In [1]:

```
# for youtube display
from IPython.display import YouTubeVideo
```

함수(function) 기초

들어가기전에

> 직사각형의 둘레와 면적을 구하는 코드를 작성해주세요.

```
height = 30
width = 20
```

예시 출력)
직사각형 둘레: 100, 면적: 600입니다.

In [7]:

```
height = 30
width = 20
# 아래에 코드를 작성하세요.
area = width * height
perimeter = 2 * (width + height)
print(f'직사각형 둘레: {perimeter}, 면적: {area}')
```

```
직사각형 둘레: 100, 면적: 600
```

- 앞서 작성한 코드에서 매번 사각형의 둘레와 면적을 구하기 위해서는 변수에 값을 바꾸거나 코드를 복사 붙여넣기 해야합니다.
- 코드가 많아질수록 문제가 발생할 확률이 높아지며, 유지 보수하기도 힘들어진다.

In [6]:

```
# 위의 코드를 함수로 아래와 같이 작성할 수 있습니다.
def rectangular(width, height):
    area = width * height
    perimeter = 2 * (width + height)
    print(f'직사각형 둘레: {perimeter}, 면적: {area}')

rectangular(20, 30)
rectangular(100, 40)
```

```
직사각형 둘레: 100, 면적: 600
직사각형 둘레: 280, 면적: 4000
```

개요

활용법

```
def func(parameter1, parameter2):
```

```
    code line1
    code line2
    return value
```

- 함수 선언은 `def` 로 시작하여 `:` 으로 끝나고, 다음은 `4spaces 들여쓰기` 로 코드 블록을 만듭니다.
- 함수는 `매개변수(parameter)` 를 넘겨줄 수도 있습니다.
- 함수는 동작후에 `return` 을 통해 결과값을 전달 할 수도 있습니다. (`return` 값이 없으면, None을 반환합니다.)

In [ ]:

```
# 우리가 활용하는 print문도 파이썬에 지정된 함수입니다.
# 아래에서 'hi'는 parameter이고 출력을 하게 됩니다.
print('hi')
```

출처: python 공식문서

In [8]:

```
# 내장함수 목록을 직접 볼 수도 있습니다.
dir(__builtins__)
```

Out[8]:

```
['ArithmeticError',
 'AssertionError',
 'AttributeError',
 'BaseException',
 'BlockingIOError',
 'BrokenPipeError',
 'BufferError',
 'BytesWarning',
 'ChildProcessError',
 'ConnectionAbortedError',
 'ConnectionError',
 'ConnectionRefusedError',
 'ConnectionResetError',
 'DeprecationWarning',
 'EOFError',
 'Ellipsis',
 'EnvironmentError',
 'Exception',
 'False',
 'FileExistsError',
 'FileNotFoundError',
 'FloatingPointError',
 'FutureWarning',
 'GeneratorExit',
 'IOError',
 'ImportError',
 'ImportWarning',
 'IndentationError',
 'IndexError',
 'InterruptedError',
 'IsADirectoryError',
 'KeyError',
 'KeyboardInterrupt',
 'LookupError',
 'MemoryError',
 'ModuleNotFoundError',
 'NameError',
 'None',
 'NotADirectoryError',
 'NotImplemented',
```

```
'NotImplementedError',
'OSError',
'OverflowError',
'PendingDeprecationWarning',
'PermissionError',
'ProcessLookupError',
'RecursionError',
'ReferenceError',
'ResourceWarning',
'RuntimeError',
'RuntimeWarning',
'StopAsyncIteration',
'StopIteration',
'SyntaxError',
'SyntaxWarning',
'SystemError',
'SystemExit',
'TabError',
'TimeoutError',
'True',
'TypeError',
'UnboundLocalError',
'UnicodeDecodeError',
'UnicodeEncodeError',
'UnicodeError',
'UnicodeTranslateError',
'UnicodeWarning',
'UserWarning',
'ValueError',
'Warning',
'WindowsError',
'ZeroDivisionError',
'__IPYTHON__',
'__build_class__',
'__debug__',
'__doc__',
'__import__',
'__loader__',
'__name__',
'__package__',
'__spec__',
'abs',
'all',
'any',
'ascii',
'bin',
'bool',
'bytearray',
'bytes',
'callable',
'chr',
'classmethod',
'compile',
'complex',
'copyright',
'credits',
'delattr',
'dict',
'dir',
'display',
'divmod',
'enumerate',
'eval',
'exec',
'filter',
'float',
'format',
'frozenset',
'get_ipython',
'getattr',
'globals',
'hasattr',
'hash',
'help',
'hex',
'id',
'input',
```

```
 'int',
 'isinstance',
 'issubclass',
 'iter',
 'len',
 'license',
 'list',
 'locals',
 'map',
 'max',
 'memoryview',
 'min',
 'next',
 'object',
 'oct',
 'open',
 'ord',
 'pow',
 'print',
 'property',
 'range',
 'repr',
 'reversed',
 'round',
 'set',
 'setattr',
 'slice',
 'sorted',
 'staticmethod',
 'str',
 'sum',
 'super',
 'tuple',
 'type',
 'vars',
 'zip']
```

