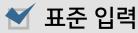


#### 학습개요와 학습목표







- 🗹 표준 출력



▼ 함수

- >> input()
  - int(), float()
- \_\_ \_ .

  >> 내장 함수 print()
- ≫ f-문자열
- >> 문자열 format() 메소드
- > 특정한 기능을 수행하는 프로그램 단위
- >> 정의와 호출
- > 람다 함수



- ☑ 표준입력 input()를 사용해 표준 입력을 저장할 수 있다.
- ☑ 표준출력을 위한 문자열을 f-문자열 또는 문자열 format() 메소드를 활용할 수 있다.
- ☑ 일반 함수와 람다 함수를 구현할 수 있다.

# LESSON 01

# 표준 입력과 출력





# → 표준 입력을 위한 함수 input()



```
# %% input()
     input()
     age = input('나이 입력 >> ')
     print(age)
  20
  나이 입력 >> 20
                            Vs code에서 노트북을 편집과
  20
                            실행한 화면으로 주피터의 실행
                            화면과 상이
     a = input('정수 입력 >> ')
     print(a, type(a))
    # print(a + 5) # 오류 발생
[38]
  정수 입력 >> 30
  30 <class 'str'>
```



# → 표준입력의 문자열을 변환

- **int()** 
  - ☑ 정수 변환
- float()
  - ☑ 실수 변환



# ⊸ 변환할 '정수나 실수' 형태의 문자열 확인



#### ValueError

```
int('300.4')
  float('3.61f')
ValueError
                                        Traceback (most recent call last)
Cell In[40], line 1
----> 1 int('300.4')
     2 float('3.61f')
ValueError: invalid literal for int() with base 10: '300.4'
   print(int(3.56))
   print(int(-3.4))
   print(float(3.4))
   print(float(3))
3
-3
3.4
3.0
```



# ⊸ 입력을 정수와 실수로 변환

# 

```
a = int(input('정수 입력 >> '))
      print(a, type(a))
      print(a + 5)
[19] 		 3.5s
   100 <class 'int'>
   105
      x = float(input('실수 입력 >> '))
      print(x, type(x))
      print(a + x)
   실수 입력 >> 4.6
   4.6 <class 'float'>
   14.6
```



### ⊸ 출력 문자열 생성



## 

```
# %% string format mathod
  print('개발환경 {}와 {}'.format('colab', 'codespace'))
  print('개발환경 {0}와 {1}'.format('colab', 'codespace'))
  print('개발환경 {1}와 {0}'.format('colab', 'codespace'))
  print('개발환경 {ide1}와 {ide2}'.format(ide1='colab', ide2='codespace'))
개발환경 colab와 codespace
개발환경 colab와 codespace
개발환경 codespace와 colab
개발환경 colab와 codespace
  print('개발환경 {}와 {}'.format('colab'))
                                       Traceback (most recent call last)
IndexError
Cell In[45], line 1
----> 1 print('개발환경 {}와 {}'.format('colab'))
IndexError: Replacement index 1 out of range for positional args tuple
```



## ⊸ f-문자열



# **→ 메소드 format()보다 활용도가 좋음**

☑ f'일반 문자열 <del>[연산식]</del> 일반 문자열'

```
# %% format string
   s = f'2026년 월드컵'
   print(s)
                                                                        Python
2026년 월드컵
   print(f'{3 * 3 * 3.14}')
   print(f'{3 ** 4}')
   print(f'{12 // 4}')
   print(f'{1.1 + 3.1}')
   print(f'{0b11 + 0xa}')
28.26
81
4.2
13
```



#### ⊸ 형식도 지정



## **❷** :05b => 앞에 0을 채우고 5자리의 2진수로

```
a = 16
  f'10진수: {a} 2진수: {a:05b} 8진수: {a:05o} 16진수: {a:05x}'
                                                                  Python
'10진수: 16 2진수: 10000 8진수: 00020 16진수: 00010'
  # %% format dict
  na = {'nation':'대한민국', 'capital':'서울'}
  f'국가: {na["nation"]} 수도: {na["capital"]}'
  f"국가: {na['nation']} 수도: {na['capital']}"
                                                                  Python
'국가: 대한민국 수도: 서울'
```



#### → 문자열 format() 메소드와 f-문자열

```
# %% format specifiers
   won, ex_rate = 150000, 1228.4
   print('{} 원은 $ {}'.format(won, won/ex_rate))
   print('{:d} 원은 $ {:10f}'.format(won, won/ex_rate))
   print('{:10d} 원은 $ {:.2f}'.format(won, won/ex_rate))
150000 원은 $ 122.110061869098
150000 원은 $ 122.110062
   150000 원은 $ 122.11
                                                         won, ex rate = 150000, 1478.2
   print(f'{won} 원은 {chr(8364)} {won/ex_rate}')
   print(f'{won:d} 원은 {chr(8364)} {won/ex_rate :10f}')
   print(f'{won:10d} 원은 {chr(8364)} {won/ex rate :.2f}')
150000 원은 € 101.4747666080368
150000 원은 € 101.474767
   150000 원은 € 101.47
```



