Python 기초 문법(2)

1. IF 조건문

In [1]:
```
money = 2000
if money == 3000:
    print('버스타기')
else:
    print('택시타기')
```

택시타기

In [2]:
```
money = 1500
if money > 3000:  # 조건문
    # 참일 경우
    print('택시타기')
    print('카카오택시 콜')
else:
    # 거짓일 경우
    print('버스타기')
    print('걸어가기')
```

버스타기
걸어가기

> **Q. 위에서 배운 것을 이용해서 나이를 입력하였을때 10대, 20대, 30대, 40대, 50대를 맞추는 프로그램을 짜보자.**
> **(예 - 32를 입력하면 30대라고 출력됨)**

In [ ]:
```

```

2) 들여쓰기(Indentation)

- 들여쓰기는 함수 몸체, 조건문 루프, 클래스 등 다양한 코드 블록을 나타낸다.
- 들여쓰기로 탭(tab)을 사용할 수 있지만 권장하지 않는다.
- 스페이스(space) 4칸을 사용하는 것이 파이썬 커뮤니티에서 보편적으로 선호되고 권장하는 방식이다.
- Python을 처음으로 접하는 사용자들은 익숙하지 않아서 들여쓰기에서 많은 에러가 발생한다.

In [3]:
```
my_age = 15
if my_age >= 19:
    print("19살 이상이네요!")
    print("당신은 성인입니다!")
else:
    print("That's nono!")
    print("미성년자입니다!")
```

```
That's nono!
미성년자입니다!
```

In [4]:
```
# 들여쓰기가 제멋대로인 경우 에러 발생
are_u_sure = True
if are_u_sure:  # if 변수가 True(참)이면
   print("Starting with 2 spaces")  # 2칸
   print("That's nono!")  # 3칸
else:
    print("It's false.")  # 4칸
```

```
Starting with 2 spaces
That's nono!
```

> **Q. 아래 조건을 이용해서 내 신용등급을 출력해보자. (위에 나온 if 조건문을 이용)**
>
> 100 ~ 90: 1
>
> 89 ~ 80: 2
>
> 79 ~ 70: 3

> 69 ~ 0: 4

In [ ]:

> **Q. 지금까지 배운 반복문, 조건문을 이용해서 아래 문제를 풀어보세요.**
>
> - 놀이공원 입장비를 계산해보세요!
> - 입장비는 나이에 따라 다릅니다.
> - 사용자로부터 입장할 사람의 수를 무한대로 받을 수 있습니다.
> - 사용자가 엔터를 입력하면 끝나고 지금까지 구한 입장비의 총합을 구해서 출력합니다.

알고리즘 (개발 순서)

1. 먼저 고객의 나이를 받아 변수에 할당해둔다.
2. 해당 변수의 값이 빈값이 아닌동안 아래 내용을 계속 실행한다.
3. 조건문을 이용해서 해당 변수의 값이 연령대에 따라 total_price에 합산될 수 있도록 분기처리한다.
4. 다 합산되었으면 다시 사용자로부터 값을 입력받아 해당 변수에 할당한다.

In [5]:
```
# 가격
BABY_PRICE = 1000  # 유아
CHILD_PRICE = 2000  # 청소년
ADULT_PRICE = 3000  # 성인
```

In [6]:
```
# 연령대
BABY_LIMIT = 2  # 2세 이하는 1000원
CHILD_LIMIT = 18  # 18세 이하는 2000원
ADULT_LIMIT = 60  # 60세 이하는 3000원, 60세 이후부터 무료
```

In [7]:
```
total_price = 0
age = input('고객의 나이를 입력하세요. (엔터는 종료)')  # 엔터는 값이 ''

# 여기부터 작성하세요.
```

In [8]:
```
# solution
```

Dictionary

- 모든 항목이 “키(key)”-“값(value)” 로 구성되는 데이터 형식이다.
- 사전형식 데이터를 만들 때는 { }를 사용한다. 키 값을 인자로 하면 [ ]를 사용하면 해당 키의 값을 얻을 수 있다.
- dic = {'name' : 'Kyung', 'phone' : 250-6133, 'birth' :800913}
- 여기서 Key는 name, phone, birth이고, 각각의 key에 해당하는 value는 kyung, 20-6133, 800913이다