함수의 return

앞서 설명한 것과 마찬가지로 함수는 반환되는 값이 있으며, 이는 어떠한 종류의 객체여도 상관없습니다.

단, 오직 한 개의 객체만 반환됩니다.

함수가 return 되거나 종료되면, 함수를 호출한 곳으로 돌아갑니다.

실습문제

> 아래의 코드와 동일한 `my_max` 함수를 만들어주세요.
>
> 정수를 두개 받아서, 큰 값을 반환합니다.
>
> (`sum()` 사용 금지.)

```
max(1, 5)
```

예상 출력)
5

In [9]:

```
max(1,5)
```

Out[9]:

5

In [15]:

```
# 여기에 my_max 함수를 만들어주세요.
def my_max(a, b):
    if a > b:
        return a
    else:
        return b

my_max(7, 3)
```

Out[15]:

```
7
```

함수의 인수

함수는 `인자(parameter)`를 넘겨줄 수 있습니다.

위치 인수

함수는 기본적으로 인수를 위치로 판단합니다.

In [18]:

```
# 함수 예시
def my_func(a, b, c):
    return a*b + c
my_func(1, 2, 3)
```

Out[18]:

```
5
```

기본 값(Default Argument Values)

함수가 호출될 때, 인자를 지정하지 않아도 기본 값을 설정할 수 있습니다.

활용법

```
def func(p1=v1):
    return p1
```

In [1]:

```
# 기본 값이 있는 함수 예시
def greeting(name='ssafy'):
    print(f'{name}, 안녕!')

greeting()
greeting('철수')
```

```
ssafy, 안녕!
철수, 안녕!
```

- 기본 인자 값이 설정되어 있더라도 기존의 함수와 동일하게 호출 가능합니다.
- 호출시 인자가 없으면 기본 인자 값이 활용됩니다.
- 단, 기본 매개변수 이후에 기본 값이 없는 매개변수를 사용할 수는 없습니다.

In [24]:

```
def greeting(name='ssafy', age):
    print(f'{name}은 {age}살 입니다.')
```

```
  File "<ipython-input-24-c264f29f23f8>", line 1
    def greeting(name='ssafy', age):
                 ^
SyntaxError: non-default argument follows default argument
```

In [26]:

```
def greeting(age, name='ssafy'):
    print(f'{name}은 {age}살 입니다.')

greeting(1)
greeting(2, '싸피싸피')
```

```
ssafy은 1살 입니다.
싸피싸피은 2살 입니다.
```

키워드 인자(Keyword Arguments)

키워드 인자는 직접적으로 변수의 이름으로 특정 인자를 전달할 수 있습니다.

In [28]:

```
# 키워드 인자 예시
def greeting(age, name='ssafy'):
    print(f'{name}은 {age}살 입니다.')

greeting(name='철수', age='24')
greeting(24, name='철수')
```

```
철수은 24살 입니다.
철수은 24살 입니다.
```

- 단 아래와 같이 활용할 수는 없습니다. 키워드 인자를 활용한 뒤에 위치 인자를 활용할 수는 없습니다.

