인공지능 로봇

파이보와 함께하는

파이썬 코딩 가이드

2024.05.08

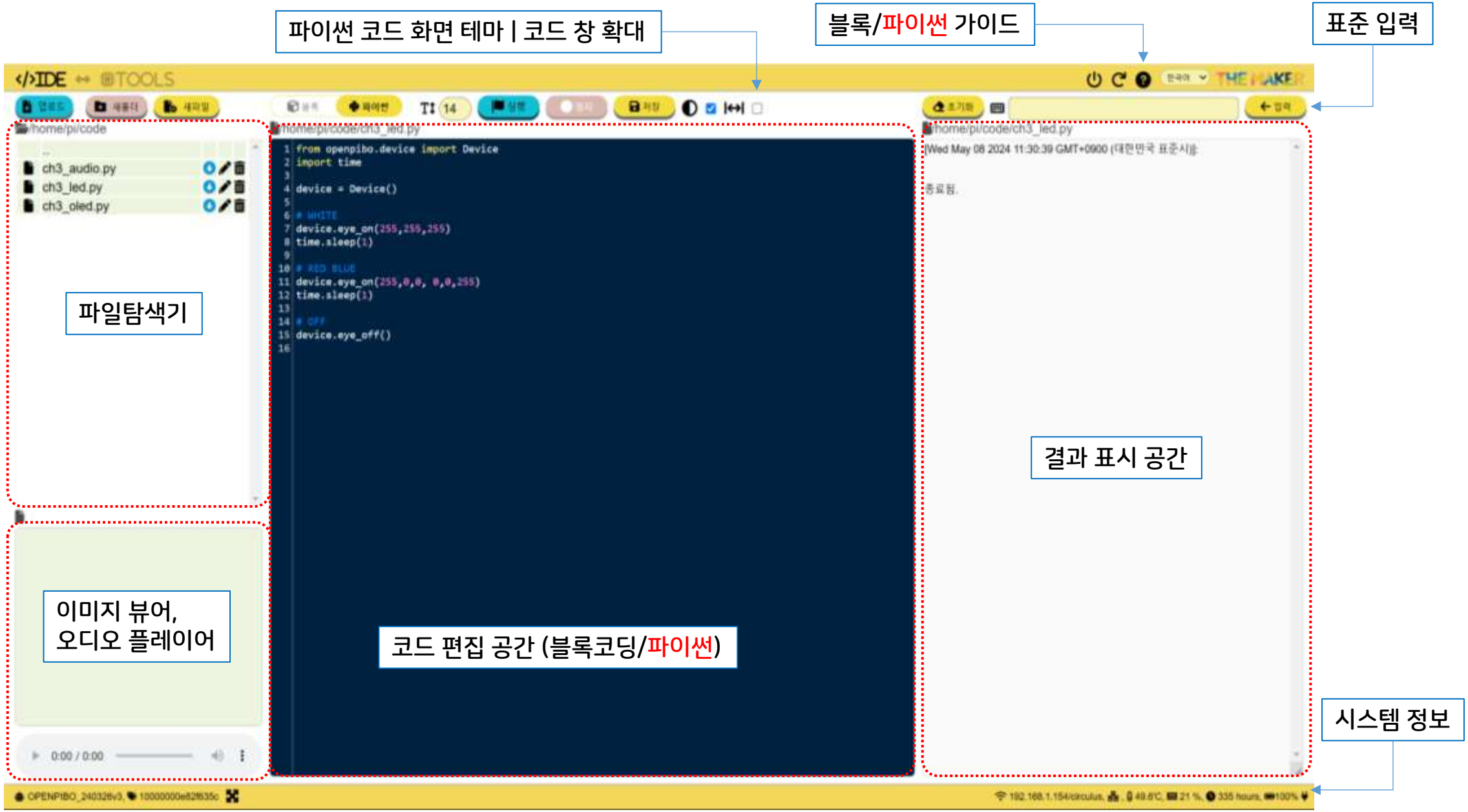
파이보 메이커 파이썬 코딩

파이보 메이커 파이썬 코딩

- 파이썬 코딩 및 개발 가능
  - 파일 탐색기, 이미지/오디오 확인, 표준 입출력
- 파이썬 기본 코딩, 컴퓨터 비전, 인공지능 코딩 가능
  - 논리/반복/수학/문자/목록/변수/함수 등 프로그래밍 파이썬 기본 코딩

