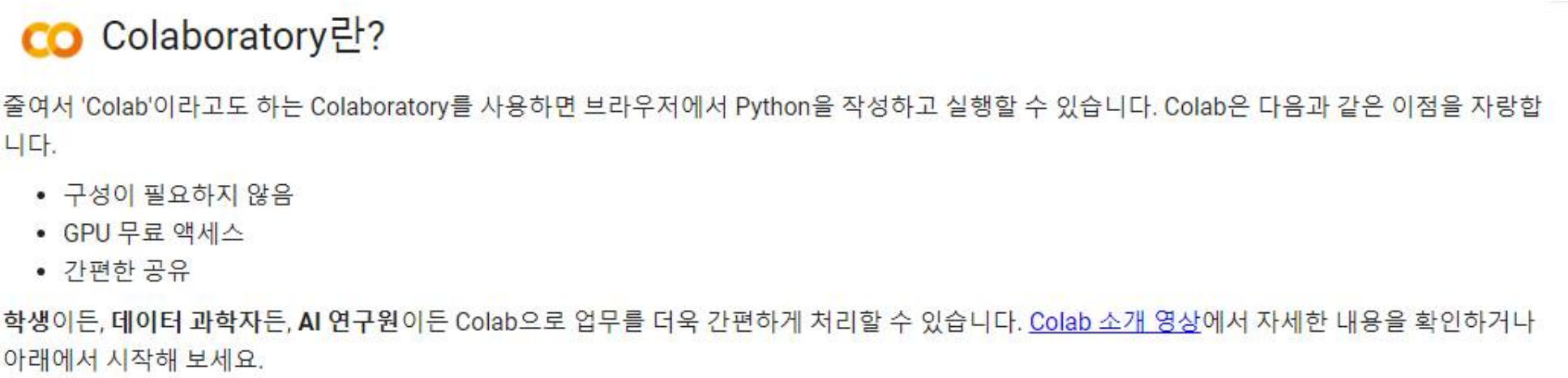
Python 기초 프로그래밍 I

자료형, 연산자 등

google Colaboratory (Colab)

- https://colab.research.google.com/notebooks/intro.ipynb
- 구글 계정 생성 및 로그인 필요
- 기초 문법 실습 시 사용 (함수 작성 전 사용 가능)
  - 인터프리터 실습



Colaboratory란?

줄여서 'Colab'이라고도 하는 Colaboratory를 사용하면 브라우저에서 Python을 작성하고 실행할 수 있습니다. Colab은 다음과 같은 이점을 자랑합니다.

- 구성이 필요하지 않음
- GPU 무료 액세스
- 간편한 공유

학생이든, **데이터 과학자**든, **AI 연구원**이든 Colab으로 업무를 더욱 간편하게 처리할 수 있습니다. Colab 소개 영상에서 자세한 내용을 확인하거나 아래에서 시작해 보세요.

파이썬 프로그래밍의 기초, 자료형

- 숫자형(Number)
  - 숫자 형태로 이루어진 자료형

항목	파이썬 사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8진수	0o34, 0o25
16진수	0x2A, 0xFF

파이썬 프로그래밍의 기초, 자료형

- 숫자형(Number)
  - 숫자 형태로 이루어진 자료형

▪ 숫자형 사용법

▪ 정수형(Integer)

- 정수를 뜻하는 자료형

```
>>> a = 123
>>> a = -178
>>> a = 0
```

▪ 실수형(Floating-point)

- 소수점이 포함된 숫자

```
>>> a = 1.2
>>> a = -3.45
```

```
>>> a = 4.24E10
>>> a = 4.24e-10
```

※ 컴퓨터식 지수 표현 방식

▪ 8진수(Octal)

- 숫자 0 + 알파벳 소문자 o 또는 대문자 O

```
>>> a = 0o177
```

▪ 16진수(Hexadecimal)

