



CHAPTER 02

파이썬 기초 Python Basic

컴퓨터소프트웨어공학과
김대영

- 파이썬 기초 문법
- 자료형
- 프로그램 제어

• 들여쓰기

- 파이썬은 중괄호와 같은 별도의 기호를 사용하지 않고 공백 문자를 이용하여 코드 블록을 지정함
- 콜론(:)은 코드 블록의 시작을 의미
- 코드 블록 내의 코드들은 반드시 들여쓰기로 구분되어야 함
- 세미콜론(;)은 한 줄에서 문장을 구분하기 위해 사용될 수 있지만 추천되지 않음

• 파이썬 코드 예

```
for x in array:  
    if x < pivot:  
        ret = value - 1  
    else:  
        ret = value + 1
```

```
a = 1; b = 2; c = 3
```

• 객체 (Object)

- 파이썬의 모든 것은 객체 모델로 구성됨
- 파이썬 객체: 숫자, 자료구조, 문자열, 함수, 클래스 등
- 변수에 저장되는 것은 객체의 주소 → 다양한 자료형 데이터 저장

• 주석 (Comment)

- 파이썬 코드의 주석을 위해 #을 사용함

```
print("perform this line.") #confirmed this line
```

• 함수(Function) 호출

- 함수는 입력 매개변수를 전달함으로써 호출하고, 함수의 결과로써 출력값을 반환함

```
result = func(x, y, z)  
Result2 = gFunc()
```

• 객체의 메소드(Method) 호출

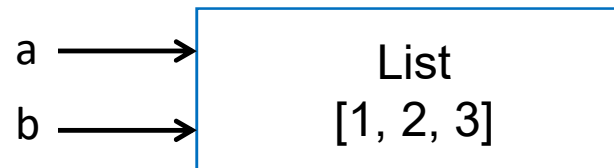
- 객체의 내부 함수인 메소드를 호출하여 객체 내부 데이터에 접근함

```
object.method(x, y, z)
```

• 변수(Variable)

- 대입 연산자 (=)를 사용하여 연산자 오른쪽 객체를 변수에 연결
- 변수에 객체를 할당하는 것은 객체에 참조를 생성하는 것과 같음
 - 바인딩(Binding): 객체를 변수의 이름에 연결

```
a = [1, 2, 3]  
b = a
```



- 동적 참조(Dynamic reference)

- 데이터 타입(자료형)은 객체 참조에서 중요한 문제가 아님
- 객체의 주소가 변수에 저장되기 때문에 다양한 자료형 사용가능
- 데이터가 사용되는 시점에서 묵시적 형변환 수행

```
num = 5
type(num)
> Int

ch = 'foo'
type(ch)
> str
```

• 자료형 확인

- 묵시적 형변환을 수행하는 객체의 자료형 확인
- *isinstance()* 함수를 사용하여 객체의 자료형 확인

```
a = 5  
isinstance(a, int)  
> True
```

• 속성(Attribute) & 메소드(Method)

- 속성: 객체 내부에 저장된 데이터 객체
- 메소드: 속성에 접근하기 위한 함수
- 속성과 메소드는 객체의 멤버접근 연산자(.)를 사용하여 접근
 - object.attribute_name / object.method_name

```
a = 'foo'
a.<Tab>
a.capitalize      a.format          a.isupper         a.rindex          a.strip
a.center           a.index           a.join            a.rjust           a.swapcase
a.count            a.isalnum         a.ljust           a.rpartition      a.title
a.decode           a.isalpha         a.lower           a.rsplitt         a.translate
a.encode           a.isdigit         a.lstrip          a.rstrip          a.upper
...
```

- getattr()를 사용하여 속성과 메소드 접근

```
getattr(a, 'split')
```

• 모듈 импорт(Module Import)

- my_module.py 에 존재하는 데이터와 함수를 사용하고자 할 때,

```
import my_module
result = my_module.func()    # 함수 사용
num = my_module.CON         # 데이터 사용
```

```
from my_module import func, CON
```

```
import my_module as mm
from my_module import CON as c, func as f

result = mm.func()
num = c
```

• 연산자(Operators)

<i>Operator</i>	<i>Desc.</i>
<code>a + b</code>	plus
<code>a - b</code>	minus
<code>a * b</code>	multiply
<code>a / b</code>	divide
<code>a // b</code>	quotient of <i>a</i> divided by <i>b</i>
<code>a ** b</code>	<i>a</i> to the <i>b</i> power (a^b)
<code>a & b</code>	AND
<code>a b</code>	OR
<code>a ^ b</code>	EXCLUSIVE OR
<code>a == b</code>	if <i>a</i> is <i>b</i> , TRUE
<code>a != b</code>	if <i>a</i> is not <i>b</i> , TRUE
<code>a < b, a <= b</code>	if <i>a</i> < <i>b</i> , TRUE; if <i>a</i> <= <i>b</i> , TRUE
<code>a > b, a >= b</code>	if <i>a</i> > <i>b</i> , TRUE; if <i>a</i> >= <i>b</i> , TRUE
<code>a is b</code>	if <i>a</i> references the same object with <i>b</i> , TRUE
<code>a is not b</code>	if <i>a</i> references a different object with <i>b</i> , TRUE