In [29]:

```
greeting(age=24, '철수')
```

```
  File "<ipython-input-29-d457a6f39e1c>", line 1
    greeting(age=24, '철수')
                     ^
SyntaxError: positional argument follows keyword argument
```

우리가 주로 활용하는 `print()` 함수는 파이썬 표준 라이브러리의 내장함수 중 하나이며, 다음과 같이 구성되어 있다.

In [32]:

```
print('hi', end='!')
```

```
hi!
```

가변 인자 리스트

앞서 설명한 `print()` 처럼 정해지지 않은 임의의 숫자의 인자를 받기 위해서는 가변인자를 활용합니다.

가변인자는 `tuple` 형태로 처리가 되며, `*` 로 표현합니다.

활용법

```
def func(*args):
```

In [35]:

```
# 가변 인자 예시 (print문은 *obejcts를 통해 임의의 숫자의 인자를 모두 처리합니다.)
print('hi', '안녕', 'guten tak', 'nihao', sep=', ')
```

```
hi, 안녕, guten tak, nihao
```

In [38]:

```
# args는 tuple!
def my_func(*args):
    print(type(args))
my_func(1, 2)
```

```
<class 'tuple'>
```

실습문제

> 정수를 여러 개 받아서 가장 큰 값을 반환(return)하는 `my_max()` 을 만들어주세요. 단, `sum()` 사용 금지.

```
my_max(10, 20, 30, 50)
```

예시출력)
50

In [ ]:

```
max(1,2,3,4)
```

In [41]:

```
# 아래에 코드를 작성해주세요.
def my_max(*args):
    max_num = args[0]
    for i in range(1, len(args)):
        if max_num < args[i]:
            max_num = args[i]
    print(max_num)

my_max(1, 2, 3, 4)
```

```
4
```

정의되지 않은 인자들 처리하기

정의되지 않은 인자들은 `dict` 형태로 처리가 되며, `**` 로 표현합니다.

주로 `kwagrs` 라는 이름을 사용하며, `**kwargs` 를 통해 인자를 받아 처리할 수 있습니다.

활용법

```
def func(**kwargs):
```

우리가 dictionary를 만들 때 사용할 수 있는 `dict()` 함수는 파이썬 표준 라이브러리의 내장함수 중 하나이며, 다음과 같이 구성되어

있다.

In [42]:

```
# 딕셔너리 생성 함수 예시
dict(한국어='안녕',영어='hi')
```

Out[42]:

```
{'한국어': '안녕', '영어': 'hi'}
```

In [45]:

```
# fake_dict
def fake_dict(**kwargs):
    print(type(kwargs))
    for kw in kwargs:
        print(kw, ' : ', kwargs[kw])

fake_dict(한국어='안녕',영어='hi')
```

```
<class 'dict'>
한국어  :  안녕
영어  :  hi
```

실습문제

> 앞선 fake_dict는 단순히 dictionary 형태로 print를 합니다.
>
> `my_dict()` 함수를 만들어 실제로 dictionary를 반환하는 함수를 만들어보세요.

In [51]:

```
def my_dict(**kwargs):
#     for kw in kwargs:
#         kwargs[kw] = kwargs[kw]
    return kwargs

my_dict(한국어='안녕',영어='hi')
```

Out[51]:

```
{'한국어': '안녕', '영어': 'hi'}
```

dictionary를 인자로 넘기기(unpacking arguments list)

`**dict`를 통해 함수에 인자를 넘길 수 있습니다.

In [52]:

```
# user 검증(유사 로그인)
def user(username, password, password_confirmation):
    if password == password_confirmation:
        print(f'{username}님 회원가입이 완료되었습니다.')
    else:
        print('비밀번호와 비밀번호 확인이 일치하지 않습니다.')
```

In [55]:

```
my_account = {
                'username' : '서호성',
                'password' : '1234',
                'password_confirmation' : '1234'
}
```

In [56]:

```
user(my_account)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-56-8e3e907cd536> in <module>
----> 1 user(my_account)

TypeError: user() missing 2 required positional arguments: 'password' and 'password_confirmation'
```

In [57]:

```
user(**my_account)
```

서호성님 회원가입이 완료되었습니다.

