```
#### 파이썬 이란
- 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어
- 컴퓨터 프로그래밍 교육을 위해 많이 사용하지만, 기업의 실무를 위해서도 많이 사용하는 언어. 구글에서 만든 소프트웨어의 50%이상이 파이썬으로 작성

#### 파이썬의 특징
- 파이썬은 인간다운 언어이다 (=고수준 언어)
- 파이썬은 문법이 쉬워 빠르게 배울 수 있다
- 파이썬은 무료이지만 강력하다
- 파이썬은 간결하다
- 파이썬은 프로그래밍을 즐기게 해준다
- 파이썬은 개발 속도가 빠르다

#### 파이썬으로 할 수 있는 일
- 웹 개발: Django, Flask 등의 프레임워크를 사용하여 웹 애플리케이션을 개발할 수 있습니다.
- 데이터 분석: Pandas, NumPy(수치), SciPy(통계, 수리)와 같은 라이브러리를 사용하여 데이터를 분석하고 처리할 수 있습니다.
- 머신 러닝과 인공지능: TensorFlow, PyTorch, Scikit-learn 등의 라이브러리를 활용하여 머신 러닝 모델을 구축하고 훈련시킬 수 있습니다.
- 자동화: 파이썬 스크립트를 작성하여 일상적인 작업을 자동화하고, 시스템 관리 작업을 수행할 수 있습니다.
- 게임 개발: Pygame과 같은 라이브러리를 사용하여 간단한 게임을 개발할 수 있습니다.
- 모바일 애플리케이션 개발: Kivy 또는 BeeWare와 같은 라이브러리를 사용하여 모바일 애플리케이션을 개발할 수 있습니다.
- 데스크탑 애플리케이션 개발: PyQt, Tkinter 등의 라이브러리를 활용하여 데스크탑 애플리케이션을 개발할 수 있습니다.
- 시스템 스크립팅과 네트워킹: 시스템 유틸리티를 개발하거나 네트워크 프로토콜을 구현할 수 있습니다.
- 임베디드 시스템과 하드웨어 제어: 라즈베리 파이와 같은 임베디드 시스템을 제어하고 하드웨어를 프로그래밍할 수 있습니다.
- 사이언티픽 컴퓨팅: 과학적 연산과 시뮬레이션을 위해 파이썬을 활용할 수 있습니다.
- 교육: 파이썬은 초보자에게 프로그래밍을 가르치는 데 이상적인 언어로 평가받고 있습니다.
- 파이썬의 다양한 라이브러리와 프레임워크 덕분에, 이러한 분야에서의 작업이 더욱 쉽고 효율적으로 수행될 수 있습니다.
```

```
  File "<ipython-input-7-0b6cd532b9c2>", line 2
    - 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 인터프리터 언어
      ^
SyntaxError: invalid decimal literal
```

Next steps: Fix error

```
# ! 기호는 Colab셀에서 Unix/Linux 셸 명령어를 실행
!python--version
```

```
/bin/bash: line 1: python--version: command not found
```

```
# % 기호는 IPython 환경(즉, colab이 포함된 Jupyter 환경)에서 제공하는 매직 명령어를 사용
# 라인 매직은 단일 라인에 대해 실행되며 % 하나를 사용하고 셸 매직은 셸 전체에 적용되며 %%를 사용
# 현재 작업폴더 확인
%pwd
```

```
'/content'
```

```
%%time
#간단한 for 루프를 사용한 계산
sum = 0
for i in range(100000):
  sum += i
print(sum)
```

```
4999950000
CPU times: user 19.7 ms, sys: 825 µs, total: 20.5 ms
Wall time: 20.5 ms
```

```
#### 용어
- 식별자 : 프로그램밍 언어에서 이름을 붙일 때 사용하는 단어. 주로 변수 또는 함수 이름 등으로 사용
- 주석 : 프로그램을 설명하기 위해 사용. # 기호로 주석 처리
- 연산자 : 스스로 값이 되는 것이 아니고 값과 값 사이에 무언가 기능을 적용할 때 사용
- 자료 : 리터럴이라고 하는데 숫자이든 문자이든 어떠한 값 자체를 의미. 1, 10, "Hello"
- 키워드 : 파이썬이 만들어질 때 이미 사용하겠다고 예약해 놓는 것. False, None, True, ...
- 프로그램밍 언어에서 사용자가 이름을 정할 때 키워드는 사용할 수 없음(식별자를 키워드로 사용할 수 없음.)
#### 식별자
count, user_name, _is_valid, calculate_area, Car, model, year, math 및 m 모두 유효한 식별자. 각각의 식별자는 특정한 데이터 또는 기능에 이름을 부여하C
```

```
  File "<ipython-input-11-4070495c74c8>", line 2
    - 식별자 : 프로그램밍 언어에서 이름을 붙일 때 사용하는 단어. 주로 변수 또는 함수 이름 등
으로 사용
      ^
SyntaxError: illegal target for annotation
```

