

Raspberry Pi를 활용한 IoT 프로젝트

파이썬 기초 명령어 익히기

3일차

담당교수 : 조도은

<https://github.com/DoEunCho/raspberrypi>



1일차 : 라즈베리파이 소개와 환경 구축(3H)

2일차 : 라즈베리 파이를 위한 리눅스 기초 배우기(3H)

3일차 : 파이썬 기초 명령어 익히기(3H)

4일차 : 라즈베리 파이 GPIO와 센서 동작하기(3H)

5일차 : 나만의 가상비서 만들기(구글 어시스턴트)(3H)



▪ Python 기초 명령어 익히기

- ▣ Python 시작하기
- ▣ Python 자료형
- ▣ Python 제어문
- ▣ Python 함수 사용법
- ▣ Python 클래스와 모듈



파이썬 개요

- 비영리재단인 파이썬 소프트웨어 재단이 관리하는 **인터프리터 방식**(직접 해석 바로 실행 방식)의 객체 지향 언어
- 인터프리터 형식은 **한 번에 한 줄씩 소스코드를 읽어서 기계어로 변환 후 바로 실행**

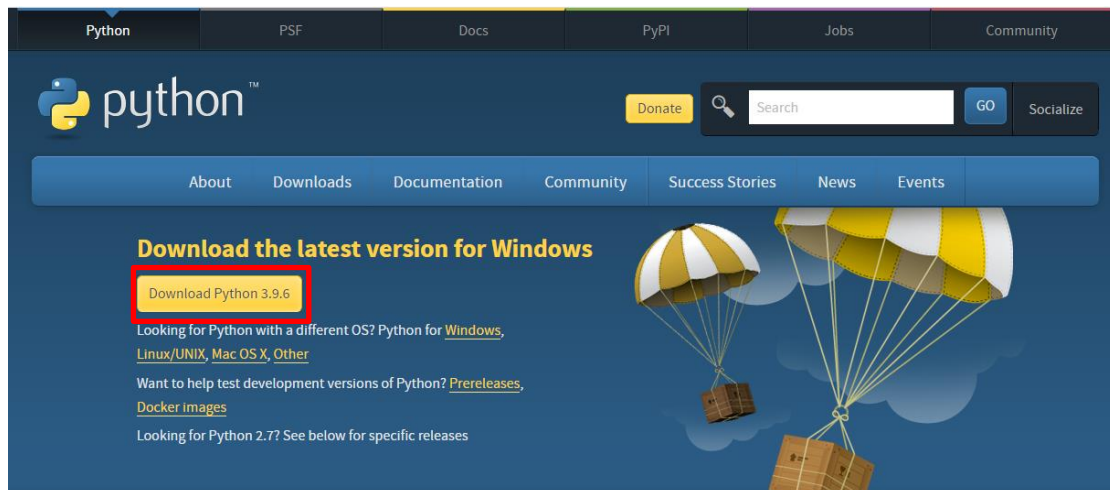
파이썬의 특징

- 초보자도 쉽게 배울 수 있음
- 문법이 다른 언어에 비해 간단하고 가독성이 뛰어남
- 다양한 라이브러리 지원
- 라이브러리를 이용하면 짧은 프로그램 작성이 가능



파이썬 설치하기

- 파이썬 윈도우용 설치 파일을 다운로드하기 위해 파이썬 다운로드 페이지에 접속한 후 [Download Python]버튼을 클릭한다.
- 파이썬 홈페이지 : <https://www.python.org/downloads/>



