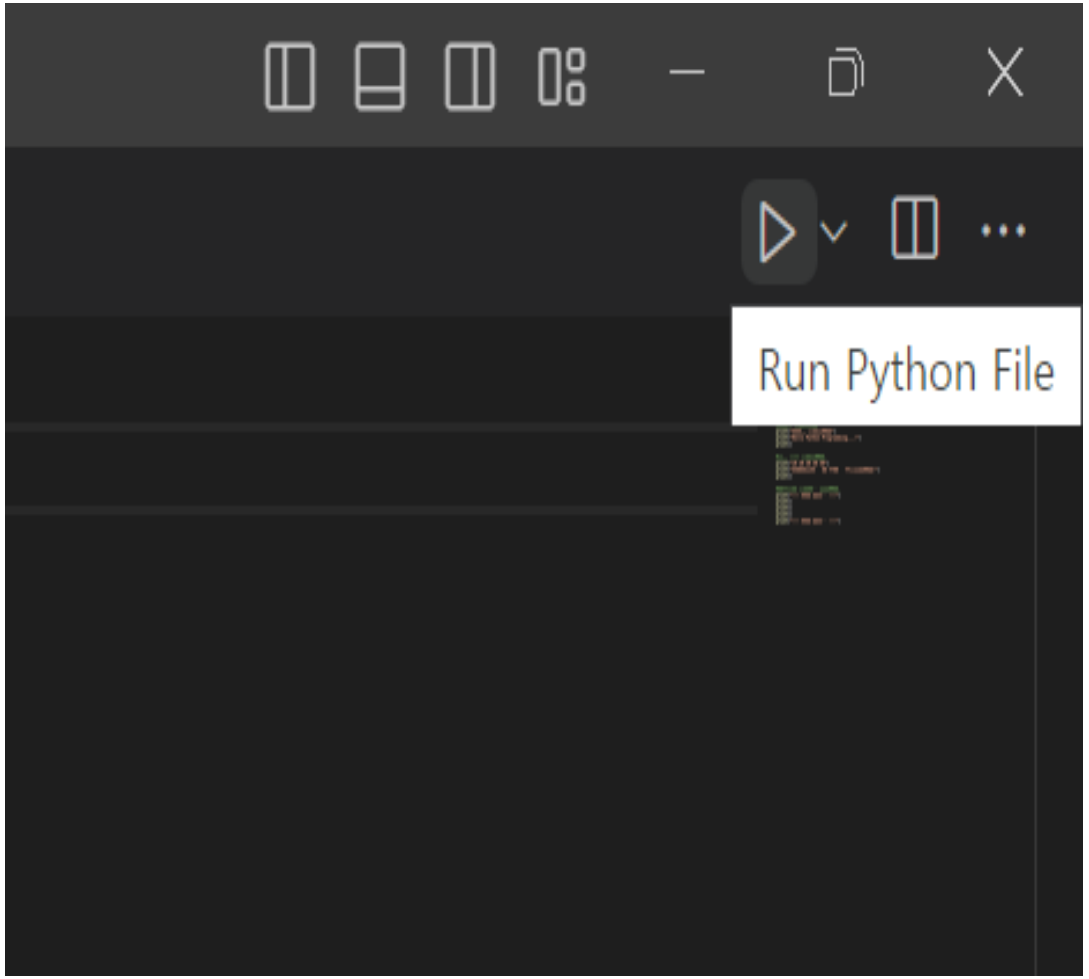
푸딩 1주차 스터디

1 파이썬 시작하기

2 자료형

출력 : print()



▪ 파이썬의 가장 기본적인 출력 방법
: print( )

▪ 괄호 안에 출력하고 싶은 것을 나열한 후,
위 오른쪽 부분의 ▷ 버튼을 누릅니다.

Part 1

주석처리

```
print("Hello World!")
print("주석처리 하는 법")
print("ctrl + /")

# print("Hello World!")
# print("주석처리 하는 법")
# print("ctrl + /")
```

- **주석 처리** 하는 방법 (2가지)
  ① 주석 처리하고 싶은 부분을 드래그 한 후,
  ctrl + / 을 누른다.
  ② 주석 처리하고 싶은 부분을 드래그 한 후,
  ctrl + k를 누른 상태에서 c 를 누른다.

- **주석 처리** 없애는 방법
  ① 주석 처리 없애고 싶은 부분을 드래그 한 후,
  ctrl + / 을 누른다.
  ② 주석 처리 없애고 싶은 부분을 드래그 한 후,
  ctrl + k를 누른 상태에서 u 를 누른다.

