학습 내용 및 목표

- if문에 대해 대해 알아보고 실습해 본다.
- for문에 대해 대해 알아보고 실습해 본다.
- 파이썬의 자료형에 대해 알아보고 실습해 본다.

1-1 if

- 기본 구조1

```
if (조건식):
  실행문1
```

- 기본 구조2

```
if (조건식):
  실행문1
else:
  실행문2
```

- 기본 구조3

```
if (조건식) & (조건식):
  실행문1
  실행문2
else:
  실행문3
  실행문4
```

- 기본 구조4

```
if (조건식) & (조건식):
  실행문1
  실행문2
elif (조건식):
  실행문3
elif (조건식):
  실행문4
else:
  실행문3
  실행문4
```

In [3]:

```
people = 10

# 인덴트가 다를 경우, 에러가 발생.
if people >= 10:
  print("사람이 10명이상입니다.") # indent라고 한다. 동일하게 해 준다.
   print("사람이 10명이상입니다.") # 이 줄은 인덴트(들여쓰기) 공간이 다르다. 에러 발생.
  print("사람이 10명이상입니다.")
```

```
  File "<ipython-input-3-a5b51e4e91fc>", line 5
    print("사람이 10명이상입니다.") # 이 줄은 인덴트(들여쓰기) 공간이 다르다. 에러
 발생.
    ^
IndentationError: unexpected indent
```

In [4]:

```
people = 10

# 인덴트가 다를 경우, 에러가 발생.
if people >= 10:
  print("사람이 10명이상입니다.") # indent라고 한다. 동일하게 해 준다.
  print("사람이 10명이상입니다.") # 이 줄은 인덴트(들여쓰기) 공간이 다르다. 에러 발생.
  print("사람이 10명이상입니다.")
```

```
사람이 10명이상입니다.
사람이 10명이상입니다.
사람이 10명이상입니다.
```

- (생각해 보기) 사람의 수를 입력받아,
  - 10명 이상이면, 10명 이상입니다. 출력
  - 10명 미만이면, 10명 미만입니다. 출력

In [5]:

```
num = int( input("사람의 수는 :") )
if num >= 10:
 print("사람이 10명이상입니다.") # indent라고 한다. 동일하게 해 준다.
 print("사람이 10명이상입니다.")
else:
 print("사람이 10명 미만입니다")
```

```
사람의 수는 :9
사람이 10명 미만입니다
```

- 영화관 입장 판정 프로그램 작성
  - 입구의 안내원 또는 티켓 판매 기계가 물어봅니다.
  - '나이가 어떻게 되나요?' 질문
    - 15세 이하는 영화관 입장 불가
    - 15세 이상은 다음 단계 진행

In [6]:

```
age = int(input("나이가 어떻게 되나요?"))

if (age>=15):
 print("입장 가능합니다. 어떤 영화 티켓을 구매하시겠어요?")
else:
 print("입장이 불가합니다. 15세이상만 입장이 가능합니다.")
```

```
나이가 어떻게 되나요?17
입장 가능합니다. 어떤 영화 티켓을 구매하시겠어요?
```

- 학점 판정 프로그램 만들어보기
  - 4.5 이면 A+
  - 4.0 이상이면 A
  - 3.5 이상이면 B+
  - 3.0 이상이면 B
  - 2.5 이상이면 C+
  - 나머지 F

In [9]:

```
score = float( input("학점입력해 주세요 : ") )

if score==4.5:
 grade = 'A+'
elif score>=4.0:
 grade = 'A'
elif score>=3.5:
 grade = 'B+'
else:
 grade = 'F'

print("당신의 학점은 ", grade , "입니다.")
```

```
학점입력해 주세요 : 4.
당신의 학점은  A 입니다.
```

- 실습 - 아래와 같이 판정하는 프로그램을 작성해 보자.
  - 점수가 90점 이상이면 A
  - 점수가 80점 이상이면 B
  - 점수가 70점 이상이면 C
  - 나머지 점수는 F

1-2 for

- 기본 구조1

```
for 변수 in range(시작값, 끝값, 증감값):
  실행문1
```

- 1-10까치 출력하는 프로그램 만들기

In [10]:

```
for num in range(1,11,1):
  print(num)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [11]:

```
for num in range(1,11):  # 마지막 증감값 1은 기본값으로 생략되면 1씩 증가
  print(num)
```

```
1
2
3
4
5
6
7
8
9
10
```

- 기본 구조2
  - 변수의 값은 for문 루프안에서 사용이 가능.
  - range 부분에 자료형(리스트, 딕셔너리)이 들어가는 것이 가능.