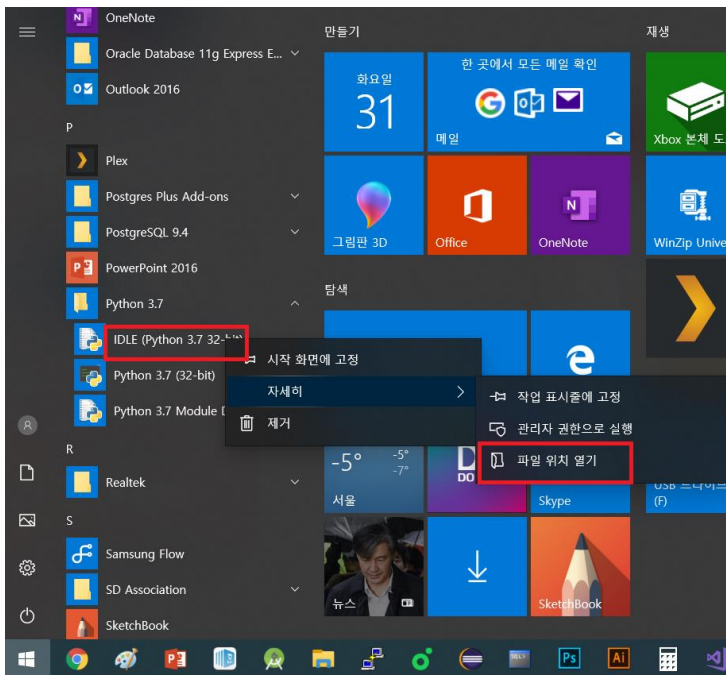


Add Python 3.9 to PATH를 체크하고 [Install Now]를 클릭한다.





- 설치가 완료되면 Windows 프로그램에서 그림과 같이 **IDLE(Python 3.9)** 파일 위치 열기를 한다.



IDLE(Python 3.9) 파일 위치 열기



파이썬 실행하기

- IDLE Python을 실행하고 간단한 테스트를 한다.

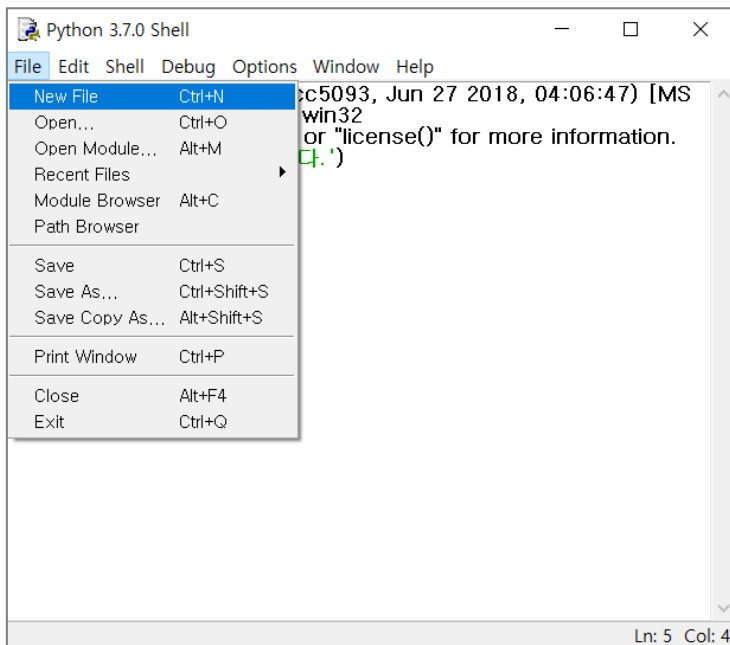
```
>>> print("출력 테스트 입니다.")
```

```
IDLE Shell 3.9.6
File Edit Shell Debug Options Window Help
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win
32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("출력 테스트 입니다.")
출력 테스트 입니다.
>>> |
```