자료형

자료형과 문자열

- 자료형과 기본 자료형
- 문자열 만들기
-문자열 연산자

>>

숫자

-숫자 종류
-숫자 연산자

>>

변수와 입력

-변수 만들기/ 사용하기
-???
-사용자 입력: input()

>>

숫자와 문자열의 다양한 기능

-문자열: format() 함수
-대소문자 바꾸기: upper(), lower()
-문자열 양 옆 공백 제거: strip()
-문자열 구성파악: is000()
-문자열 찾기: find(), rfind()
-문자열과 in 연산자
-문자열 자르기: split()

자료형과 기본 자료형

- **문자열 : str**

예) “안녕하세요“, “Hello World”

- **정수 : int**

예) 1025, 273, 23

- **실수 : float**

예) 10.0, 9.35

- **불 : Boolean**

예) 친구의 로그인 상태 등 -> True, False

문자열 만들기

```
print("안녕하세요")



print('안녕하세요')
```

▪ **큰따옴표로 문자열 만들기**
: 문자열은 문자들을 큰따옴표(“)로 감싸서 만듭니다.

▪ **작은따옴표로 문자열 만들기**
: 작은따옴표(‘)로도 문자열을 만들 수 있습니다.

문자열 만들기

```
print('"안녕하세요"라고 말했습니다.')

print("'안녕하세요'라고 말했습니다.")

print("\"안녕하세요\"라고 말했습니다.")
```

▪ **큰따옴표를 문자열 내부에 넣기**
: 작은따옴표로 문자열을 만들면 됨

▪ **작은따옴표를 문자열 내부에 넣기**
: 작은따옴표로 문자열을 만들면 됨

▪ **이스케이프 문자를 사용**
: 이스케이프 문자 : 역슬래시(₩)기호와 함께 조합해서 사용하는 특수한 문자

- ₩” : 큰따옴표를 의미
- ₩’ : 작은따옴표를 의미

문자열 만들기

이스케이프 문자

- ₩n : 줄바꿈을 의미합니다.
- ₩t : 탭을 의미합니다.
- ₩₩ : 역슬래시를 의미합니다.

문자열 만들기

```
#여러 줄 문자열

print("""동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세
무궁화 삼천리 화려강산 대한사람
대한으로 길이 보전하세""")
```

▪ **여러 줄 문자열**

큰따옴표 또는 작은따옴표를 세 번 반복한 기호 사용

문자열 만들기

```
#줄바꿈 없이 문자열 만들기

print("""\
동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세
무궁화 삼천리 화려강산 대한사람
대한으로 길이 보전하세\
""")
```

▪ 줄바꿈 없이 문자열 만들기

파이썬에서는 '코드를 쉽게 보려고 줄바꿈한 것이지, 실질적인 줄바꿈이 아니다' 라는 것을 나타낼 때, 줄 뒤에 ₩ 기호를 사용

** ₩ 위치 주의!

문자열 연산자

```
#문자열 연산자

print("안녕" + "하세요")



print("김푸딩" * 3)
```

- “문자열” + “문자열”
- 문자열 반복 연산자 : *

문자열 연산자

```
# 문자 선택 연산자(인덱싱) : []

print("문자 선택 연산자에 대해 알아볼까요?")
print("안녕하세요"[0])
print("안녕하세요"[1])
print("안녕하세요"[2])
print("안녕하세요"[3])
print("안녕하세요"[4])
```

▪ **문자 선택 연산자(인덱싱) : [ ]**

-문자열 내부의 문자 하나를 선택하는 연산자

-대괄호 안에는 선택할 문자의 위치를 지정,
이 숫자를 인덱스라고 부름

-파이썬은 제로 인덱스를 씀

→제로 인덱스란? 숫자를 0부터 셈

안 녕 하 세 요

[0] [1] [2] [3] [4]

문자열 연산자

```
print("문자 선택 연산자 : 뒤에서부터 선택하기
")
print("안녕하세요"[-5])
print("안녕하세요"[-4])
print("안녕하세요"[-3])
print("안녕하세요"[-2])
print("안녕하세요"[-1])
```

▪ **문자 선택 연산자(인덱싱) : [ ]**

뒤에서부터 선택하기

안	녕	하	세	요
[-5]	[-4]	[-3]	[-2]	[-1]

문자열 연산자

```
#문자열 범위 선택 연산자(슬라이싱) : [:]
print("안녕하세요"[1:4])


#문자열 범위 선택 연산자
print("안녕하세요"[1:])
print("안녕하세요"[:3])
```

