kt ds University

파이썬 기초 요약

변수와 자료형

kt ds University

변수

- 객체(값)를 저장하는 메모리 공간
- 변수에 객체가 바인딩 됨

```
>>> x = 100
>>> id(x)
1773787088
```


```
>>> x = 10000
>>> y = 10000
>>> id(x), id(y)
(44212048, 44212048)
>>> y = 10001
>>> id(y)
48072352
```

파이썬에서 모든 자료 타입은 객체임

http://pythontutor.com/

리스트

하나의 변수에 여러 값을 할당하는 자료형

- 하나의 자료형으로만 저장하지 않고, 정수형이나 실수형 같은 다양한 자료형을 포함할 수 있음.
- C나 자바 같은 프로그래밍언어에서는 배열이라고 표현.

```
empty = []                          # 비어있는 리스트 선언
nums = [100, 200, 300]      # 숫자로 이루어진 리스트
colors = [‘red’, ‘blue’, ‘green’]  # 단어(문자)로 이루어진 리스트
items = [‘red’, 10, 21.5, ‘hello’] # 여러 자료형으로 이루어진 리스트
```

조건문과 반복문

조건문

- 조건에 따라 특정 동작을 하도록 하는 프로그래밍 명령어
- 반드시 조건의 참(True)과 거짓(False)으로 구분되어야 함

if-else 문

```
if <조건>:
   <수행명령>
else:
   <수행명령>
```

- if 뒤에는 조건문이 있어야 하며, **조건문이 끝나면 반드시 콜론(:)을 붙여야 함**
- 해당 조건이 참일 경우 수행할 명령을 **들여쓰기 후** 작성함 – 기존 다른 언어들과의 차이점
- else문은 생략 가능.

조건문과 반복문

반복문

- 정해진 동작을 반복적으로 수행할 때 내리는 명령어
- 반복 시작 조건, 종료 조건, 수행 명령으로 구성. 들여쓰기와 블록으로 구분.

for 문

- 1부터 5까지 출력하기

```
for i in [1, 2, 3, 4, 5]:
    print(i)
```

- 1부터 100까지 출력하기

```
for i in range(100):
    print(i)
```

```
for 변수 in range(시작번호, 마지막번호, 증가값)
```

시작번호와 증가값은 생략가능하며, 생략했을 경우 초기값으로 시작번호는 0, 증가값은 1 이다.

함수

장점

- 필요할 때 마다 호출 가능
- 논리적인 단위로 분할 가능
- 코드의 캡슐화
  입력값과 출력값을 명확히 하여 인터페이스가 잘 정의된 경우 코드 내부구조를 몰라도 쉽게 사용 가능

함수의 선언

```
def 함수 이름(매개변수 #1 …):
    수행문1
    수행문2
    return <반환값>
```

- 함수 이름은 소문자로 입력, 띄어쓰기 할 경우에는 _ 를 사용
- 함수의 시작과 끝을 명시하지 않음
- 동사와 명사를 함께 사용하는 경우가 많음 예) find_number
- 수행문은 반드시 들여쓰기한 후 코드를 입력해야 함

함수

함수의 인수(argument)

- 가변인수(variable-length arguments)

  함수의 매개변수가 정해지지 않은 채 진행해야 하는 경우 사용

  변수명 앞에 *를 붙임.- *를 붙임으로써 여러 개의 변수를 담는 컨테이너로서의 속성을 부여하게 됨

```
def asterisk_test(a, b, *args):
    return a + b + sum(args)

print(asterisk_test(1, 2, 3, 4, 5))
```

```
def asterisk_test(a, b, *args):
    print(args)

print(asterisk_test(1, 2, 3, 4, 5))
```

```
def asterisk_test(*args):
    x, y, *z = args              # 언패킹
    return x,y,z

print(asterisk_test(3, 4, 5, 10, 20))
```

함수의 인수(argument)

- 키워드 가변인수(keyword variable-length arguments)

  *를 2개 사용하여 함수의 매개변수를 표시

  키워드 가변인수는 모든 매개변수의 맨 마지막에 선언

```
def kwargs_test(a, b, *args, **kwargs):
     print(a + b + sum(args))
     print(kwargs)

kwargs_test(3,4,5,6,7,8,9, first=3, second=4, third=5)
```

클래스 구현하기

```
class SoccerPlayer(object):
```
[클래스 예약어] [클래스 이름] [상속받는 객체명]

```
class SoccerPlayer(object):
    def __init__(self, name, position, back_number):
        self.name = name
        self.position = position
        self.back_number = back_number
    def change_back_number(self, new_number):
        print("선수의 등 번호를 변경한다: From %d to %d" % (self.back_number, new_number))
        self.back_number = new_number
    def __str__(self):
        return "Hello, My name is %s. I play in %s in center." % (self.name, self.position)
```

모듈과 패키지

모듈(module)

- 작은 프로그램 조각 – 파이썬에서는 .py로 저장하는 파일 자체가 모듈
- 하나하나 연결해 어떤 목적을 가진 프로그램을 만드는 작은 프로그램

패키지(package)

- 모듈의 묶음
- 하나의 패키지 안에 여러 개의 모듈이 있고, 이 모듈들이 서로 포함 관계를 가지며 패키지를 형성

모듈과 패키지

네임스페이스(namespace)

- 모듈 호출의 범위를 지정
- 모듈 이름에 alias를 생성하여 모듈 안으로 코드를 호출하는 방법

```
import fah_converter as fah
print(fah.covert_c_to_f(41.6))
```

- from 구문을 사용하여 모듈에서 특정 함수 또는 클래스만 호출하는 방법

```
from fah_converter import covert_c_to_f
print(covert_c_to_f(41.6))
```

- 해당 모듈 안에 있는 모든 함수, 클래스, 변수를 가져오는 * 를 사용하는 것.

```
from fah_converter import *
print(covert_c_to_f(41.6))
```