Ln: 5 Col: 4



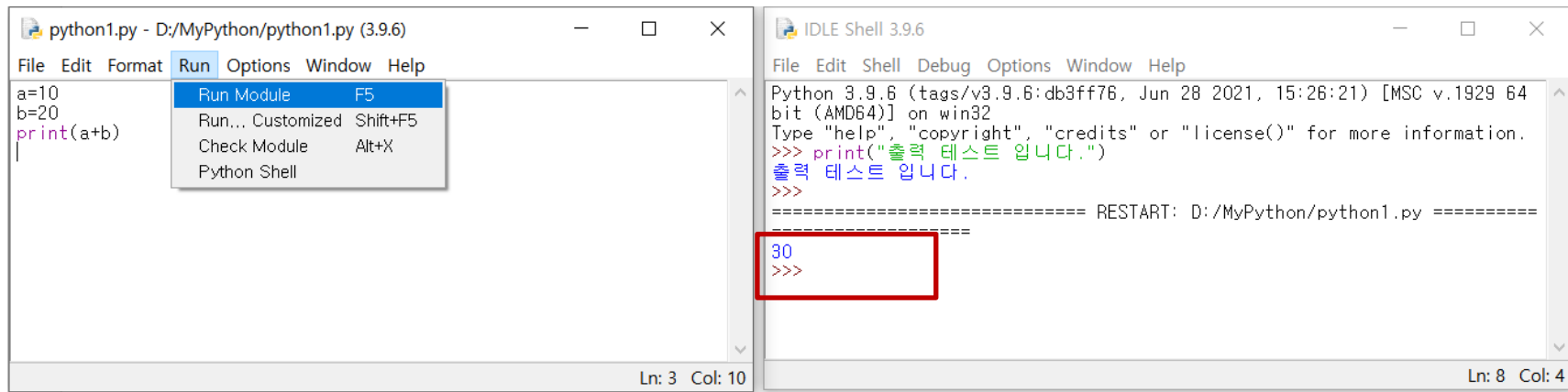
- IDLE에서 새로운 파일(New File)을 선택한다.





- 간단한 코드를 입력하고 그림과 같이 저장 후에 실행한다.

```
a = 10  
b = 20  
print(a+b)
```





- Python에서 사용하는 숫자, 문자열 등 모든 자료 형태를 자료형이라 한다.
- Python에서 변수의 자료형을 확인하고 싶을 때는 `type()`라는 함수를 사용하면 된다.
- 파이썬의 자료형
 - 숫자형
 - 문자열 자료형
 - 리스트 자료형
 - 튜플 자료형
 - 딕셔너리 자료형



■ 숫자형

- 정수 및 실수를 표현하는 타입이다.
- 다음은 숫자형 변수를 선언하고 출력하는 예제이다.

```
01 a = 123
02 print(type(a))
03 a = 100*100
04 print(a)
05 a, b, = 9, 2
06 print(a*b)
```

```
<class 'int'>
10000
18
>>>
```

01 : 정수 a에 123을 저장한다.
02 : a가 어떤 자료형 타입인지 출력을 한다.
03 : a에 100 * 100을 저장한다.
04 : a값을 출력한다.
05 : a는 9, b는 2의 값을 저장한다.
06 : a * b의 값을 출력한다.

<실행 결과>



■ 문자형

- 문자열을 표현하는 타입이다.
- 다음은 두 가지 방법으로 문자열 변수를 선언하는 예제이다.

```
str01.py - C:\Users\User\Desktop
File Edit Format Run Options
a = "파이썬 만세"
print(a)
print(type(a))
b = 'python go'
print(b)
```

01 : a를 문자열 타입으로 쌍 따옴표를 사용하여 선언
04 : b를 문자열 타입으로 홑 따옴표를 사용하여 선언



■ 리스트

- 리스트는 여러 개의 값들을 하나의 변수로 묶어서 사용하는 타입이다.
- 다음은 리스트를 사용하지 않고 숫자형 변수 4개를 선언하여 입력 받은 값들의 합을 출력한다.

```
a, b, c, d = 0, 0, 0, 0
hap = 0
a = int(input("1번째 숫자 : "))
b = int(input("2번째 숫자 : "))
c = int(input("3번째 숫자 : "))
d = int(input("4번째 숫자 : "))
hap = a + b + c + d

print("합계 ==> %d" % hap)
```

<실행결과>

```
1번째 숫자 : 10
2번째 숫자 : 20
3번째 숫자 : 30
4번째 숫자 : 40
합계 ==> 100
>>> |
```



■ 리스트

- 다음은 리스트 변수를 선언하여 list01.py와 같은 기능을 만들어 보는 예제이다.

```
aa = [0, 0, 0, 0]
hap = 0
aa[0] = int(input("1번째 숫자 : "))
aa[1] = int(input("2번째 숫자 : "))
aa[2] = int(input("3번째 숫자 : "))
aa[3] = int(input("4번째 숫자 : "))
hap = aa[0] + aa[1] + aa[2] + aa[3]

print("합계2 ==> %d" % hap)
```