▪ 문자열 범위 선택 연산자(슬라이싱) : [:]

*** 마지막 숫자 포함하지 않음

ex) print("안녕하세요"[1:4])

→ 녕하세

문자열 범위 선택 연산자

뒤의 값을 생략 - 마지막글자까지

앞의 값을 생략 - 처음글자부터 지정

***인덱싱 : [ ] 기호를 이용해 문자열의 특정 위치에 있는 문자를 참조하는 것

***슬라이싱 : [:] 기호를 이용해 문자열의 일부를 추출하는 것

문자열 연산자

```
#문자열 길이 구하기 : len

print(len("김푸딩은귀엽다"))
```

- 문자열의 길이 구하기 : len

숫자의 종류

```
print(type(52))

print(type(143.0))
```

▪ **정수형(int)**
소수점이 없는 숫자 (0, 1, -143)
▪ **실수형(float)**
소수점이 있는 숫자 (0.0, 52.233...)

▪ **0이랑 0.0 구분하기**
print(type(0))
→ <class 'int'>
print(type(0.0))
→ <class 'float'>

숫자 연산자

```
#정수 나누기 연산자 //
print(3 / 2)
print(3 // 2)

#나머지 연산자 %
print(5 % 2)
```

- **사칙 연산자 : +, -, *, /**
- **정수 나누기 연산자 : //**

: 숫자를 나누고 소수점 이하의 자릿수를 떼어 버린 후, 정수 부분만 남기는 것

print(3 / 2)
→ 1.5
print(3 // 2)
→ 1

- **나머지 연산자 : %**
- **연산자의 우선순위**

: 사칙연산 순서와 같음

변수 만들기 & 사용하기

```
# 변수 선언과 할당
pi = 3.14159265
r = 10

# 변수 참조
print("원주율 =", pi)
print("반지름 =", r)
print("원의 둘레 =", 2*r*pi)
print("원의 넓이 =", r**2*pi)
```

- **변수를 활용하는 방법**
  1. 변수를 선언
  2. 변수에 값을 할당
  3. 변수를 참조

복합 대입 연산자

```
### 복합 대입 연산자
number = 100
number += 10
number += 20
number += 30
print("number =", number)
```

- **복합 대입 연산자**

자료형에 적용하는 기본 연산자와 = 연산자를 함께 사용

ex) a += 10

이것은 a = a+10 이라고 하는 것과 같은 결과

연산자 이름	설명
+=	숫자 덧셈 후 대입
-=	숫자 뺄셈 후 대입
*=	숫자 곱셈 후 대입
/=	숫자 나눗셈 후 대입
%=	숫자의 나머지를 구한 후 대입
**=	숫자 제곱 후 대입

문자열 복합 대입 연산자

```
### 문자열 복합 대입 연산자
string = "안녕하세요"
string += "! "
string *= 2
# 이렇게 두줄 따로따로 있는거면 사칙연산처럼 곱하기 먼저 있는게 아니고, 위에서부터 차례대로 됨
print("string:", string)
```

- **문자열 복합 대입 연산자**

문자열도 마찬가지로 복합 대입 연산자 사용 가능

연산자 이름	설명
+=	문자열 연결 후 대입
*=	문자열 반복 후 대입

사용자 입력 : input()

```
### 사용자 입력 : input()
input("인사말을 입력하세요 > ")

string = input("인사말을 입력하세요 > ")
print(string)
```

- **사용자 입력 : input()**
  input 함수 괄호 안에 입력한 내용
  : 프롬프트 문자열
  input 과 같이 함수의 결과로 나오는 값
  : 리턴값

input() 함수의 입력 자료형

```
#### input() 함수의 입력 자료형

print(type(string))

number = input("숫자를 입력하세요> ")
print(number)
print(type(number))
```

- **input() 함수의 입력 자료형**

print(type(string))
→ <class 'str'>

print(type(number))
→ <class 'str'>

input 함수는 사용자가 무엇을 입력해도 결과는 무조건 **문자열 자료형**

input() 함수의 입력 자료형

```
### 입력 자료형 확인하기

# 입력을 받습니다
string = input("입력> ")
# 출력합니다
print("자료:", string)
print("자료형:", type(string))
```

- **input() 함수의 입력 자료형**

print(type(string))
→ <class 'str'>

print(type(number))
→ <class 'str'>

input 함수는 사용자가 무엇을 입력해도 결과는 무조건 **문자열 자료형**

문자열을 숫자로 바꾸기

```
# 문자열을 숫자로 바꾸기

# int () 함수 활용
string_a = input("입력A > ")
#input 함수를 이 밑줄에서 int로 바꿔서 더하기가 가능하게
int_a = int(string_a)

string_b = input("입력B > ")
int_b = int(string_b)

print("문자열 자료:", string_a + string_b)
#진짜로 10 25 나란히 1025
print("숫자 자료: ", int_a + int_b)  #10 더하기 25
```