- 숫자 0 + 알파벳 소문자 x

```
>>> a = 0x8ff
>>> b = 0xABC
```

Practice #1 (숫자형)

pycharm

```
a = 123
b = -123
print(a, b)

fa = 12.345
fb = -3.14
fexpA = 4.24E3
fexpB = 4.24E-3

print(f'float number example: {fa}, {fb}, {fexpA}, {fexpB}')

#8진수 practice
octA = 0o1777
print(octA)

#16진수 (HexaDecimal)
hexA = 0x1FF24
print(f'{hexA:d}, {hexA:0x}, {hexA:f}, {hexA:.2f}')
```

colab

자료형-숫자형 연습

```
#정수형
a=123
a
```

```
123
```

```
[4] #실수형
b=12.34
floatTmp=3.14E7
print(floatTmp)
print(f'{floatTmp+b}')
b
```

```
31400000.0
31400012.34
12.34
```

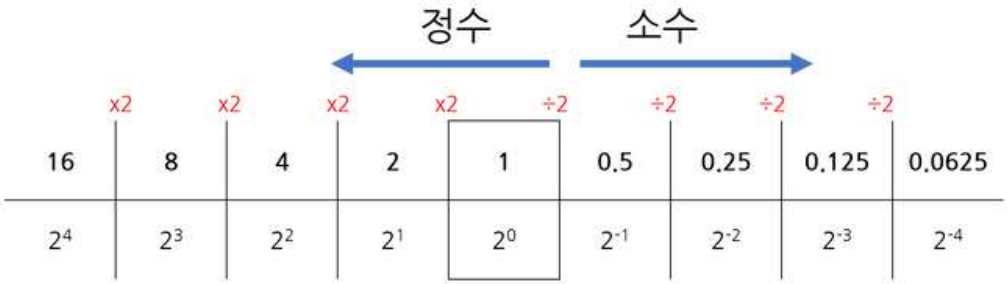
```
#Octal test, #Hex Test
octA = 0o77
hexA = 0xff12

print(octA, f'{hexA:0x}')
```

```
63 ff12
```

2진수, 8진수, 16진수, Floating Point

[Decimal to IEEE 754 Floating Point Representation - YouTube]

- 컴퓨터가 실수를 표현하는 방식

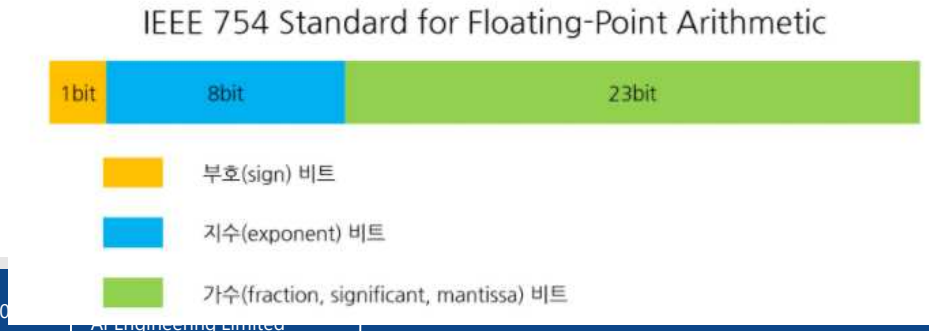


13=8+4+1
0.75 = 0.5+0.25

```
#floating Point Example
num = 0.0
for idx in range(0, 100):
    num += 0.1
print(num)
print(f'{num:.18f}')

tmp = 0.1 + 100
print(tmp)
print(f'{tmp:.18f}')
```

14.3 = 1110.0100110011001100 11...(0011 무한 반복)

- Floating Point
  - 고정 소수점(부호1bit, 정수 16bit, 소수 15bit)
    - (0)0000000000001110.0100110011001100
  - 부동 소수점



- 부호비트(1bit): 0 (양수)
- 지수비트(8bit): $1.110*2^3$ (127+3=130)
  - 10000010
- 가수비트(23bit): 1.0을 뺀 나머지
  - 11001001100110011001100...