  (블록코딩 가이드와 연계할 수 있도록 같은 순서 구성)
  - opencv, dlib, tensorflow 등 활용도 높은 패키지를 활용
- openpibo-python 패키지를 활용해 파이보 로봇 / 파이브레인 사용 가능
  - 전용 파이썬 패키지를 제공하고, 공개하여 쉽게 기기를 활용
  - 자체 인공지능 클라우드 서버를 활용하여 음성합성/인식 등을 사용할 수 있도록 함
  - https://themakerrobot.github.io/openpibo-python/build/html/index.html

파이보 메이커 IDE 개발 환경



파이썬 기본 코딩 - 논리

파이썬 기본 코딩 – 논리 (예제1)

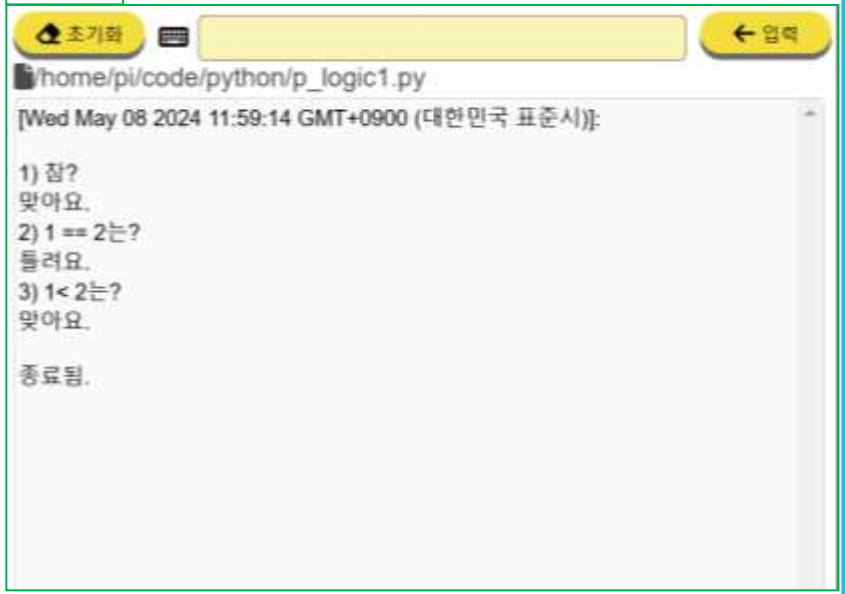
b_logic1.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_logic1.py

```
1  print('1) 참?')
2  if True:
3    print('맞아요.')
4
5  print('2) 1 == 2는?')
6  if 1 == 2:
7    print('맞아요.')
8  else:
9    print('틀려요.')
10
11 print('3) 1< 2는?')
12 if 1 < 2:
13   print('맞아요.')
14 else:
15   print('틀려요.')
```

결과



파이썬 기본 코딩 – 논리 (예제2)

b_logic2.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_logic2.py

```
1  print('1) 참 and 참')
2  if True and True:
3    print('맞아요.')
4  else:
5    print('틀려요.')
6
7  print('2) 참 and 거짓')
8  if True and False:
9    print('맞아요.')
10 else:
11   print('틀려요.')
12
13 print('3) 거짓 or 참')
14 if False or True:
15   print('맞아요.')
16 else:
17   print('틀려요.')
18
19 print('4) 거짓 or 거짓')
20 if False or False:
21   print('맞아요.')
22 else:
23   print('틀려요.')
24
25 print('5) 한줄 - 참')
26 print('맞아요.' if True else '틀려요.')
```