- **문자열을 숫자로 바꾸기**

-input() 함수의 입력 자료형은 항상 문자열이기 때문에, 입력받은 문자열을 숫자로 변환해야 숫자 연산에 활용할 수 있음
-영어로는 cast 라고 부름

int() 함수
: 문자열을 int 자료형으로 변환함(정수)

float() 함수
: 문자열을 float 자료형으로 변환함(실수)

int() 함수와 float() 함수 조합하기

```
# int 와 float 함수를 통과하고 나면 정말 자료형이 바뀔까?
output_a = int("52")
output_b = float("51.173")

print(type(output_a), output_a)
print(type(output_b), output_b)

# int() 함수와 float() 함수 조합하기
input_a = float(input("첫 번째 숫자> "))
input_b = float(input("두 번째 숫자> "))

print("덧셈 결과:", input_a + input_b)
#float 으로 해서 10이랑 25 더해도 35.0으로 나온다.
```

input_a = float(input("첫 번째 숫자> "))
input_b = float(input("두 번째 숫자> "))

*** print("덧셈 결과:", input_a + input_b)
#float 으로 해서 10이랑 25 더해도 35.0으로 나온다.

문자열 : format() 함수

```
# 문자열의 format() 함수

# format 함수로 숫자를 문자열로 변환하기
string_a = "{}".format(10)
## 숫자 10의 자료형은 문자열이 됨

#출력하기
print(string_a)
print(type(string_a))
```

1. 문자열의 format() 함수

# format 함수로 숫자를 문자열로 변환하기
string_a = "{}".format(10)

문자열 : format() 함수

```
# format() 함수의 다양한 형태

# format() 함수로 숫자를 문자열로 변환하기
format_a = "{}만 원".format(5000)
format_b = "{}개의 초콜릿".format(5000)
format_c = "{} {} {}".format(3000, 4000,
5000)
format_d = "{} {} {}".format(1, "문자열",
True)

# 출력하기
print(format_a)
print(format_b)
print(format_c)
print(format_d)
```

{}안에 + 붙이면 출력값도 + 붙어서 나옴

format()함수 : 정수를 특정 칸에 출력하기

```
# 정수를 특정 칸에 출력하기

# 정수
output_a = "{:d}".format(52)
## {:d} - int 자료형의 정수를 출력하겠다고 직접적으로 지정하는 것. 매개변수로 정수만 올 수 있음.

# 특정 칸에 출력하기
output_b = "{:5d}".format(52)
## 특정 칸에 맞춰서 숫자를 출력하는 형태. {:5d} - 5칸을 잡고 뒤에서부터 52라는 숫자를 채움.
output_c = "{:10d}".format(52)

# 빈칸을 0으로 채우기
output_d = "{:05d}".format(52)
## 빈칸을 0으로 채우는 형태. {:05d} - 5칸을 잡고 뒤에서부터 52라는 숫자를 넣은 후, 앞의 빈 곳을 0으로 채움.
output_e = "{:05d}".format(-52)
## - 부호가 있을 때 : 맨 앞자리를 부호로 채우고 나머지 빈 곳을 0으로 채움.
output_z = "{:05d}".format(+52)
## 부호가 있어도 +면 0으로 채워지는구나~
```

```
print("# 기본")
print(output_a)
print("# 특정 칸에 출력하기")
print(output_b)
print(output_c)
print("# 빈칸을 0으로 채우기")
print(output_d)
print(output_e)
print(output_z)
```

format()함수 : 기호 붙여 출력하기

```
# 기호 붙여 출력하기

output_f = "{:+d}".format(52)
# 양수 안에 + 붙이면 출력값도 + 붙어서 나옴
output_g = "{:+d}".format(-52)
# 음수 (-52)
output_h = "{: d}".format(52)
#양수: 기호 부분 공백 (52)
output_i = "{: d}".format(-52)
# 음수: 기호 부분 공백 (-52)

print("# 기호와 함께 출력하기")
print(output_f)
print(output_g)
print(output_h)
print(output_i)
```

