code>	For booleans, True if a or b is True, but not both. For integers, take the bitwise EXCLUSIVE-OR.
<code>a == b</code>	True if a equals b
<code>a != b</code>	True if a is not equal to b
<code>a &lt;= b, a &lt; b</code>	True if a is less than (less than or equal) to b
<code>a &gt; b, a &gt;= b</code>	True if a is greater than (greater than or equal) to b
<code>a is b</code>	True if a and b reference same Python object
<code>a is not b</code>	True if a and b reference different Python objects

## Sequence 연산자

### Operation

`s + r`

`s * n, n * s`

`v1, v2..., vn = s`

`s[i]`

`s[i:j]`

`s[i:j:stride]`

`x in s, x not in s`

`for x in s:`

`all(s)`

`any(s)`

`len(s)`

`min(s)`

`max(s)`

`sum(s [, initial])`

### Description

Concatenation

Makes *n* copies of *s*, where *n* is an integer

Variable unpacking

Indexing

Slicing

Extended slicing

Membership

Iteration

Returns True if all items in *s* are true.

Returns True if any item in *s* is true.

Length

Minimum item in *s*

Maximum item in *s*

Sum of items with an optional initial value

## 모듈

- `import math`
- `from math import log`

## 파이썬 Standard Library

- 파이썬에 내장되어 있는 라이브러리 패키지
- 대표적으로 `math`, `sys`, `os`, `datetime`, `random` 등
- 기타 수많은 패키지들은 별도로 설치 (`pip` 등 이용)

## Math

- `e`, `pi` : 수학 상수
- `log`
- `exp`
- `sqrt`
- `pow`
- `sin`, `cos`, `tan` 등등

## datetime

- 날짜, 시간을 다루는 모듈

```
from datetime import datetime
```

```
dt = datetime(2018, 2, 6, 20, 15, 20) #연,월,일,시,분,초
```

```
print(dt.strftime("%Y/%m/%d")) #2018/02/06 출력
```

```
dt2 = datetime.strptime('2018-10-20', '%Y-%m-%d')
```

```
delta = dt2 - dt
```

```
delta.days()
```

```
import dateutil.relativedelta as rd
```

```
d = rd.relativedelta(months=1, days=10)
```

```
dt3 = dt2 + d
```

## time

- 계산 시간 구하기

```
import time
```

```
t0 = time.time()
```

```
...
```

```
elapsed = time.time() - t0
```

## random

- 난수 생성
- choice(seq), choices(population, weights)
- sample(population, k)
- random()
- uniform(a,b)
- gauss(mu,sigma)

## 파일 읽기 / 쓰기

- `f = open(filename, mode)`
- `mode`

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, failing if the file already exists
'a'	open for writing, appending to the end of the file if it exists

- `f.write(str)` : str 쓰기
- `f.read()` : 전체 읽기
- `f.readline()` : 한줄 읽기
- `f.readlines()` : 리스트로 반환