강원대학교
AI 소프트웨어학과

인공지능
- 제어문과 함수 -

KNU 강원대학교

제어문

- 프로그램의 규칙을 만들어주는 문장
- If 문 : 특정 조건에 의해 이루어지는 문장
- for 문 : 특정 범위에 대해서 반복하는 문장
- While 문 : 특정 조건에 대해서 반복하는 문장

구분		사용방법
If 문	조건문	If 조건: 실행할 명령1 실행할 명령2 else: 실행할 명령3
for 문	반복문	For I range(반복횟수): 실행할 명령1 실행할 명령2
While 문	반복문	While 조건: 실행할 명령1 실행할 명령2

제어문

- 프로그램의 규칙을 만들어주는 문장
- for 문 : 특정 범위에 대해서 반복하는 문장
- If 문 : 특정 조건에 의해 이루어지는 문장
- While 문 : 특정 조건에 대해서 반복하는 문장

for 문

- for 문 : 특정 범위에 대해서 반복하는 문장

```
for 변수 in [문자열, 리스트,...]
    특정 명령1
    특정 명령2
    .
    .
    .
```

a 부터 b-1까지에 해당되는 값들을 하나씩 뽑아서 i에 저장하고 저장된 i를 특정명령을 작동할 수 있게 만들어라

for 문

- for 문 : 특정 범위에 대해서 반복하는 문장

```
for i in range(a, b):
    특정 명령1
    특정 명령2
        .
        .
        .
```

a 부터 b-1까지에 해당되는 값들을 하나씩 뽑아서 i에 저장하고 저장된 i를 특정명령을 작동할 수 있게 만들어라

for 문

- for 문 : 특정 범위에 대해서 반복하는 문장

```
for i in range(1, 10):
    print(i)
```

a 부터 b-1까지에 해당되는 값들을 하나씩 뽑아서 i에 저장하고 저장된 i를 특정명령을 작동할 수 있게 만들어라

```
a=range(1, 10)
list(a)
```

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

for 문

- for 문 : 특정 범위에 대해서 반복하는 문장

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

range에 범위를 지정하지 않고 하나의 값만 넣으면
0 부터 지정 값 전까지를 나타냄

```
a=range(10)
list(a)
```

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

for 문

- **for 문 : 특정 범위에 대해서 반복하는 문장**

```
for i in [1,2,3,4,5,6,7,8,9]:
    print(i)
```

Tab

```
a=range(1, 10)
list(a)
```

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

for 문

- for 문 : 특정 범위에 대해서 반복하는 문장

```
for i in [1,2,3,4,5,6,7,8,9]:
    for j in [1,2,3,4,5,6,7,8,9]:
        print(i*j)
```

첫번째 Tab으로 넘어가고 두번째 탭으로 갈때 두번째 탭의 모든 것을
끝내고 다시 첫번째 Tab으로 넘어 감

for 문

1단 부터 9단까지 구구단을 만드시오

```
--------1단---------
구구단 : 1X1=1입니다.
구구단 : 1X2=2입니다.
구구단 : 1X3=3입니다.
구구단 : 1X4=4입니다.
구구단 : 1X5=5입니다.
구구단 : 1X6=6입니다.
구구단 : 1X7=7입니다.
구구단 : 1X8=8입니다.
구구단 : 1X9=9입니다.
--------2단---------
구구단 : 2X1=2입니다.
구구단 : 2X2=4입니다.
구구단 : 2X3=6입니다.
구구단 : 2X4=8입니다.
구구단 : 2X5=10입니다.
구구단 : 2X6=12입니다.
구구단 : 2X7=14입니다.
구구단 : 2X8=16입니다.
구구단 : 2X9=18입니다.
```

for 문

- **for 문 : 특정 범위에 대해서 반복하는 문장**

```
for i, j in zip(특정범위1, 특정범위2):
    print(i, j)
```

Tab

for 문

학과 교수님들의 이름 리스트와 전화번호 리스트를 만들어 for문을 활용해 print 하시오

for 문

- for 문 : 딕셔너리의 경우 a={키 : 변수값, 키: 변수값}

```
for i, j in a.items():
    print(i, j)
```
키, 변수값을 동시에 나타냄

```
for i in a.values():
    print(i)
```
키에 대한 변수값을 나타냄

```
for i in a.keys():
    print(i)
```
키값을 나타냄