Next steps: Fix error

```
# 변수 식별자
count = 10
user_name = "Alice"
_is_vaild + True

# 함수 식별자
def calculate_area(radius):
  return 3.14159 * radius * radius

# 클래스 식별자
class Car:
  def__init__(self, model, year):
    self.model = model
    self.year = year

# 모듈 식별자
import math as m



## 식별자 기본규칙
* 키워드를 사용하면 안된다.
* 특수문자는 언더 바(_)만 사용
* 숫자로 시작하면 안된다.
* 공백을 포함할 수 없다.


import keyword
print(keyword.kwlist)
len(keyword.kwlist)


alpha
break #키워드
alpha10
_alpha
273alpha #숫자
AlpHa
ALPHA
has space #공백


Q. 주어진 문자열 리스트에서 유효한 Python 변수 이름만을 추출하여 반환하는 함수를 작성하세요.


identifiers = ["var1", "2things", "variable_name", "time!"]


# isidentifier()는 파이썬의 문자열 메서드로, 주어진 문자열이 유효한 식별자인지를 확인
# isidentifier()는 예약어인지는 따로 체크하지 않음
import keyword
def valid_identifiers(identifiers):
  valid = []
  for identifier in identifiers:
    if identifier.isidentifier() and not keyword.iskeyword(identifier):
      valid.append(identifier)
    return valid


# 예제 실행
identifiers = ["var1", "2things", "variable_name", "time!", "True"]
print(valid_identifiers(identifiers))


파이썬은 snake_case와 CamelCase를 모두 사용
- itemlist : item_list itemList
- loginstatus : login_status loginStatus
- 캐멀 케이스(대문자로 시작) 클래스
- 스네이크 케이스(소문자로 시작) 뒤에 ()가 있다 - 함수
- 스네이크 케이스(소문자로 시작) 뒤에 ()가 없다 - 변수


# 연산자
a = 5
b = 3
c = b % a  #몫
d = b // a #나머지
e = b / a
print(c)
print(d)
print(e)
```

```
a = 3
b = 5

if a > b:
  print(a)
```