파이썬 프로그래밍의 기초, 자료형

- 사칙연산 등

▪ 숫자형 활용하기 위한 연산자

▪ 사칙연산

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>> a * b
12
>>> a / b
0.75
```

▪ // 연산자 : 나눗셈 후 몫 반환

```
>>> 7 // 4
1
```

▪ ** 연산자 : 제곱

```
>>> a = 3
>>> b = 4
>>> a ** b
81
```

▪ % 연산자 : 나눗셈 후 나머지 반환

```
>>> 7 % 3
1
>>> 3 % 7
3
```

Practice #2 (연산자)

```
num1 = 3
num2 = 4.23

sumR = f'{num1+num2}'
print(sumR, f'{num1*num2}, {num1/num2}')

num1 = 7
num2 = 4

# 나누기 몫
print(num1//num2)

# **연산자 (승수 계산)
result = 2**8
print(result)

# % 연산자: 나눗셈 후 나머지 반환
print((8 % 3), (7 % 3))

# 소수 판단
testNum = 7

for i in range(2, testNum):
```

pycharm

3. 소수(Prime Number) 구하기
소수 : 약수가 1과 자기 자신 밖에 없는 수

```
# 소수 판단
testNum = 12
for i in range(2, testNum):
    if (testNum % i) == 0:
        print(f'{testNum} is not prime Number')
        break
    print(f'{testNum} is Prime Number')
```

파이썬 프로그래밍의 기초, 자료형

- 문자열 자료형
  - 문자, 단어 등으로 구성된 문자들의 집합

```
"Life is too short, You need Python"
"a"
"123"
```

▪ 문자열 사용법

▪ 문자열 만드는 방법

1. 큰따옴표(")

```
"Hello World"
```

2. 작은따옴표(')

```
'Python is fun'
```

3. 큰따옴표 3개(""")

```
"""Life is too short, You need python"""
```

4. 작은따옴표 3개(''')

```
'''Life is too short, You need python'''
```

파이썬 프로그래밍의 기초, 자료형

- 문자열 자료형
  - 문자, 단어 등으로 구성된 문자들의 집합

▪ 문자열 사용법

▪ 문자열 안에 작은따옴표나 큰따옴표를 포함시키기

1. 작은따옴표(')
   - 큰따옴표(")로 둘러싸기

```
>>> food = "Python's favorite food is perl"
```

2. 큰따옴표(")
   - 작은따옴표(')로 둘러싸기

```
>>> say = '"Python is very easy." he says.'
```

3. 백슬래시(₩) 활용하기
   - 백슬래시(₩) 뒤의 작은따옴표(')나 큰따옴표(")는 문자열을 둘러싸는 기호의 의미가 아니라 문자 ('), (") 그 자체를 의미

```
>>> food = 'Python\'s favorite food is perl'
>>> say = "\"Python is very easy.\" he says."
```

파이썬 프로그래밍의 기초, 자료형

- 문자열 자료형
  - 문자, 단어 등으로 구성된 문자들의 집합

▪ 문자열 사용법

▪ 여러 줄인 문자열을 변수에 대입하고 싶을 때

1. 이스케이프 코드 '₩n' 삽입

```
>>> multiline = "Life is too short\nYou need python"
```

2. 작은따옴표 3개(''')

```
>>> multiline = '''
... Life is too short
... You need python
... '''
```

3. 큰따옴표 3개(""")

```
>>> multiline = """
... Life is too short
... You need python
... """
```

Practice #3 (문자열)

```python
# string practice 1
str1 = "python is interpreter language\n"
str2 = '"python is very easy language to learn"'
print(str1, str2)
str1 = "We have to lean the \"python\""
str2 = """What happen if we use the 3 double single quotation"""
print(str1, str2)

# string practice 2
multiline = "Life is too short\nYou need python"
print(multiline)
```

```python
multiline = """
Life is too short,
U need python
"""
print(multiline)

multiline = '''
3 Single quotation is same as
3 double quotation!
'''
print(multiline)
```

파이썬 프로그래밍의 기초, 자료형

▪ 문자열 연산하기

1. 문자열 더해서 연결하기(Concatenation)

```
>>> head = "Python"
>>> tail = " is fun!"
>>> head + tail
'Python is fun!'
```

2. 문자열 곱하기

```
>>> a = "python"
>>> a * 2
'pythonpython'
```

3. 문자열 길이 구하기

- 파이썬 기본 내장 함수 len( )

```
>>> a = "Life is too short"
>>> len(a)
17
```

파이썬 프로그래밍의 기초, 자료형

▪ 문자열 인덱싱

- **인덱싱(Indexing)**
  - '가리킨다'는 의미
  - 파이썬은 0부터 숫자를 셈
  - a[번호]
    - 문자열 안의 특정 값 뽑아냄
    - 마이너스(-)
      - 문자열 뒤부터 셈

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0										1										2											3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3