format()함수:조합해보기

```
# 조합하기

output_h = "{:+5d}".format(52)
# 기호를 뒤로 밀기: 양수
output_i = "{:+5d}".format(-52)
# 기호를 뒤로 밀기: 음수
output_j = "{:=+5d}".format(52)
# 기호를 앞으로 밀기: 양수 / 기호와 공백을 조합할 때는 =
기호를 앞에 붙일 수 있음.
# 5칸의 공간을 잡았을 때 기호를 빈칸 앞에 붙일 것인지, 숫
자 앞에 붙일 것인지를 정하는 기호. 조합 순서 중요!!
output_k = "{:=+5d}".format(-52)
# 기호를 앞으로 밀기: 음수
output_l = "{:+05d}".format(52)
# 0으로 채우기: 양수
output_m = "{:+05d}".format(-52)
# 0으로 채우기: 음수
```

```
print("#조합하기")
print(output_h)
print(output_i)
print(output_j)
print(output_k)
print(output_l)
print(output_m)
```

format() 함수 : 부동 소수점 출력의 다양한 형태

```
# 부동 소수점 출력의 다양한 형태

# float 자료형 기본
output_a = "{:f}".format(52.273)
output_b = "{:15f}".format(52.273)   # 15칸 만들기
output_c = "{:+15f}".format(52.273)   # 15칸에 부호 추가하기
output_d = "{:+015f}".format(52.273)   # 15칸에 부호 추가하고 0으로 채우기

print(output_a)
print(output_b)
print(output_c)
print(output_d)
```

format() 함수 : 소수점 아래 자릿수 지정하기

```
# 소수점 아래 자릿수 지정하기

output_a = "{:15.3f}".format(52.273)
# 15칸 잡고 소수점을 각각 3자리로 출력함.
output_b = "{:15.2f}".format(52.273)
output_c = "{:15.1f}".format(52.273)

print(output_a)
print(output_b)
print(output_c)
```

- 이렇게 입력하면 15칸을 잡고 소수점을 각각 3자리, 2자리, 1자리로 출력함
- 이때 자동으로 반올림도 일어남

format() 함수 : 의미 없는 소수점 제거하기

```
# 의미 없는 소수점 제거하기 => {:g} 사용하면 됨

output_a = 52.0
output_b = "{:g}".format(output_a)

print(output_a)
print(output_b)
```

- 의미 없는 소수점 제거하기
→ **{:g}** 사용하면 됨

대소문자 바꾸기 : upper( ) & lower( )

```
# 대소문자 바꾸기 => upper () & lower()

a = "Hello Python Programming...!"
print(a.upper())
print(a)
# 비파괴적 함수라 원본은 변하지 않음
print(a.lower())
```

- **upper( ) 함수**
  문자열의 알파벳을 대문자로 만듦
- **lower( ) 함수**
  문자열의 알파벳을 소문자로 만듦
- **비파괴적 함수**
  원본을 변화시키지 않는 함수

문자열 양 옆 공백 제거 : strip()

```
# 문자열 양 옆의 공백 제거하기 => strip()
# 왼쪽 공백 제거 lstrip()
# 오른쪽 공백 제거 rstrip()

input_a = """
        안녕하세요
문자열의 함수를 알아봅니다
        """
print(input_a)

print(input_a.strip())
```

- strip( )
  문자열 양 옆의 공백을 제거
- lstrip( )
  문자열 왼쪽의 공백을 제거
- rstrip()
  문자열 오른쪽의 공백을 제거

문자열 찾기 : find() & rfind()

```
# 문자열 찾기 find() rfind()

output_a = "안녕안녕하세요".find("안녕")
print(output_a)
# 문자열 가장 앞글자 0이라서 0이라고 뜸

output_b = "안녕안녕하세요".rfind("안녕")
# rfind 오른쪽부터 찾아서 처음 등장하는 위치
를 찾음
print(output_b)
```

- find( )
  왼쪽부터 찾아서 처음 등장하는 위치를 찾음
- rfind( )
  오른쪽부터 찾아서 처음 등장하는 위치를 찾음

문자열과 in 연산자

```
# 문자열과 in 연산자

print("안녕" in "안녕하세요")

print("잘자" in "안녕하세요")
```

In 연산자의 출력은 **True**(맞다) 또는 **False**(아니다)라고 나옴

문자열 자르기 : split()

```
# 문자열 자르기 split() - 문자열을 특정한 문자로 자를 때 split()을 씀

a = "10 20 30 40 50".split(" ")
# split() 괄호 안의 문자열인 띄어쓰기를 기준으로 자름.
print(a)
# 실행결과대로 리스트가 나옴.
```

문자열을 **특정한 문자로 자를 때 split( )** 함수를 사용