<실행결과>

```
1번째 숫자 : 1
2번째 숫자 : 2
3번째 숫자 : 3
4번째 숫자 : 4
합계2 ==> 10
>>> |
```



■ 리스트

- 다음은 리스트에 **append()** 함수를 이용하여 값을 추가하고 리스트내용을 출력하는 예제이다.

```
File Edit Format Run Opt
aa = []
aa.append(0)
aa.append(0)
aa.append(0)
aa.append(0)
print(len(aa))
print(aa)
bb = []
for i in range(0, 100):
    bb.append(i)
print(bb)
```

<실행결과>

```
4
[0, 0, 0, 0]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54,
55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
94, 95, 96, 97, 98, 99]
>>> |
```




■ 리스트

- 다음은 리스트에서 특정 위치(index)에 해당하는 값을 출력 예제이다.

```
File Edit Format Run Options Window Help
1 aa = [10, 20, 30, 40]
2 print("aa[-1]=%d, aa[-2]=%d" %(aa[-1], aa[-2]))
3
4 print(aa[0:2]) #aa리스트의 0번 인덱스에서 2번 인덱스 앞까지 출력
5 print(aa[2:4]) #aa리스트의 2번 인덱스에서 4번 인덱스 앞까지 출력
6 print(aa[0:]) #aa리스트의 0번 인덱스부터 출력
7 |
```

<실행결과>

```
aa[-1]=40, aa[-2]=30
[10, 20]
[30, 40]
[10, 20, 30, 40]
```



■ 리스트

- 다음은 리스트에서 다양하게 사용할 수 있는 함수를 사용하는 예제이다.

```
File Edit Format Run Options Window Help
1 aa = [30, 10, 20]
2
3 # aa리스트를 문자열로 출력
4 print("현재의 리스트 : %s" % aa)
5
6 aa.append(40) # aa리스트에 40을 추가
7 print("append후의 리스트 : %s" % aa)
8
9 aa.pop() # aa리스트의 마지막 값 꺼내기
10 print("pop후의 리스트 : %s" % aa)
11
12 aa.sort() # aa리스트의 값 정렬하기
13 print("sort후의 리스트 : %s" % aa)
14
15 aa.reverse() # aa의 리스트의 값 역순으로 정렬
16 print("reverse후의 리스트 : %s" % aa)
17
18 aa.insert(2, 22) # aa 리스트의 2번 인덱스에 22값을 삽입
19 print("insert후의 리스트 : %s" % aa)
20
21 print("20값의 위치 : %d" %aa.index(20)) # 20값의 인덱스 번호 출력
22
23 aa.remove(22) # 22의 값을 삭제
24 print("remove후의 리스트 : %s" % aa)
25
26 aa.extend([77, 88, 77]) # 다른 리스트를 덧붙여 확장하기
27 print("extend후의 리스트 : %s" % aa)
28
29 print("77값의 개수 : %d" %aa.count(77)) #77의 개수 출력하기
30
31
```

<실행결과>

현재의 리스트 : [30, 10, 20]
 append후의 리스트 : [30, 10, 20, 40]
 pop후의 리스트 : [30, 10, 20]
 sort후의 리스트 : [10, 20, 30]
 reverse후의 리스트 : [30, 20, 10]
 insert후의 리스트 : [30, 20, 22, 10]
 20값의 위치 : 1
 remove후의 리스트 : [30, 20, 10]
 extend후의 리스트 : [30, 20, 10, 77, 88, 77]
 77값의 개수 : 2



■ 리스트

- 다음은 2차원 리스트를 선언하고, 리스트 요소 값을 출력하는 예제이다.