아래와 같은 별 트리를 만들어 주세요

```
*
**
***
****
*****
******
*******
********
*********
```

for 문

import os ← os란 패키지 사용하기

os.getcwd() ← 현재 작업 위치 검색

os.chdir("원하는 디렉토리 경로") ← 원하는 디렉토리 경로로 작업환경을 이동

dir_list=os.listdir() ← 현재 작업 위치에 있는 파일 리스트 불러오기

for문을 활용해 특정폴더 안에 있는 파일 리스트들을 불러오고 이를 print하시오

01 제어문

for 문

- 리스트에 김철수/20, 이철수/24, 김영희/21, 이영희/24, 김민준/35, 박민준/40, 김지영/34, 박지영/35 를 저장하고

- 하나의 빈 리스트에 아래와 같이 추가하시오
- 문자를 나누는 코드 → 문자.split("문자 안에서 나누고 싶은 기준이 되는 문자")

```
[['김철수', '20'], ['이철수', '24'], ['김영희', '21'], ['이영희', '24'], ['김민준', '35'], ['박민준', '40'], ['김지영', '34'], ['박지영', '35']]
```

- 위와 같이 생성 후 for문을 써 아래와 같이 나타내시오

김철수는 20살이다.
이철수는 24살이다.
김영희는 21살이다.
이영희는 24살이다.
김민준는 35살이다.
박민준는 40살이다.
김지영는 34살이다.
박지영는 35살이다.

If 문

- if 문 : 특정 조건에 대해서 결과를 도출하는 방법

조건에 해당하는 값에 대해 그 조건에 해당하는 특정 명령을 작동해라

```
if 조건:
    특정 명령1
else:
    특정 명령2
```

If 문

- **if 문 : 특정 조건에 대해서 결과를 도출하는 방법**

```
if a>10:
    print(a)
else:
    print("a는 10보다 작다.")
```

If 문

- 다중 if 문 : 특정 조건에 대해서 결과를 도출하는 방법
- 주로 특정 값의 범위를 정하거나 특정 조건을 부여할 때 사용

```
if a>=10:
    print("첫번째")
elif a>=8 and a<10:
    print("두번째")
```

If 문

- 다중 if 문 : 특정 조건에 대해서 결과를 도출하는 방법
- 주로 특정 값의 범위를 정하거나 특정 조건을 부여할 때 사용

```
if a>=10 and b>=10:
    print("첫번째")
elif a>=8 and a<10 and b>=9 and b<10:
    print("두번째")
```

If 문

- 다중 if 문 : 특정 조건에 대해서 결과를 도출하는 방법
- 주로 특정 값의 범위를 정하거나 특정 조건을 부여할 때 사용

```
if a>=10 and b>=10:
    print("첫번째")
elif a>=8 and a<10 and b>=9 and b<10:
    print("두번째")
else:
    print("나머지 전부")
```

If 문

- If문은 문자에 대해서도 특정 조건을 부여할 수 있음 이때 in, not in으로 조건 정의

```
if a in 문자:
    print("포함")
else:
    print("미포함")
```

```
if 문자 in a:
    print("포함")
else:
    print("미포함")
```

If 문

- If문은 문자에 대해서도 특정 조건을 부여할 수 있음 이때 in, not in으로 조건 정의

In [104]:
```
a=input()
if "k" in a:
    print("포함")
else:
    print("미포함")
```

```
korea
포함
```

In [103]:
```
a=input()
if a in "k":
    print("포함")
else:
    print("미포함")
```

```
korea
미포함
```

for 문

for문과 if문을 활용해 특정 폴더에 png, jpg파일만 불러와서 리스트 형태로 저장하는 방법

import os
→ os.chdir("원하는 디렉토리 경로") → 원하는 디렉토리 경로로 작업환경을 이동

os.listdir()
→ 해당하는 디렉토리안의 파일들의 리스트

변수.endswith("끝부분에 포함될 문자")
→ 해당 변수에 끝부분에 포함될 문자 True, False로 분류

While 문

- While : 특정 조건에 대해서 결과를 도출하는 방법
- For문과 while문의 차이 : for 문은 지정된 거 까지 진행, while은 특정 값이 될 때 까지 계속 진행
- 조건이 참(True)인 경우 내부의 수행 부분을 진행하고, 거짓(False)인 경우 while문을 종류함

```
while [조건문]:
    특정 명령
```