## ⊸ 형식 인자



#### ☑ 좌측, 중앙, 우측 정렬

```
x = .123
   print(f'{x:8.1%} {x:.2%} {x:f}')
                                                                          Python
   12.3% 12.30% 0.123000
   # %%
   p1 = 'java'
   p2 = 'python'
   p3 = 'kotlin'
   print('1234567890' * 3)
   print(f'{p1:10s}{p2:10s}{p3:10s}')
   print(f'{p1:<10s}{p1:^10s}{p1:>10s}')
   print(f'{p2:<10s}{p2:^10s}{p2:>10s}')
123456789012345678901234567890
          python
                    kotlin
java
java
             java
                           java
python
            python
                        python
```

# LESSON 02

# 함수 정미와 호출





## ⊸ 함수 정의



# **❷** 특정한 기능을 수행하는 프로그램 단위

```
# %%
    print('안녕, 현수!')
    print('안녕, 수희!')
    print('안녕, 지수!')
    # %% 함수
    def greet():
        print('안녕, 현수!')
        print('안녕, 수희!')
        print('안녕, 지수!')
    greet()
안녕, 현수!
  안녕, 수희!
  안녕, 지수!
  안녕, 현수!
  안녕, 수희!
  안녕, 지수!
```



# → 함수의 반환 return



# **魦 없으면 None 반환**

```
# %%
    def greet(name):
       print(f'안녕, {name}!')
    greet('현수')
    greet('수희')
    greet('지수')
    greet('모든 친구들')
 안녕, 현수!
 안녕, 수희!
  안녕, 지수!
  안녕, 모든 친구들!
    # %% 반환 값 None
    def greet(name):
       print(f'안녕, {name}!')
    print(greet('모든 친구들'))
안녕, 모든 친구들!
  None
```



## ⊸ 함수의 반환 return

```
# %%
  area = 24
  print(f'{area} 평은 {area*3.3:.2f} 제곱미터')
  area = 32
  print(f'{area} 평은 {area*3.3:.2f} 제곱미터')
24 평은 79.20 제곱미터
32 평은 105.60 제곱미터
  # %%
  def toArea(pg):
      return pg * 3.3
  toArea(16)
  pg = 24
  print(f'{pg} 평은 {toArea(pg):.2f} 제곱미터')
  pg = 32
  print(f'{pg} 평은 {toArea(pg):.2f} 제곱미터')
24 평은 79.20 제곱미터
32 평은 105.60 제곱미터
```



- ⊸ 함수도 객체
- **⋧ 자료형 Class function**
- id(obj)
  - ☑ 객체 obj의 메모리 주소

```
# 함수 정의
  def hello():
      print('안녕')
  print(type(hello))
<class 'function'>
  # 정의된 함수 주소 출력
  print(hello)
  print(id(hello))
<function hello at 0x0000028B61237F60>
2797653426016
```

#### ☑ 함수 정의와 호출



# ⊸ 객체인 함수



```
# 함수 호출
   print(hello())
  # 함수 대입
  my_hello = hello
  my_hello()
   id(my_hello), id(hello)
✓ 0.0s
안녕
None
안녕
(2797653426016, 2797653426016)
```



# → 람다 함수: 익명 함수





# ⊸ 람마 함수도 객체



```
my_hello = lambda : print('안녕')
my_hello()

> 0.0s

Python

print(my_hello, type(my_hello))
print(my_hello())

print(my_hello())

**Gunction <lambda> at 0x0000028B612E4680> <class 'function'>
안녕
None
```



# → 람다 함수 정의와 호출

```
# 람다 함수 정의
pyeong_to_ms = lambda pg: f'{pg} 평은 {pg*3.3:.2f} 제곱미터'

# 람다 함수 호출
print(pyeong_to_ms(24))
print(pyeong_to_ms(32))

** 24 평은 79.20 제곱미터
32 평은 105.60 제곱미터
```

# SUMMARY

# 학습정긴





•••

## 🔅 표준 입력과 출력

- >> input('prompt message')
  - > int(), float()
- f-string
  - > f'string1 {exp1} string2 {exp2}'
- >> 문자열 format() 메소드
  - > 'string1 {} string2 {}'.format(exp1, exp2)







•••

- 👸 함수 구현과 호출
  - >> def func(param, …):
    - **■** > ····
    - > return ···
  - >> func(argument, ···)
- 🔅 람다 함수
  - > lambda x, y: x + y