• mutable(Mutable) 객체

- 수정 및 변경이 가능한 객체를 의미
- 리스트, 사전, 배열, 사용자 정의 클래스 객체

```
my_list = ['foo', 1, [2,3]]
```

```
my_list[2] = (4,5)
```

```
my_list  
> ['foo', 1, (4,5)]
```

- **이뮤터블(Immutable) 객체**

- 수정 및 변경이 불가능한 객체를 의미
- 문자열, 튜플 객체

```
my_tuple = (3, 5, (4,5))
```

```
my_tuple[1] = 'four' → TypeError
```

- **int & float**

```
ival = 123456  
fval = 3.141592
```

```
3/2  
> 1.5
```

```
3//2  
> 1
```

- **string**

```
a = 'represent string I'  
b = "represent string II"  
  
c = ""  
    we can use several lines.  
    ""
```

- **string slicing**

- 슬라이싱(slicing): 파이썬 시퀀스 자료구조에 구현, 자료의 일부분을 분리

```
s = 'python'
list(s)
> ['p', 'y', 't', 'h', 'o', 'n']

s[:3]
> 'pyt'
```

- **string **

- 이스케이프 문자, 특수한 목적의 문자를 나타내기 위해 사용

```
s = '12\\34'  
print(s)  
> 12\34
```

- **string +**

- 문자열 결합

```
a = 'hello, '  
b = 'python'  
  
a + b  
> 'hello, python'
```

- 형식 문자열(Formatted string)

```
template = '{0:.2f} {1:s} are worth US${2:d}'  
template.format(4.5560, 'Argentine Pesos', 1)  
  
> '4.56 Argentine Pesos are worth US$1'
```

- {0:.2f} → 첫 번째 인자는 소수점 이하 둘째자리까지 표시한 부동소수점 데이터
- {1:s} → 두 번째 인자는 문자열
- {2:d} → 세 번째 인자는 정수

- **바이트(Bytes) & 유니코드(Unicode)**

- encode() 메소드: 문자열을 UTF-8 바이트로 변환
- decode() 메소드: UTF-8 바이트를 문자열로 변환

```
val = "Korean"  
val_utf8 = val.encode('utf-8')  
type(val_utf8)  
> bytes  
  
val_utf8.decode('utf-8')  
> 'Korean'
```

- **Boolean**

- True & False 사용

- **형변환**(Type casting)

- int, float, bool, str 형변환 함수

```
s = '3.14159'
fval = float(s)
type(fval)
> float

int(fval)
> 3

bool(fval)
> True

bool(0)
> False
```

- **None**

- 파이썬에서는 Null의 의미로 None을 사용

```
a = None
a is None
> True
b = 5
b is not None
> True
```

- 함수사용에서 반환값이 없는 경우 묵시적으로 None을 반환함

- 날짜(Date) & 시간(Time)

- datetime 모듈 사용
- datetime 함수로 생성된 객체에 date 메소드와 time 메소드를 사용하여 날짜와 시간 정보 추출

```
from datetime import datetime, date, time
dt = datetime(2020, 10, 29, 20, 30, 21)
dt.day
> 29
dt.minute
> 30
dt.strftime('%m/%d/%Y %H:%M')
> '10/29/2020 20:30'
```

- **if, elif, else 선택 구문**

- if 구문 : 주어진 조건이 참인 경우 블록 내 코드 수행
- elif 구문: if 구문의 조건이 거짓이면 다음 elif 구문 조건을 확인, elif 구문의 조건이 참인 경우 블록 내 코드 수행
- else 구문: if 구문과 elif 구문의 조건이 모두 거짓인 경우 else 구문의 블록 내 코드 수행

```
if x < 0:
    print("It's negative")
elif x == 0:
    print("Equal to zero")
elif 0 < x < 5:
    print("Positive but smaller than 5")
else:
    print("Positive and larger than or equal to 5")
```

- **if, elif, else 선택 구문**

- and, or 논리 연산을 사용하여 다중 조건 구현
- 다중 조건의 경우 왼쪽에서 오른쪽으로 조건 확인

```
if a < b or c > d:  
    print("good job")
```

```
if x < m and y > n:  
    print("good job")
```

- **for** 반복 구문

- 리스트와 같은 컬렉션 내 이터레이터를 순회
- 기본 구조

```
#collection: list, tuple, etc.  
for value in collection:  
    ...
```