Tab

While 문

- While : 특정 조건에 대해서 결과를 도출하는 방법
- For문과 while문의 차이 : for 문은 지정된 거 까지 진행, while은 특정 값이 될 때 까지 계속 진행
- 조건이 참(True)인 경우 내부의 수행 부분을 진행하고, 거짓(False)인 경우 while문을 종류함

```
while True:
    특정 명령
```

Tab

추가적인 조건이 없을 경우 무한 반복

While 문

- While : 특정 조건에 대해서 결과를 도출하는 방법
- For문과 while문의 차이 : for 문은 지정된 거 까지 진행, while은 특정 값이 될 때 까지 계속 진행
- 조건이 참(True)인 경우 내부의 수행 부분을 진행하고, 거짓(False)인 경우 while문을 종료함

```
while i<100:
    i=i+2
    print(i)
```

While 문

- While : 특정 조건에 대해서 결과를 도출하는 방법
- For문과 while문의 차이 : for 문은 지정된 거 까지 진행, while은 특정 값이 될 때 까지 계속 진행
- 조건이 참(True)인 경우 내부의 수행 부분을 진행하고, 거짓(False)인 경우 while문을 종료함

```
while i<100:
    i=i+2
    if i%3==0:
        break
    print(i)
```

```
while i<100:
    i=i+2
    if i%3==0:
        continue
    print(i)
```

- **지금까지 배운것을 활용해 키오스크를 만들어 보자**

```
어서오세요 AI 식당입니다.
햄버거: 15000원
피자: 22000원
음료: 2000원
감자튀김: 5000원
구매하고싶은 품목을 선택해주세요 (or done to exit): 햄버거
------------------------------------------------------------
구매하실 수량을 입력해주세요 : 5
현재 총 구매 금액은 75000원 입니다.
------------------------------------------------------------
구매하고싶은 품목을 선택해주세요 (or done to exit):
------------------------------------------------------------
햄버거: 15000원
피자: 22000원
음료: 2000원
감자튀김: 5000원
구매하고싶은 품목을 선택해주세요 (or done to exit): 음료
------------------------------------------------------------
구매하실 수량을 입력해주세요 : 5
현재 총 구매 금액은 85000원 입니다.
------------------------------------------------------------
구매하고싶은 품목을 선택해주세요 (or done to exit):
------------------------------------------------------------
햄버거: 15000원
피자: 22000원
음료: 2000원
감자튀김: 5000원
구매하고싶은 품목을 선택해주세요 (or done to exit): done
------------------------------------------------------------
귀하의 총 구매금액은 :85000원 입니다.
금액을 투입하세요 : 100000
투입금액은 100000원 이고, 거스름돈은 15000원 입니다.
감사합니다 안녕히 가세요
```

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함

- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
def 함수명(값1, 값2):
    수행문장
    수행문장
```

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함

- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
def 함수명(a, b):
    print(a)
    print(b)
```

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함
- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
def 함수명(a, b):
    c=a+b
    print(c)
```

```
def 함수명(a, b):
    c=a+b
    return c
```

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함
- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
def 함수명(a, b):
    c=a+b
    print(c)
```

```
def 함수명(a, b):
    c=a+b
    return a, b, c
```

return은 여러 개의 값을 반환할 수 있음

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함
- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
def 함수명(a, b):
    c=a+b
    return a, b, c
```

return은 여러 개의 값을 반환할 수 있음

함수안에서 선언된 변수는 단순히 함수 안에서만 작동되고 실제로 저장되지 않음

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함

- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
In [2]:
1 def t(a, b):
2     c=a+b
3     return c
```

```
In [3]:
1 t(1,2)
```

```
Out[3]: 3
```

```
In [4]:
1 a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 a

NameError: name 'a' is not defined
```

```
In [5]:
1 c
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 c

NameError: name 'c' is not defined
```