In [9]:
```
a = {1:'hi'}
```

In [10]:
```
a = {'a':[1,2,3]}
```

In [11]:
```
dic = {}
dic['물리학과'] = '물리학의 각 분야에 걸친 이론과 응용방법을 심오하게 교수, 연구함으로써 독창적 능력을 함양하고 고도 산
dic['우주과학과'] = '우주과학과는 천체 및 우주에서 일어나는 제반 현상을 과학적으로 탐사하고 연구하는 학과이다. 본 학과
dic['우주탐사학과'] = '경희대학교 우주탐사학과(School of Space Research /KHU)는 교육과학기술부 제 1유형의 세계수

print(dic['물리학과'])

print(dic['우주과학과'])

print(dic['우주탐사학과'])
```

물리학의 각 분야에 걸친 이론과 응용방법을 심오하게 교수, 연구함으로써 독창적 능력을 함양하고 고도 산업사회를 선도해 갈 지도적 인재를 양성함을 목적으로 한다.
우주과학과는 천체 및 우주에서 일어나는 제반 현상을 과학적으로 탐사하고 연구하는 학과이다. 본 학과는 인류의 우주진출이 더욱 활발해 지고 있는 이 시대에 그를 위한 지식과 기술의 개발과 보급을 목적으로 설립되었다. 현대 천문학에서부터 인공위성과 우주선의 활용에 이르는 기초와 응용의 병행 학습을 통하여 21세기 우주 시대가 요구하는 첨단분야에서 국제적인 경쟁력이 있는 인재를 양성하는 데에 우주과학과의 교육 목적이 있다.
경희대학교 우주탐사학과(School of Space Research /KHU)는 교육과학기술부 제 1유형의 세계수준의 연구중심 대학 육성(WCU)사업에 달궈도 우주 탐사 연구 과제가 선정됨에 따라 설립된 대학원 학과로서 우리나라의 우주탐사를 위하여 본격적으로 전문 인력 양성의 기틀을 마련하고자 한다. 한국 정부의 대학 교육 지원 과정의 세계 수준의 연구중심 대학(WCU) 육성사업을 통해 연구 역량이 높은 우수 해외 학자를 유치 활용하여, 국내 대학과 협력하여 핵심 성장 동력을 창출할 수 있는 분야의 연구를 활성화하는데 그 목적이 있다. 다양한 프로젝트를 통해 국가적 발전을 선도하는 신 성장 동력을 창출하는 기술을 개발하는데 집중하고 있으며, 기초과학, 인문과학, 사회과학의 학제적 통합을 통해 학계 및 사회, 국가적 발전에 기여 할 수 있도록 정부에서 적극적으로 추진하고 있다.

In [12]:
```
for item in dic.keys():
    print(item)

for item in dic.values():
    print(item)

'철학과' in dic.keys()
```

```
물리학과
우주과학과
우주탐사학과
물리학의 각 분야에 걸친 이론과 응용방법을 심오하게 교수, 연구함으로써 독창적 능력을 함양하고 고도 산업사회를 선도해 갈 지도적 인재를 양성함을 목적으로 한다.
우주과학과는 천체 및 우주에서 일어나는 제반 현상을 과학적으로 탐사하고 연구하는 학과이다. 본 학과는 인류의 우주진출이 더욱 활발해 지고 있는 이 시대에 그를 위한 지식과 기술의 개발과 보급을 목적으로 설립되었다. 현대 천문학에서부터 인공위성과 우주선의 활용에 이르는 기초와 응용의 병행 학습을 통하여 21세기 우주 시대가 요구하는 첨단분야에서 국제적인 경쟁력이 있는 인재를 양성하는 데에 우주과학과의 교육 목적이 있다.
경희대학교 우주탐사학과(School of Space Research /KHU)는 교육과학기술부 제 1유형의 세계수준의 연구중심 대학 육성(WCU)사업에 달궈도 우주 탐사 연구 과제가 선정됨에 따라 설립된 대학원 학과로서 우리나라의 우주탐사를 위하여 본격적으로 전문 인력 양성의 기틀을 마련하고자 한다. 한국 정부의 대학 교육 지원 과정의 세계 수준의 연구중심 대학(WCU) 육성사업을 통해 연구 역량이 높은 우수 해외 학자를 유치 활용하여, 국내 대학과 협력하여 핵심 성장 동력을 창출할 수 있는 분야의 연구를 활성화하는데 그 목적이 있다. 다양한 프로젝트를 통해 국가적 발전을 선도하는 신 성장 동력을 창출하는 기술을 개발하는데 집중하고 있으며, 기초과학, 인문과학, 사회과학의 학제적 통합을 통해 학계 및 사회, 국가적 발전에 기여 할 수 있도록 정부에서 적극적으로 추진하고 있다.
```