자료형

```
#### 자료형
- 자료형 또는 데이터 타입이란 숫자, 문자 등과 같이 여러 종류의 데이터를 구분하기 위한 분류
- 파이썬의 자료형은 크게 숫자(numbers), 시퀀스(sequence), 매핑(mapping) 등으로 나눌 수 있다.
- 파이썬의 기본 자료형
  - 수치형
    - 정수형 : int는 정수(integer)를 나타낸다. 양의 정수와 음의 정수, 숫자 0
    - 실수형 : float는 원래 부동소수점수(floating-point number)를 가리키는데, 지금은 단순히 소수점 이하를 표현할 수 있는 수이다.
    - 복소수형 : 복소수를 complex로 나타내고 제곱하면 -1이 되는 수 i를 '허수(imaginary number)'라고 하는데 허수 i를 j로 표현
  - 시퀀스 : 문자열(str), 리스트(list), 튜플(tuple), 사용자 정의 클래스가 시퀀스에 속한다. for 문에서 사용할 수 있는 것들이 바로 시퀀스
    - 문자열 : 문자를 한 줄로 표현하며 문자열 인덱스를 이용해 문자열의 일부를 복사
    - 리스트 : 대괄호([ ])로 감싸 주고 각 요솟값은 쉼표(,)로 구분
    - 튜플 : 튜플은 ( )으로 둘러싸고 각 요솟값은 쉼표(,)로 구분
  - 매핑
    - 사전 : 딕셔너리(dict)는 키(key)와 값(value)의 짝으로 이뤄지는데 이런 것을 매핑
  - 집합 : 집합을 표현하는 세트(set)
  - 블린 : 참, 거짓을 표현


#정수형, 실수형
i1 = 3
f1 = 3.5
print(i1)
print(f1)


#정수(int)
print(int(True))
print(int(False))
print(int('100'))
print(int(3.14))


#실수(float)
print(float(True))
print(float(False))
print(float('100'))
print(float(3.14))


# 사칙연산 : +, *, /, //, %, **
a = 10
b = 2.3
print(a+b)
print(a*b)
print(a/b)
print(a//b)
print(a%b)
print(a**b)


# 문자열
string = '문자열'
a = "100"


print(string)
a


a = 100
b = 50
print(a+b)
a = '100'
b = '50'
print(a+b)


a = int('100')
b = int('50')
print(a+b)


# 사용자 입력
input('인사말을 입력하세요')
```

```
input('안녕하세요')


a = input()
b = input()
print(a+b)


#input() 함수는 입력받은 데이터를 string으로 처리한다.
a = int(input())
b = int(input())
print(a+b)


# 실수 + 실수 -> 실수
# Q. 3과 3.3을 입력하고 숫자 연산을 수행하여 6.3을 출력하세요.
a = 3
b = 3.3
print(a+b)


# 정수 + 실수 -> 실수
# Q. 3과 3.3을 입력하고 숫자 연산을 수행하여 6.3을 출력하세요.
num1 = int(input("정수를 입력하세요 : "))
num2 = float(input("소수점 이하를 포함한 실수를 입력하세요 : "))
print(num1 + num2)


#Q.a = 54와 b=52.273일때 a+b는 5252.273으로 출력되도록 a와 b를 문자열 자료형으로 변환하세요.
a = 52
b =52.273
a = str(a)
b = str(b)
print(a+b)


# 문자열 : "", '', """""", ''''''
# p1 = 'python's value is great'
p1 = "python's value is great"
print(p1)


# 백슬래시 (₩)는 문자열 안에서 작은따옴표 (')를 문자열의 끝을 의미하지 않는 일반 문자로 포함시키기 위해 사용
# 이 기술을 이스케이프라고 함.
p1 = 'python₩'s value is great'
print(p1)


# ₩n 행 바꿈
# y1 = "Once you study data analysis You need Python"
y1 = "Once you study data analysis₩nYou need Python"
print(y1)
y2 = "Once you study data analysis₩tYou need Python"
print(y2)


y2 ='''
Once you study data analysis
You need Python
'''
print(y2)
y3 ="""
Once you study data analysis
You need Python
"""
print(y3)


a1 = 'Python'
a2 =  ' is easy to learn'
print(a1+a2)
print("="*25)


a1 = '현지니는'
a2 = '지금 커피가 먹고 싶다 '
print(a2+a1)


# Q. 아래와 같이 출력할 수 있는 string을 만드세요
string = '''
"What a wonderful world!"
he said loudly
'''
print(string)
```

```
# Q. 다른 타입의 숫자 2개를 입력받아 큰 수를 출력하세요.
num1 = int(input("정수를 입력하세요"))
num2 = int(input("정수를 입력하세요"))
print(num1) if num1>num2 else print(num2)
```

```
num1 = int(input("첫번째 숫자"))
num2 = int(input("두번째 숫자"))
if num1>num2:
  print(num1)
elif num1<num2:
  print(num2)
else:
  print("큰 수가 없습니다.")
```

```
첫번째 숫자3
두번째 숫자3
큰 수가 없습니다.
```

```
# 자료형 - List
# 리스트는 []로 표시하며 []안의 요소를 콤마로 구분하여 순서있게 나열
list1 = [1, 2, 3, 4, 5]
list2 = ['a', 'b', 'c']
list3 = [1, 'a', 'abc', [1, 2, 3, 4, 5], ['a', 'b', 'c']]
print(list1)
print(list2)
print(list3)
```

```
[1, 2, 3, 4, 5]
['a', 'b', 'c']
[1, 'a', 'abc', [1, 2, 3, 4, 5], ['a', 'b', 'c']]
```

```
# 인덱싱
list1 = [1, 2, 3, 4, 5]
list1[2]
list1[0:2]
list1[:4]
```