<실행결과>

```
File Edit Format Run Options Window Help
1 aa = [[1,2,3,4],
2       [5,6,7,8]
3       [9,10,11,12]]
4
5 print(aa[0][0]) # 0행 0열의 값 출력
6
7 print(aa[0]) # 0행의 값 모두 출력
8
9 print(aa[1][2]) # 1행 2열의 값 출력
10
```

```
1
[1, 2, 3, 4]
7
```



- if문

- if문은 '만약 지정한 조건에 맞으면 지정한 코드를 실행하라' 는 의미이다.

```
if 조건 :  
    조건이 참일 경우 실행 문장
```

```
if 조건 :  
    조건 참일 경우 실행 문장  
else :  
    조건이 거짓일 경우 실행 문장
```

```
if 조건1 :  
    조건1 참일 경우 실행 문장  
elif 조건2 :  
    조건2가 참일 경우 실행 문장  
elif 조건3 :  
    조건3가 참일 경우 실행 문장  
else :  
    모든 조건이 거짓일 경우 실행 문장
```



- 다음은 if문 및 else문 그리고 elif문 설명을 위한 예제이다.

```
File Edit Format Run Options Window Help
#if문
a = 23
if a < 50:
    print('50보다 작군요')
#if else문
if a < 20:
    print('20보다 작군요')
else:
    print('20보다 크군요')
#elif문
age = int(input('현재 나이를 입력하세요. '))
if age < 10:
    print('유년층 입니다.')
elif age < 20:
    print('10대입니다.')
elif age < 30:
    print('20대입니다.')
elif age < 40:
    print('30대입니다.')
else:
    print('장년층입니다')
```

실행결과

50보다 작군요
20보다 크군요
현재 나이를 입력하세요.



▪ for문

- for문은 특정 코드를 지정한 횟수만큼 반복하는 문장이다.

```
for 변수 in 리스트 혹은 튜플 :  
    실행 문장
```

```
for 변수 in range(시작숫자, 종료숫자, step):  
    실행 문장
```



for문

- 다음은 for문을 다양한 형태로 선언한 예제이다.

File Edit Format Run Options Window Help

```
for i in range(0, 5, 1):  
    print(i)  
    print("-----")  
for j in [1,3,5,7,9]:  
    print(j)  
    print("-----")  
for k in range(0, 3, 1):  
    print("꿈은 이루어 진다.")
```

실행결과

Python 3.7.0 Shell

File Edit Shell Debug Options Wind

Python 3.7.0 (v3.7.0:1bf9cc509
Type "copyright", "credits" or "li

>>>

RESTART: C:\Users\User\De
rryPi4_source\ch03\for01.py

0
1
2
3
4

1
3
5
7
9

꿈은 이루어 진다.
꿈은 이루어 진다.
꿈은 이루어 진다.

>>>



■ for문

- 다음은 for문을 이용하여 1부터 10까지의 합을 구하는 예제이다.

```
File Edit Format Run Options Window Help
#for문을 이용하여 1에서 10까지 합을 구하시오.
sum = 0
for i in range(1, 11, 1):
    sum+=i
print("sum : %d" % sum)
print("-----")
#for문을 이용하여 1에서 10까지 식과 합을 구하시오.
sum = 0
for j in range(1, 11, 1):
    if j<10:
        print("%d + " % j, end="")
    elif j==10:
        print("%d = " % j, end="")
    sum+=j
print("%d" % sum)
```

실행결과

```
sum : 55
-----
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55
>>>
```




■ while문

- while문은 조건을 지정하고, 조건이 true일 때 실행되며, 조건이 false가 될 때까지 반복한다. 따라서 조건이 false가 되지 않으면 무한루프(무한반복)가 발생하게 된다.
- 일반적인 while문 선언과 while문 안에 if문을 중첩으로 선언한 예제이다.

```
File Edit Format Run Options Window Help
str = "꿈은 이루어 진다."
i = 0
while i < 3:
    print(str)
    i = i + 1
print("-----")
#while문으로 입력한 숫자만큼 str을 반복 출력하시오.
i = int(input("반복 횟수 숫자를 입력하세요. "))
j = 1
flag = True
while flag:
    j = j + 1
    if i < j:
        flag = False
    print(str)
```