Out[12]:
```
False
```

In [13]:
```
'철학과' in dic
```

Out[13]:
```
False
```

In [14]:
```
'철학과' in dic.keys()
```

Out[14]:
```
False
```

In [15]:
```
'우주탐사학과' in dic
```

Out[15]:
```
True
```

In [16]:
```
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
```

```
cute
True
```

In [17]:
```
d['fish'] = 'wet'    # Set an entry in a dictionary
print(d)
print(d['fish'] )     # Prints "wet"
```

```
{'cat': 'cute', 'dog': 'furry', 'fish': 'wet'}
wet
```

In [18]:
```
print(d.get('monkey', 'N/A')) # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
```

```
N/A
wet
```

In [19]:
```
del d['fish']        # Remove an element from a dictionary
print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"
```

```
N/A
```

In [20]:
```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    print(animal)
```

```
person
cat
spider
```

In [21]:
```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))
```

```
A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

In [22]:
```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
```

```
A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

튜플

- 튜플은 리스트와 거의 같은 용도로 사용되나 내용을 갱신할 수 없다는 제약을 갖는다. 즉, 상수화된 리스트라고 볼 수 있다.
- 이렇게 상수화된 리스트를 사용하는 이유는 더 이상 내용을 변경하는 것을 방지하는 목적도 있고, 처리속도를 빠르게 한다는 장점도 있다.
- 리스트를 만들때와 달리 튜플을 만들려면 ( )을 사용하거나 아무 괄호를 넣지 않아도 된다.

In [23]:
```
t1 = (1, 3, 5)
t2 = 2, 4, 6

print(type(t1))
print(type(t2))

print(t1)
print(t2)
```

```
<class 'tuple'>
<class 'tuple'>
(1, 3, 5)
(2, 4, 6)
```

In [24]:
```
t = (1, 2, 3)
print(t)
l = list(t)
print(l)
t2 = tuple(l)
print(t2)
```

```
(1, 2, 3)
[1, 2, 3]
(1, 2, 3)
```

함수

- 목적: 코드 재사용과 반복작업을 최소화 하기 위해

1) 만드는 방법

- 함수는 `def` 로 정의한다.

In [25]:
```
def sum(x, y, a):
    return x + y + a
```

In [26]:
```
print(sum(1, 2, 3))
```

```
6
```

In [27]:
```
def add(x, y):
    print(x + y)
```

In [28]:
```
add(3, 6)
```

```
9
```

In [29]:
```
def sum(a,b):
    print("%d와 %d의 합은 %d입니다." % (a, b, a+b))
```

In [30]:
```
sum (1,3)
```

```
1와 3의 합은 4입니다.
```

In [31]:
```
#def => define
def multiply(a,b):
    c=a*b
    return c

print(multiply(3,4))
print(multiply(4,6))
```

```
12
24
```

In [32]:
```
def multiply(a,b,c):
    f=a*b*c
    return f

print(multiply(1,2,3))
```

```
6
```

In [33]:
```
def sign(n):
    if n > 0:
        return '양수'
    elif n <0:
        return '음수'
    else:
        return '0'

for n in range(-3,3):
    print(sign(n))

# for n in [-1,0,1]:
#     print(sign(n))
```

```
음수
음수
음수
0
양수
양수
```

In [34]:
```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'
```

```
for x in [-1, 0, 1]:
    print(sign(x))
```

```
negative
zero
positive
```

In [35]:
```
def hello(name, loud=False):
    if loud:
        print('HELLO, %s' % name.upper())
    else:
        print('Hello, %s!' % name)

hello('Bob')
hello('Fred', loud=True)
```

```
Hello, Bob!
HELLO, FRED
```

In [36]:
```
seed = 3

def updown(mind, guess):
    if mind < guess:
        return 'down'
    elif mind > guess:
        return 'up'
    else:
        return 'correct'

updown(seed, 3)
```

Out[36]:
```
'correct'
```

> **Q. 3개의 인자값을 받아 평균을 반환하는 함수를 만들어보세요.**

In [37]:
```
# 여기에 코드를 작성해보세요.
```

In [ ]:
```

```

> **Q. 미국에 있는 주(State)의 수도는 무엇인지 출력하는 함수 `find_capital` 를 만들어보세요.**

In [38]:
```
# 미국의 주의 수도를 찾는 함수를 만드세요.
STATES_CAPITALS = {
    'Alabama': 'Montgomery',
    'Alaska': 'Juneau',
    'Arizona': 'Phoenix',
    'Wyoming': 'Cheyenne',
}

# 여기에 코드를 작성하세요.
```

In [39]:
```
# 미국의 주의 수도를 찾는 함수를 만드세요.
```