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

```
list3 = [1, 'a', 'abc', [1, 2, 3, 4, 5], ['a', 'b', 'c']]
print(list3[3][2])
print(list3[4][0])
```

```
3
a
```

```
# 리스트 수정, 삭제
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
b = ['a', 'b', 'c']
a[0] = 1
print(a)
b[1] = 'a'
print(b)
del a[0]
print(a)
del b[1]
print(b)
```

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

```
# 끝에서 부터 인덱싱 : -1
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
b = ['a', 'b', 'c']
print(a[-2])
print(a[-4:-1])
print(a[-4:])
```

```
8
[6, 7, 8]
[6, 7, 8, 9]
```

```
# 리스트 확장
h = [1, 2, 3]
h.extend([4,5]) # 반복 가능한 객체의 모든 요소를 리스트에 추가
print(h)
h.append(6) # 단일 객체를 추가
print(h)
h.insert(0,5)
print(h)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6]
[5, 1, 2, 3, 4, 5, 6]
```

```
Q. list1에서 다음을 수행하세요
- 33을 출력
- 82를 리스트에 추가
- 87의 인덱스 구하기 - 도전
- 인덱스 3에서 10까지의 값을 출력하고 list2에 저장한 후 내림차순 정렬하기 - 도전
- 39를 11로 변경
- [69,45,58] 출력
- 짝수 인덱스의 값으로 구성된 리스트 출력하기
- 인덱스가 가장 큰수를 삭제하기
- 인덱스 3, 5인 값으로 4칙 연산하기

list1 = [58, 45, 69, 19,  4, 87, 29, 13, 39, 15, 54, [8, 33, 11], 27, 49, 63, 98, 22, 82]
```

```
  File "<ipython-input-43-0026c5ab9eb5>", line 2
    - 33을 출력
      ^
SyntaxError: invalid decimal literal
```

Next steps: Fix error

```
list1 = [58, 45, 69, 19,  4, 87, 29, 13, 39, 15, 54, [8, 33, 11], 27, 49, 63, 98, 22, 82]

print(list1[11][1])
list1.append(82)
print(list1)
print(list1.index(87))
list2 = list1[3:11]
print(list2)
list2.sort(reverse=True)
print("list2.sort(reverse=True)")
print(list2)
list1[8] = 11
print(list1)
list3 = list1[:-16]
list3.sort(reverse=True)
print("list3.sort(reverse=True)")
print(list3)
list1[0::2]
del list1[-1]
print(list1)
a = list1[3]
b = list1[5]
print(a+b)
print(a*b)
print(a/b)
print(a-b)
```

```
33
[58, 45, 69, 19, 4, 87, 29, 13, 39, 15, 54, [8, 33, 11], 27, 49, 63, 98, 22, 82, 82]
5
[19, 4, 87, 29, 13, 39, 15, 54]
list2.sort(reverse=True)
[87, 54, 39, 29, 19, 15, 13, 4]
[58, 45, 69, 19, 4, 87, 29, 13, 11, 15, 54, [8, 33, 11], 27, 49, 63, 98, 22, 82, 82]
list3.sort(reverse=True)
[69, 58, 45]
[58, 45, 69, 19, 4, 87, 29, 13, 11, 15, 54, [8, 33, 11], 27, 49, 63, 98, 22, 82]
106
1653
0.21839080459770116
-68
```

```
print(list1)
print(list1[7::-1])
print(list1[7::2])
print(list1[7::-2])
```

```
[58, 45, 69, 19, 4, 87, 29, 13, 11, 15, 54, [8, 33, 11], 27, 49, 63, 98, 22, 82]
[13, 29, 87, 4, 19, 69, 45, 58]
[13, 15, [8, 33, 11], 49, 98, 82]
[13, 87, 19, 45]
```

1. 메서드 vs 함수:
- sort(): 리스트 객체의 내장 메서드입니다. 즉, 리스트에서만 사용할 수 있습니다.
- sorted(): 내장 함수로, 어떤 반복 가능한(iterable) 객체에도 사용될 수 있습니다. 예를 들면 리스트, 튜플, 딕셔너리, 문자열 등에 사용할 수 있습니다.