함수안에서 선언된 변수는 단순히 함수 안에서만 작동되고 실제로 저장되지 않음

return으로 선언되었어도 저장 X

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함

- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
In [6]:
1 c=5
2 def t(a, b):
3     c=a+b
4     return c
```

```
In [7]:
1 t(1,2)
```

```
Out[7]: 3
```

```
In [8]:
1 c
```

```
Out[8]: 5
```

함수 안에서 변수가 선언 되었어도 함수 밖에서 선언된 변수가 실제로 저장됨

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함
- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
In [6]:
1 c=5
2 def t(a, b):
3     c=a+b
4     return c
```

```
In [9]:
1 d=t(1,2)
```

```
In [10]:
1 d
```

```
Out[10]: 3
```

return된 값을 새로운 변수명으로 저장해야 선언이 됨

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함
- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
In [6]:
1 def t(a, b):
2     c=a+b
3     return a,b,c
```

```
In [9]:
1 d=t(1,2)
```

```
In [19]:
1 d
```

```
Out[19]: (1, 2, 3)
```

return된 값이 두개 이상일때 변수 하나로 저장 가능하고, 각각의 값을 각각 다른 변수로 저장가능

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함

- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
In [18]:  1  e, f, g=t(1,2)

In [20]:  1  e

Out[20]: 1

In [21]:  1  f

Out[21]: 2

In [22]:  1  g

Out[22]: 3
```

return된 값이 두개 이상일때 변수 하나로 저장 가능하고, 각각의 값을 각각 다른 변수로 저장가능

- 입력값에 따라 결과가 다를수는 있지만 로직 자체는 같은 경우에 하나의 포장지 안에 넣어서 계속 재사용할 수 있게 만들어 놓은것을 함수라 정의함

- 파이썬에서 함수를 알려주기 위해서는 def 라는 키워드를 사용함

```
In [18]: 1 e, f, g=t(1,2)

In [20]: 1 e
Out[20]: 1

In [21]: 1 f
Out[21]: 2

In [22]: 1 g
Out[22]: 3
```

만약에 return된 값들의 수와 다르게 선언을 할 경우 error

```
In [33]: 1 e, f=t(1,2)
---------------------------------------------------------------------------
ValueError                                Traceback
Cell In[33], line 1
----> 1 e, f=t(1,2)

ValueError: too many values to unpack (expected 2)
```

- **중고폰 가격을 책정해주는 챗봇을 만들어보자**

```
[42]  phone(200000)
```

```
핸드폰에 어떤 문제가 있나요? : 액정파손
핸드폰에 어떤 문제가 있나요? : 카메라 문제
핸드폰에 어떤 문제가 있나요? : 터치불량
핸드폰에 어떤 문제가 있나요? : f
Exiting the program...
130000
```

- *args 는 arguments의 줄임말로 여러 개의 값을 함수로 받고자 할 때 사용함

```
def 함수명(a, b, c):
    print(sum(a,b,c))
def 함수명(*args):
    print(sum(args))
def 함수명(*쓰고 싶은 변수명):
    print(sum(쓰고 싶은 변수명))
```

```
def num(*args):
  print(sum(args))
```

```
num(1,2,3,4,5,6,7,8,9,10)
num(1,2,3,4)
```

```
55
10
```

- ***args 는 arguments의 줄임말로 여러 개의 값을 함수로 받고자 할 때 사용함 또한 추가적인 변수를 입력하고 싶을때 해당 변수의 이름을 지정해 출력**

```
def 함수명(*args, 추가적인 변수):
    print(sum(args))
    print(추가적인 변수)
```

```
def num(*args, name):
  print(sum(args), name)
```

```
num(1,2,3,4,5,6,7,8,9,10, "김창균")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-43-499528319ae1> in <module>
----> 1 num(1,2,3,4,5,6,7,8,9,10, "김창균")

TypeError: num() missing 1 required keyword-only argument: 'name'
```

SEARCH STACK OVERFLOW

```
num(1,2,3,4,5,6,7,8,9,10, name="김창균")
```

```
55 김창균
```

- 중고폰 가격을 책정해주는 챗봇을 *args를 이용해 만들어보자

```
phone("액정파손","카메라 문제", price=500000)
```

```
440000
```