```
>>> a = "Life is too short, You need Python"
>>> a[0]
'L'
>>> a[12]
's'
>>> a[-1]
'n'
```

```
a[0]: 'L', a[1]: 'i', a[2]: 'f', a[3]: 'e', a[4]: ' ', ···
```

▪ 문자열 슬라이싱

▪ 슬라이싱(Slicing)

- '잘라낸다'는 의미
- a[시작 번호:끝 번호]
  - 시작 번호부터 끝 번호까지의 문자를 뽑아냄
  - 끝 번호에 해당하는 것은 포함하지 않음

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0										1										2											3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3

```
>>> a = "Life is too short, You need Python"
>>> a[0:4]
'Life'
```

```
>>> a = "20010331Rainy"
>>> date = a[:8]
>>> weather = a[8:]
>>> date
'20010331'
>>> weather
'Rainy'
```

Practice #4 (문자열 연산, 인덱싱, 슬라이싱)

```
# String Operation
str1 = "Hi Everyone"
str2 = "My name is james"
str3 = "**********" # Comment Block
print(f'{str3*10}\n{str1+str2},\n'
      f'This Block is comment block\n'
      f'{str3*10} ')

# indexing/Slicing
lenStr2 = len(str2)
# j --> "J"
print(str2[-5])
#str2[-5] = "J" # Error
#str2 = str2[:-5]+"J"+str2[lenStr2-4:]
print(str2)

str2 = "20210308Sunny"
print('Today weather is ' + '\"' + str2[8:] + '\"')
```

▪ 문자열 포매팅

▪ 포매팅(Formatting)

1. 숫자 바로 대입
   - 문자열 포맷 코드 %d

```
>>> "I eat %d apples." % 3
'I eat 3 apples.'
```

```
>>> number = 3
>>> "I eat %d apples." % number
'I eat 3 apples.'
```

2. 문자열 바로 대입
   - 문자열 포맷 코드 %s

```
>>> "I eat %s apples." % "five"
'I eat five apples.'
```


파이썬 프로그래밍의 기초, 자료형

▪ 문자열 포매팅

▪ 문자열 포맷 코드

코드	설명
%s	문자열(String)
%c	문자 1개(Character)
%d	정수(Integer)
%f	부동 소수(Floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 '%' 자체)

```
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
'I ate 10 apples. so I was sick for three days.'
```

```
>>> "I have %s apples" % 3
'I have 3 apples'
>>> "rate is %s" % 3.234
'rate is 3.234'
```

▪ 문자열 포매팅

▪ 문자열 포맷 코드 활용법

1. 정렬과 공백

- %s를 숫자와 함께 사용하면, 공백과 정렬 표현 가능

```
>>> "%10s" % "hi"
'        hi' ◀── hi가 오른쪽 정렬됨
```


```
>>> "%-10sjane" % 'hi'
'hi        jane' ◀── hi가 왼쪽 정렬됨
```

▪ 문자열 포매팅

▪ 문자열 포맷 코드 활용법

2. 소수점 표현하기

- %f를 숫자와 함께 사용하면, 소수점 뒤에 나올 숫자의 개수 조절 및 정렬 가능

```
>>> "%0.4f" % 3.42134234
'3.4213'
```

```
>>> "%10.4f" % 3.42134234
'    3.4213'
```

▪ 문자열 포매팅

recommendation

▪ f 문자열 포매팅

- 파이썬 3.6 버전부터 f 문자열 포매팅 기능 제공
- 문자열 앞에 f 접두사를 붙이면, f 문자열 포매팅 기능 사용 가능

```
>>> name = '홍길동'
>>> age = 30
>>> f'나의 이름은 {name}입니다. 나이는 {age}입니다.'
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

```
>>> age = 30
>>> f'나는 내년이면 {age+1}살이 된다.'
'나는 내년이면 31살이 된다.'
```

Practice #5 (문자열 Formating)

```
number = 10
day = "three"
str1 = "I ate %d apples.\nSo I was sick for %s days." % (number, day)
print(str1)
str1 = f"I ate {number} apples.\nSo I was sick for {day} days."
print(str1)

str1 = "%4s\n%4s\n%4s\n%4s" % ("a", "ab", "abc", "abcd")
print(str1)

str1 = "%-10sjane" % ("hi")
print(str1, len(str1))

print(f"{3.42134234:0.4f}")
str2 = f"{3.42134234:10.3f}"
print(str2, len(str2))
```