실행결과

```
꿈은 이루어 진다.
꿈은 이루어 진다.
꿈은 이루어 진다.
-----
반복 횟수 숫자를 입력하세요. 5
꿈은 이루어 진다.
꿈은 이루어 진다.
꿈은 이루어 진다.
꿈은 이루어 진다.
꿈은 이루어 진다.
>>>
```



■ 함수

```
def Name (매개변수 리스트) :  
    함수 명령문  
    return result
```

- 두 수 중 큰 수를 반환하는 함수를 작성하는 예

```
def max(x, y) :  
    if x > y:  
        result = x  
    else :  
        result = y  
    return result
```

- 함수 호출하기

```
z = max (6, 10)
```



■ 함수 선언과 호출하기

- 다음은 기본적인 함수를 선언하고 간단하게 함수를 호출하는 예제이다.

```
File Edit Format Run Options Window Help
def mydef01():
    print("함수를 선언합니다.")
mydef01()
def mydef02(str="인수함수를 선언합니다."):
    print(str)
mydef02()
mydef02("인수를 넣습니다.")
```

실행결과

```
함수를 선언합니다.
인수함수를 선언합니다.
인수를 넣습니다.
>>>
```



■ 함수 인수 사용하기

- 다음은 여러 개의 인수를 입력 받아서 계산을 하는 함수를 선언하는 예제이다.

```
File Edit Format Run Options Window Help
def mydef01():
    i = 10
    j = 20
    print(i+j)
mydef01()
def mydef02(i, j):
    print(i+j)
mydef02(10, 20)
# 계산할 숫자를 두 개 입력하세요.
def mydef03(i, j, p):
    if p == '+':
        print(i+j)
    elif p == '-':
        print(i-j)
    elif p == '*':
        print(i*j)
    elif p == '/':
        print(i/j)
n = int(input("첫 번째 숫자를 입력하세요. "))
m = int(input("두 번째 숫자를 입력하세요. "))
p = input("연산자를 입력하세요(+, -, *, /)")
mydef03(n, m, p)
```

실행결과

```
30
30
첫 번째 숫자를 입력하세요. 30
두 번째 숫자를 입력하세요. 10
연산자를 입력하세요(+, -, *, /)*
300
>>>
```



■ 클래스

- 클래스(class)는 프로그래밍 과정에서 객체를 정의하는 데이터와 이를 활용하는 기능을 가질 수 있는 구조를 의미한다.
- 각 클래스는 객체의 상태를 정의할 수 있는 속성(attributes)과 객체의 기능을 정의하는 메서드(methods)를 가질 수 있는 구조이다.
- 객체의 속성과 객체의 기능인 메소드가 있는 클래스를 선언하고 그 클래스를 사용하여 객체를 생성 하는 예제이다.



■ 클래스

File Edit Format Run Options Window Help

```
class Automobile:
    name = ""
    velocity = 0
    def velocityPlus(self):
        self.velocity = self.velocity + 1
        print("속도는 %d 입니다." % self.velocity)
    def velocityDw(self):
        self.velocity = self.velocity - 1
        if self.velocity < 0:
            self.velocity = 0
        print("속도는 %d 입니다." % self.velocity)
ac = Automobile()
ac.velocityPlus()
ac.velocity = 20
ac.velocityDw()
```

실행결과

```
속도는 1 입니다.
속도는 19 입니다.
>>>
```



■ 모듈

- 모듈은 미리 작성된 함수 코드를 모아 놓은 Python 파일이다.
- 파이썬에서는 모듈화를 통해서 미리 구현된 다양한 라이브러리를 사용할 수 있다.
- 또는 직접 모듈을 개발 할 수도 있으며 Python 개발환경이 기본적으로 제공하고 있는 다양한 Python 모듈을 사용할 수 있다.
- 다음은 간단한 일반 함수 두 개를 포함하고 있는 모듈을 만드는 예제이다.
코드를 작성하고 module01.py로 저장한다.

```
File Edit Format Run Options Window
def mydef01():
    print("일반 함수입니다.")
def mydef02(n, m):
    print(n*m)
```



■ 모듈

- 다음은 일반 함수 두 개를 선언한 module01.py 모듈을 사용하는 예제이다.