2. 반환 값:
- sort(): 리스트를 원 위치에서(in-place) 정렬하고 None을 반환합니다. 따라서 원래의 리스트 자체가 변경됩니다.
- sorted(): 정렬된 새로운 리스트를 반환합니다. 원래의 객체는 변경되지 않습니다.

3. 유용성:
- sort(): 리스트에서만 작동하기 때문에 리스트만 정렬할 수 있습니다.
- sorted(): 다양한 객체를 정렬할 수 있으며 결과는 항상 리스트로 반환됩니다.

```
# sort()
my_list = [3, 1, 2]
my_list.sort()
print(my_list)

# sorted()
my_tuple = (3, 1, 2)
new_list = sorted(my_tuple)
print(my_tuple)
print(new_list)
```

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

```
list = [5, 6, 7, 8, 9]
new_list = list.sort()
print(list)
print(new_list)
```

```
[5, 6, 7, 8, 9]
None
```

```
list = [5, 6, 9, 7, 8]
new_list = sorted(list)
print(list)
print(new_list)
```

```
[5, 6, 9, 7, 8]
[5, 6, 7, 8, 9]
```

```
list1 = [58, 45, 69, 19,  4, 87, 29, 13, 39, 15, 54, [8, 33, 11], 27, 49, 63, 98, 22, 82]
list2 = list1[3:11]
print(list2)
l_sort = sorted(list2,reverse=True)
l_sort
```

```
[19, 4, 87, 29, 13, 39, 15, 54]
[87, 54, 39, 29, 19, 15, 13, 4]
```

```
list3 = list1[3:11]
list3.sort(reverse=True)
print(list3)
```

```
[87, 54, 29, 19, 15, 13, 11, 4]
```

Task1_0425. 주어진 숫자 리스트에서 최소값과 최대값을 찾아 출력하세요.

```
numbers = [58, 45, 69, 19, 4, 87, 29, 13, 39, 15]
```

```
numbers = [58, 45, 69, 19, 4, 87, 29, 13, 39, 15]
l_sort = sorted(numbers,reverse=True)
l_sort
print(l_sort[0])
print(l_sort[-1])
```

```
87
4
```

Task2_0425. 주어진 숫자 리스트의 모든 요소의 합계와 평균을 계산하고 출력하세요

```
numbers = [58, 45, 69, 19, 4, 87, 29, 13, 39, 15]
```

```
numbers = [58, 45, 69, 19, 4, 87, 29, 13, 39, 15]
result = sum(numbers)
print(result)

result1 = len(numbers)
print(result1)
print(result/result1)
```

```
378
10
37.8
```

Task3_0425. 주어진 리스트에서 특정 요소가 등장하는 모든 인덱스를 리스트로 만들어 출력하세요.

```
items = ['apple', 'banana', 'cherry', 'apple', 'cherry', 'apple']
target = 'apple'
```

```
items = ['apple', 'banana', 'cherry', 'apple', 'cherry', 'apple']
target = 'apple'
indices = [index for index, item in enumerate(items) if item == target]
print(indices)
```

```
[0, 3, 5]
```

Task4_0425. 주어진 리스트에서 연속해서 반복되는 요소만 제거하고, 결과 리스트를 반환하세요. 단, 처음 등장하는 요소는 유지해야 합니다.

예를 들어, ['a', 'a', 'b', 'c', 'c', 'c', 'd', 'e', 'e']가 입력되면, ['a', 'b', 'c', 'd', 'e']를 출력해야 합니다.

코딩을 시작하거나 AI로 코드를 생성하세요.

Task5_0425 주어진 정수 리스트와 회전 횟수 k에 대해 리스트를 오른쪽으로 k만큼 회전(이동)시킨 결과를 반환하세요. k가 리스트의 길이보다 클 수 있으며, 이

예를 들어, [1, 2, 3, 4, 5]와 k=2가 주어지면, 결과는 [4, 5, 1, 2, 3]이 되어야 합니다.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.