▪ 문자열 관련 함수

- 문자열 자료형이 가진 내장 함수
- **count( )**
  - 문자 개수 세는 함수

```
>>> a = "hobby"
>>> a.count('b')
2
```

- **find( )**
  - 찾는 문자열이 처음 나온 위치 반환
  - 없으면 -1 반환

```
>>> a = "Python is the best choice"
>>> a.find('b')
14 ◀── 문자열에서 b가 처음 나온 위치
>>> a.find('k')
-1
```


▪ 문자열 관련 함수

▪ index( )

- find와 마찬가지로,
  찾는 문자열이 처음 나온 위치 반환
- 단, 찾는 문자열이 없으면 오류 발생

```
>>> a = "Life is too short"
>>> a.index('t')
8
>>> a.index('k')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

k가 없기 때문에 오류 발생

▪ join( )

- 문자열 삽입

```
>>> ",".join('abcd')
'a,b,c,d'
```

▪ upper( )

- 소문자를 대문자로 변환

```
>>> a = "hi"
>>> a.upper()
'HI'
```

▪ 문자열 관련 함수

▪ lower( )

- 대문자를 소문자로 변환

```
>>> a = "HI"
>>> a.lower()
'hi'
```

▪ lstrip( )

- 가장 왼쪽에 있는 연속된 공백 삭제

```
>>> a = " hi "
>>> a.lstrip()
'hi '
```

▪ rstrip( )

- 가장 오른쪽에 있는 연속된 공백 삭제

```
>>> a= " hi "
>>> a.rstrip()
' hi'
```

▪ strip( )

- 양쪽에 있는 연속된 공백 삭제

```
>>> a = " hi "
>>> a.strip()
'hi'
```


▪ 문자열 관련 함수

▪ replace( )

- replace(바뀌게 될 문자열, 바꿀 문자열)
- 문자열 안의 특정 값을 다른 값으로 치환

```
>>> a = "Life is too short"
>>> a.replace("Life", "Your leg")
'Your leg is too short'
```

▪ split( )

- 공백 또는 특정 문자열을 구분자로 해서 문자열 분리
  분리된 문자열은 리스트로 반환됨

```
>>> a = "Life is too short"
>>> a.split() ◀── 공백을 기준으로 문자열 나눔
['Life', 'is', 'too', 'short']
>>> b = "a:b:c:d"
>>> b.split(':') ◀── : 기호를 기준으로 문자열 나눔
['a', 'b', 'c', 'd']
```


Practice #6 (문자열 관련 함수)

```
str1 = """
Spread out before him that April day was the largest flotilla Communist-ruled China
 48 ships, dozens of fighter jets, more than 10,000 military personnel.
"""

cnt_a = str1.count("ships")
print(cnt_a)
print(str1.find("jets"), str1.find("h"), str1.index("h"), str1.find("korea"))
print(str1.upper())
print(str1.lower())

str1 = "12345"
print(",".join(str1))

str_list = "Life is too short"
print(str_list.replace("short", "long"))
print(str_list)

str_list = str_list.split()
print(str_list, type(str_list))

str_list = "    ".join(str_list)
print(str_list, type(str_list))
```

```
str_txt = "vehicle 0 0 50 50 vehicle 50 50 250 250"
```

Vehicle x y w h가 반복되는 데이터가 있을 경우,
해당 데이터에서 넓이가 100이상인 vehicle인 경우에는
Vehicle->truck으로 변경

C dll 연동: Hello World

- C:\windows\SysWOW64\msvcrt.dll (MS Visual C Runtime DLL)-> 현재 작업 directory에 복사

- dynamic-link library(동적 링크 라이브러리)

```
from ctypes import cdll
libc = cdll.LoadLibrary('msvcrt.dll')
libc.printf(b'hello world!\n')

listTmp = [1, 2, 3, 4, 5, 6, 7]
print(f'{len(listTmp)}')

for idx in listTmp:
    libc.printf(b"%d ", idx)
```

Thank you

Q&A

www.kopo.ac.kr
jsshin7@kopo.ac.kr