File Edit Format Run Options Window Help

```
import module01
import sys
import math
module01.mydef01()
module01.mydef02(10, 20)
print("-----")
print(sys.builtin_module_names)
print(round(3.14))
```

실행결과

일반 함수입니다.
200

```
-----
('_abc', '_ast', '_bisect', '_blake2', '_codecs', '_codecs_cn', '_codecs_hk', '_codecs_iso2022',
'_codecs_jp', '_codecs_kr', '_codecs_tw', '_collections', '_csv', '_datetime', '_functools', '_
heapq', '_imp', '_io', '_json', '_locale', '_lspref', '_md5', '_multibytecodec', '_opcode', '_oper
ator', '_pickle', '_random', '_sha1', '_sha256', '_sha3', '_sha512', '_signal', '_sre', '_stat', '_
string', '_struct', '_symtable', '_thread', '_tracemalloc', '_warnings', '_weakref', '_winapi', '_arr
ay', '_atexit', '_audioop', '_binascii', '_builtins', '_cmath', '_errno', '_faulthandler', '_gc', '_itertools', '_m
arshal', '_math', '_mmap', '_msvcrt', '_nt', '_parser', '_sys', '_time', '_winreg', '_xxsubtype', '_zipimport
', '_zlib')
3
>>>
```

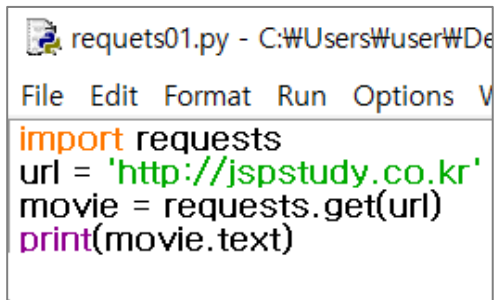



■ 라이브러리

- 라이브러리는 모듈들을 모아놓은 곳으로, 파이썬을 설치하면 자동적으로 내장 모듈을 제공하지만 외부에서 제공되는 모듈도 존재한다.
- 외부에서 제공되는 모듈을 사용하려면 외부 라이브러리 설치를 해야 한다.
- **requests**라는 외부 라이브러리는 웹에 요청을 해서 데이터를 가져오는 역할을 하는 라이브러리인데 설치하려면 아래 명령어를 실행한다.

pip3 install requests

- 다음은 설치된 requests 라이브러리를 불러온 뒤, 지정한 url로 부터 요청 정보를 받아 출력한다



```
reqeuts01.py - C:\Users\Wuser\De
File Edit Format Run Options V
import requests
url = 'http://jspstudy.co.kr'
movie = requests.get(url)
print(movie.text)
```

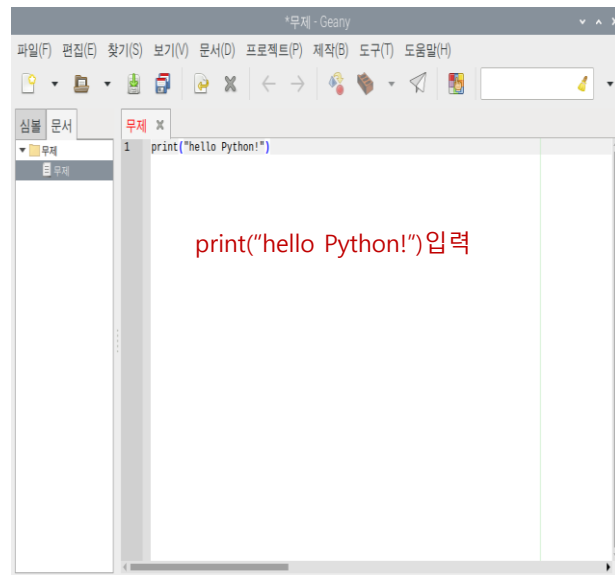
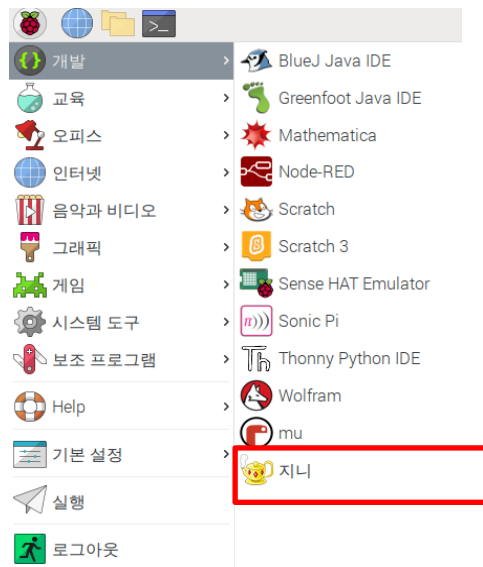