```
for 변수 in ['하나', '둘', '셋', '넷', '다섯']:
  실행문1
  실행문2
```

- 리스트 자료형

In [13]:

```
al = ['하나', '둘', '셋', '넷', '다섯']
```

In [14]:

```
al = ['하나', '둘', '셋', '넷', '다섯']

for one in al:
  print(one)
```

```
하나
둘
셋
넷
다섯
```

In [16]:

```
num = {'one':'하나', 'two':'둘', 'three':'셋', 'four':'넷', 'five':'다섯'}

for one in num:
  print(one)
```

```
one
two
three
four
five
```

In [18]:

```
num = {'one':'하나', 'two':'둘', 'three':'셋', 'four':'넷', 'five':'다섯'}

for one in num:
  print(one, num[one])
```

```
one 하나
two 둘
three 셋
four 넷
five 다섯
```

1-3 자료형 - 문자열

- 문자열

In [19]:

```
string = "Hello World!"
string[0] # 첫번째 선택 (파이썬은 숫자 0부터 시작)
```

Out[19]:

```
'H'
```

- slicing - 일부 선택

In [21]:

```
# Hello 선택
# : 을 기준으로 앞이 시작값, 뒤에값이 끝나는 값.
string[0:5] # 첫번째(0)부터 다섯번째(4)까지 선택 (파이썬은 숫자 0부터 시작)
```

Out[21]:

```
'Hello'
```

In [23]:

```
# Hello 선택
string[ :5] # 처음부터 다섯번째(4)까지 선택 (파이썬은 숫자 0부터 시작)
```

Out[23]:

```
'Hello'
```

In [26]:

```
# 전체 선택
string[ : ] # 전체를 선택
```

Out[26]:

```
'Hello World!'
```

In [27]:

```
string[2:5] # 세번째(2)부터 다섯번째(4)까지 선택 (파이썬은 숫자 0부터 시작)
```

Out[27]:

```
'llo'
```

In [28]:

```
## 인덱싱(Indexing) : 값을 하나 하나 선택
## 슬라이싱(slicing) : 값을 조각 단위로 선택
```

- 실습해 보기
  - 나의 이름을 입력받아(input), 앞의 3자리를 출력해 보자.

1-4 자료형 - 리스트

In [38]:

```
a = [1,2,3,4,5]
print(a)
```

```
[1, 2, 3, 4, 5]
```

In [39]:

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

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

In [40]:

```
print( a[1] )  # 0부터 시작하여 1은 두번째 요소를 가르킴
```

```
2
```

In [41]:

```
print( a[2:5] )
```

```
['a', 3, 'b']
```

In [42]:

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

```
b
```

In [43]:

```
# 뒤에서부터 3개 가져오기
print( a[-3] )
```

```
a
```

In [44]:

```
a = [ [1,2],
      [2,3],
      [4,5]
 ]
print(a[1]) # 2번째 값([2,3])
print(a[1][1]) # 2번째 값중의 2번째 값을 가져오기
```

```
[2, 3]
3
```

생각해보기 & 실습해 보기

- 2차 리스트를 만들고, 각각의 원소는 3개씩으로 하기.
  - 2번째 행의 3번째 값을 출력해 주세요.

리스트의 연산

In [48]:

```
list1 = [1,2,3]
print( list1 )
```

```
[1, 2, 3]
```

- 하나의 값을 추가

In [49]:

```
list1.append(4)
print( list1)
```

```
[1, 2, 3, 4]
```

In [50]:

```
[1,1] + [2,2]
```

Out[50]:

```
[1, 1, 2, 2]
```

In [54]:

```
list2 = [1,2,3,2]
list2.remove(2)  # 요소중에 2의 값을 하나 삭제(앞에서부터 봤을때)
list2
```

Out[54]:

```
[1, 3, 2]
```

In [55]:

```
## 두개의 값을 한꺼번에 추가하기
list3 = [1,2,3]
list3.append([3,4])
print(list3)
```

```
[1, 2, 3, [3, 4]]
```

In [56]:

```
## 두개의 값을 한꺼번에 추가하기
list4 = [1,2,3]
list4.extend([3,4])
print(list4)
```

```
[1, 2, 3, 3, 4]
```

1-5 리스트와 for문

In [58]:

```
list1 = [1,2,3,4,5,11,22,33,44,55]
for i in list1:
  print(i, end=" ")  # 출력 후, 한칸을 띄우고 다음번 진행
```

```
1 2 3 4 5 11 22 33 44 55
```

In [59]:

```
# range(시작값,끝값,증가값)
for i in range(0,10,1):
  print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

In [60]:

```
for i in range(10):
  print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

In [61]:

```
for i in range(5, 0,-1):
  print(i, end=" ")
```

```
5 4 3 2 1
```

In [63]:

```
season = ['봄','여름','가을', '겨울']
print("season의 요소의 개수 : ", len(season))
for one in season:
  print(one)
```

```
season의 요소의 개수 :  4
봄
여름
가을
겨울
```

1-6 딕셔너리(dict)

- [] : 리스트 => 값의 변경이 가능하다. 수정이 가능하다. 추가가능
- () : 튜플 => 값의 변경이 불가. 속도가 좀 더 빠르다.
- {} : 딕셔너리 => 한쌍의 데이터가 이루어져 있다. 키:값

- 딕셔너리는 {}로 둘러싸이며, 키:값이 한쌍을 이루고, ','를 기준으로 구분된다.

In [64]:

```
dictdat1 = { 'key1':"value1", 'key2':"value2", 'key3':'value3'}
dictdat2 = { 11:"value1", 22:"value2", 33:'value3'}
dictdat3 = { 'key1':"value1", 'key2':(1,2,3,4,5), 'key3':[1,2,3,4,5] }
```

In [65]:

```
# dictdat3['키값']
dictdat1['key3']
```

Out[65]:

```
'value3'
```

In [66]:

```
# dictdat3['키값']
dictdat3['key3']
```

Out[66]:

```
[1, 2, 3, 4, 5]
```

1-7 튜플(tuple)

- 리스트와 달리 값이 변경과 수정이 어렵다.
- 장점은 필요한 기능이 줄기때문에 리스트에 비해 상대적으로 빠르다.

In [67]:

```
tuple1 = (1,2,3)
tuple1
```

Out[67]:

```
(1, 2, 3)
```

In [68]:

```
tuple1[1]
```

Out[68]:

```
2
```

In [70]:

```
tuple1[1] = 20  # 에러 발생
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-70-4eb5dd8d8af4> in <module>()
----> 1 tuple1[1] = 20  # 에러 발생

TypeError: 'tuple' object does not support item assignment
```

In [71]:

```
tuple1 = (1,2,3)
list1 = [1,2,3]
```

리스트와 튜플의 사용 객체 확인하기

In [73]:

```
print(tuple1.__sizeof__())
dir(tuple1)
```

48

Out[73]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

In [75]:

```
print(list1.__sizeof__())
dir(list1)
```

64

Out[75]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

1-8 다음시간 내용 살펴보기 - 함수

In [81]:

```
def two_num_plus(a,b):
  print("두 값의 합은 :", a+b)
```

In [82]:

```
two_num_plus(3,5)
```

```
두 값의 합은 : 8
```

생각해보기 & 실습1

- 복권 당첨 프로그램을 만들어보기
- [1,2,3,4,5,6,7,8,9,10]
- 당첨번호는 1등, 2등, 3등은 임의로 사용자가 선정
- input()을 이용하여 사용자가 1-10 중의 하나를 선택
- 선택된 번호로 1등, 2등, 3등, 그리고 꽝을 출력해보자.

생각해보기 & 실습2

- 말하는 대화 로봇을 만들기
- '안녕' 을 입력하면 말하는 로봇은 '반가워 나는 toto야'으로 대답
- 몇가지 대화를 만들고, 이에 맞는 단어가 나오면 답변해 보자.

In [77]:

```
robot = {'안녕':'반가워 난 에프라고해.', '날씨는 좋구나':'그래 날씨도 좋고, 너도 만나서 기쁘다 좋은

print(robot.keys())
```

```
dict_keys(['안녕', '날씨는 좋구나'])
```

In [80]:

```
for i in range(10):
  word = input("대화를 입력해 주세요?(종료:q) ")
  if word in robot.keys():
    print(robot[word])
  else:
    print("무슨이야기인지 아직 모르겠어. 미안")
```

```
대화를 입력해 주세요?(종료:q) q
```

- 실습 3
  - 위의 실습2에 종료기능을 추가해 보자. q를 입력하면 더 이상 묻지 않고, 끝내기