이름공간 및 스코프(Scope)

파이썬에서 사용되는 이름들은 이름공간(namespce)에 저장되어 있습니다. 그리고, LEGB Rule을 가지고 있습니다.

변수에서 값을 찾을 때 아래와 같은 순서대로 이름을 찾아나갑니다.

- `L`ocal scope: 정의된 함수
- `E`nclosed scope: 상위 함수
- `G`lobal scope: 함수 밖의 변수 혹은 import된 모듈
- `B`uilt-in scope: 파이썬안에 내장되어 있는 함수 또는 속성

In [ ]:

```
# 따라서 첫시간에 내장함수의 식별자를 사용할 수 없었던 예제에서 오류가 생기는 이유를 확인할 수 있습니다.
```

In [58]:

```
str = '4'
str(3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-58-f5cdd7b7f4bb> in <module>
      1 str = '4'
----> 2 str(3)

TypeError: 'str' object is not callable
```

- `str()` 코드가 실행되면
- str을 Global scope에서 먼저 찾아서 `str = 4`를 가져오고,
- 이는 함수가 아니라 변수이기 때문에 `not callable`하다라는 오류를 내뱉게 됩니다.
- 우리가 원하는 `str()`은 Built-in scope에 있기 때문입니다.

In [59]:

```
del str
```

In [62]:

```
# print(a)에 무엇이 출력되는지 확인해보세요.
a = 1
def localscope(a):
    print(a)

localscope(3)
```

```
3
```

네임스페이스 원칙

- def 내에 할당된 이름들은 오직 그 def 코드에 의해서만 보인다. 함수 외부에서는 그 이름을 확인조차 할 수 없다.
- def 내에 할당괸 이름들은 비록 동일한 이름이 다른 곳에서 사용되더라고 def 바깥의 변수들과 충돌하지 않는다. 즉 def 문 안에 할당된 이름 x는 def 밖에서 할당된 x와는 전혀 다르다.

- `지역(local) 범위` : 변수가 def문 안에 할당되면 해당 함수에 대하여 지역범위를 갖는다.
- `전역(global) 변수` : 변수가 모든 def문의 바깥에서 할당되면 이는 전체 파일에 대한 전역변수다.
- `비지역(nonlocal) 변수` : 변수가 바깥 def 안에서 할당되면 이는 중첩된 함수에 대한 비지역 변수다.

```
apple = 100         # 전역 범위

def func():
    apple = 100     # 지역 범위 (apple 은 서로 다른 변수다.)
```

- 결국 네임스페이스 계층은 프로그램에서 이름 충돌을 피하도록 예방하고, 함수가 자립된 프로그램 단위로 구성되도록 한다.

In [65]:

```
# 전역 변수를 바꿀 수 있을 까요?
global_num = 3
def localscope2():
    global_num = 5
    print(f'global_num이 {global_num}로 설정되었습니다.')

localscope2()
print(global_num)
```

```
global_num이 5로 설정되었습니다.
3
```

In [66]:

```
# 굳이 전역에 있는 변수를 바꾸고 싶다면, 아래와 같이 선언할 수 있습니다.
global_num = 3
def localscope2():
    global global_num
    global_num = 5
    print(f'global_num이 {global_num}로 설정되었습니다.')

localscope2()
print(global_num)
```

```
global_num이 5로 설정되었습니다.
5
```

- def문 내에 할당된 모든 이름은 기본적으로 지역 범위를 갖기 때문에, 바깥쪽(enclosing) 함수와 전역범위 이름들을 사용하기 위해서는 global 또는 nonlocal 이름을 선언해주어야 한다.

In [68]:

```
# scope youtube embed
# from IPython.display import YouTubeVideo
YouTubeVideo('yH_1vwnp3ZQ', width='100%')
```

Out[68]:

이름공간은 각자의 수명주기가 있습니다.

- built-in scope : 파이썬이 실행된 이후부터 끝까지
- Global scope : 모듈이 호출된 시점 이후 혹은 이름 선언된 이후부터 끝까지
- Local/Enclosed scope : 함수가 실행된 시점 이후부터 리턴할때 까지