- **for** 반복 구문 사용 예

- my_list 내 원소를 불러와 total 변수에 더함
- 리스트 내 원소값이 None 인 경우 다음으로 건너뛰

```
my_list = [1, 3, None, 7, None, 11]
total = 0
for value in my_list:
    if value is None:
        continue # continue를 만나면 다음 원소로 넘어감
    total += value
```

- **for** 반복 구문 사용 예

- my_list 내 원소가 5이면 더 이상 값을 더하지 않음

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8]
total = 0
for value in my_list:
    if value == 5:
        break # break를 만나면 반복을 종료
    total += value
```

- **range()** 함수

- 연속된 정수 값을 얻기 위한 이터레이터를 반환함

```
# end
for i in range(10):
    print(i)
```

```
> 0
  1
  2
  ...
  9
```

```
# start, end, step
for i in range(0, 10, 2):
    print(i)
```

```
> 0
  2
  4
  ...
  8
```

- **while** 반복 구문

- 주어진 조건이 참이면 반복 수행
- break를 사용하여 반복 구문을 벗어날 수 있음

```
# if condition is TRUE, repeat
x = 1
while x < 10:
    total += x
```

```
# if break is used, do not repeat
x = 1
while x < 100:
    total += x
    if total > 50:
        break
```

- **pass** 구문

- pass 구문은 아무것도 하지 않음을 나타냄

```
if x < 0:
    print("less position")
elif x == 0:
    # Do something HERE!!
    pass
else:
    print("greater position")
```

- 삼항 표현식(Ternary expression)

- 기본 구조

```
true-expression if condition else false-expression
```

- 사용 예

```
x = 10  
ret = "Small-number" if x <= 100 else "Big-number"  
print(ret)  
> Small-number
```

파이썬 실습

```
from datetime import datetime, date, time
import bisect
```

```
def main():
```

```
    a = [1, 2, 3]
    print(a)
    b = a
    print(b)
```

```
    num = 5
    print(type(num))
```

```
    ch = "foo"
    print(type(ch))
```

```
    a = 5
    print(isinstance(a, int))
```

```
    a = 2
    b = 5
    print(a*b)
    print(a**b)
    print(b//a)
```

```
    alist = ["foo", 2, [4, 5]]
    print(alist)
    alist[2] = (3, 4)
    print(alist)
```

```
    atuple = (3, 4, (4, 5))
    print(atuple)
    #atuple[1] = "four"
```

```
    ival = 12345
    fval = 3.1415
    print(3/2)
    print(3//2)
```

```
    s = "python"
    print(s)
    l = list(s)
    print(l)
    print(s[:3])
    print(l[:3])
```

```
    a = "hello,"
    b = " python"
    print(a + b)
```

```
    template = "{0:.2f} {1:s} are worth US$ {2:d}"
    t = template.format(4.5560, "Argentine Pesos", 1)
    print(t)
```

```
    val = "korean"
    val_utf8 = val.encode("utf-8")
    print(type(val_utf8))
    print(val_utf8.decode("utf-8"))
```

```
    s = "3.14159"
    fval = float(s)
    print(type(fval))
    print(type(int(fval)))
```

```
    s = None
    print(s is None)
```

파이썬 실습

```
dt = datetime(2021, 3, 10, 15, 11, 30)
print(dt.day)
print(dt.minute)
t = dt.strftime("%m/%d/%Y %H:%M")
print(t)
```

```
x = 10
if x < 0:
    print("negative")
elif x == 0:
    print("equal")
elif 0 < x < 5:
    print("positive but small")
else:
    print("positive and large")
```

```
a = 1; b = 2; c = 3; d = 4
if a < b or c > d:
    print("good job")
else:
    print("not good")
```

```
if a < b and c > d:
    print("good job")
else:
    print("not good")
```

```
sequence = [1, 3, None, 7, None, 11]
total = 0
for value in sequence:
    if value is None:
        continue
    total += value
print(total)
```

```
sequence = [1, 2, 3, 4, 5, 6, 7, 8]
total = 0
for value in sequence:
    if value == 5:
        break
    total += value
print(total)
```

```
for j in range(10):
    print(j)
```

```
for j in range(0, 10, 2):
    print(j)
```

```
x = 1
while x < 10:
    total += x
    x += 1
print(total)
```

```
x = 1
while x < 100:
    total += x
    if total > 50:
        break
    x += 1
print(total)
```

```
#-----
if x < 0:
    print("less position")
elif x == 1:    #<-
    # No operation (just place-holder)
    # use for non-implemented codes
    #To Do: wirte something~~~~
    pass
else:
    print("greater position")
#-----

x = 10
ret = "Small-number" if x <= 100 else "Big-number"
print(ret)

if __name__ == '__main__': # main()
    main()
```