- 파이썬 버전 확인하기

~\$**python -V**

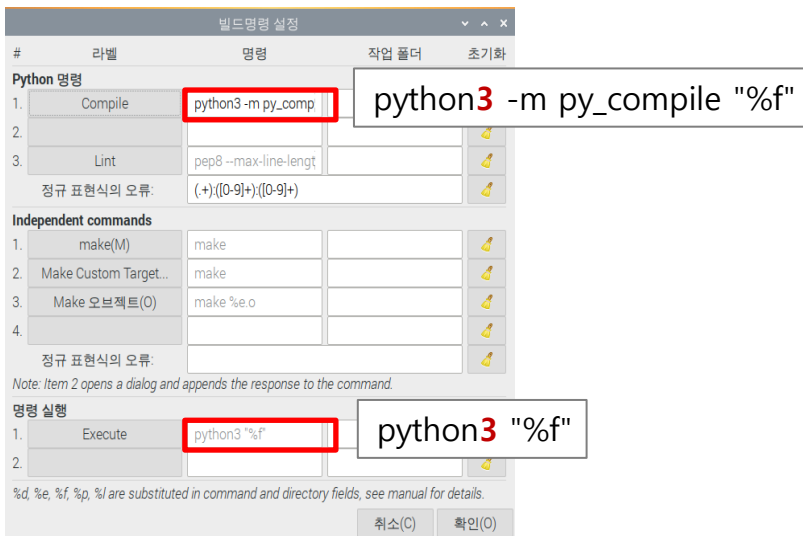
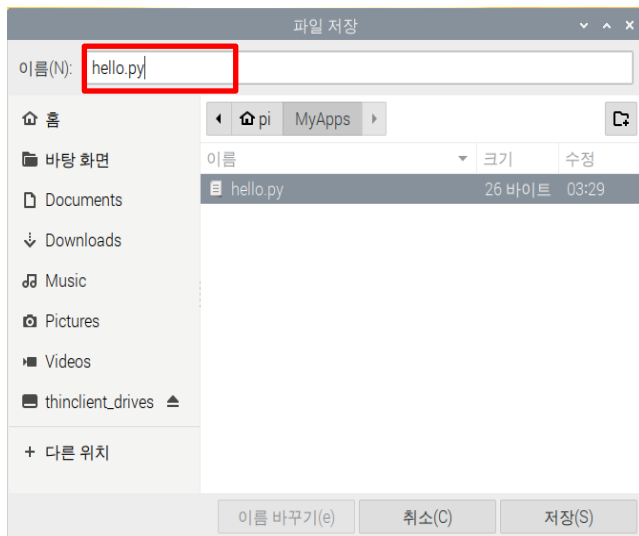
~\$**python3 -V**

- 개발-지니 실행
- 간단한 hello.py
파일 작성하기





- 다른 이름으로 저장 -> MyApps 폴더 생성 후 -> hello.py로 저장하기
- 제작 -> 빌드명령설정-> 버전 3으로 변경



Raspberry Pi를 활용한 IoT 프로젝트

Thank You