결과

초기화 | 입력

/home/pi/code/python/p_logic2.py

```
[Wed May 08 2024 12:05:46 GMT+0900 (대한민국 표준시)]:

1) 참 and 참
맞아요.
2) 참 and 거짓
틀려요.
3) 거짓 or 참
맞아요.
4) 거짓 or 거짓
틀려요.
5) 한줄 - 참
맞아요.

종료됨.
```

파이썬 기본 코딩 – 논리 (예제3)

b_logic3.py

블록 파이썬 14 실행 정지 저장

/home/pi/code/python/p_logic3.py

```
1  result = '맞아요' if 1 + 1 == 10 else '틀려요'
2  print(result)
3
4  number = 15
5  if number < 10:
6    print('10보다 작아요')
7  elif number < 20:
8    print('20보다 작아요')
9  elif number < 30:
10   print('30보다 작아요')
11 else:
12   print('30보다 커요')
```

결과

초기화 입력

/home/pi/code/python/p_logic3.py

[Wed May 08 2024 12:14:19 GMT+0900 (대한민국 표준시)]:

틀려요
20보다 작아요

종료됨.

파이썬 기본 코딩 - 반복

파이썬 기본 코딩 – 반복 (예제1)

b_loop1.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_loop1.py

```
1 print('1) 3회 반복')
2 for count in range(3):
3   print('abc')
4
5 print('2) 10 에서 13까지 반복')
6 for i in range(10, 14):
7   print(i)
```

결과

초기화 | 입력

/home/pi/code/python/p_loop1.py

[Wed May 08 2024 12:21:45 GMT+0900 (대한민국 표준시)]:

```
1) 3회 반복
abc
abc
abc
2) 10 에서 13까지 반복
10
11
12
13

종료됨.
```

파이썬 기본 코딩 – 반복 (예제2)

b_loop2.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_loop2.py

```
1  print('1) 반복 / 중단')
2  num = 1
3  while True:
4    print(num)
5    if num > 3:
6      print('반복을 중단합니다.')
7      break
8    num = num + 1
9
10 print('2) 목록 조회')
11 for item in ['사과', '배', '바나나']:
12   print(item)
```

결과

초기화 | 입력

/home/pi/code/python/p_loop2.py

```
[Wed May 08 2024 12:23:19 GMT+0900 (대한민국 표준시)]:

1) 반복 / 중단
1
2
3
4
반복을 중단합니다.
2) 목록 조회
사과
배
바나나

종료됨.
```

파이썬 기본 코딩 - 수학

파이썬 기본 코딩 – 수학 (예제)

b_math.json

블록 | 파이썬 | T↕ 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_math.py

```
1  import math
2  import random
3
4  print(1 + 1)
5  print(math.sqrt(9))
6  print(0 % 2 == 0)
7  print(round(3.1))
8  print(64 % 10)
9  print(random.randint(1, 100))
10 print(sum([1, 2, 3]))
11 print(min(max(200, 1), 100))
```

연산

숫자 체크: 홀/짝수, 소수,정수,양수,음수 등
반올림/올림/버림
나머지
랜덤 정수 추출
합계
최대/최소값

결과

초기화 | 입력

/home/pi/code/python/p_math.py

```
[Wed May 08 2024 12:26:40 GMT+0900 (대한민국 표준시)]:

2
3.0
True
3
4
96
6
100

종료됨.
```

파이썬 기본 코딩 - 문자

파이썬 기본 코딩 – 문자 (예제)

b_text.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_text.py

```
1 print('안녕' + '파이보')
2 text = '잘가'
3 text = str(text) + '파이보'
4 print(text)
5 print('abc'.upper())
6 print('abc'.replace('c', 'V'))
7 print('abc'[::-1])
8 print(len('abc'))
9 print('abc'.count('a'))
```

문자 합성

대문자/소문자 변경
특정 문자 치환
문자 뒤집기
문자 개수 확인하기
특정 문자 개수 확인하기

결과

초기화 | 입력

/home/pi/code/python/p_text.py

[Wed May 08 2024 12:33:43 GMT+0900 (대한민국 표준시)]:

안녕파이보
잘가파이보
ABC
abV
cba
3
1

종료됨.

파이썬 기본 코딩 - 목록

파이썬 기본 코딩 – 목록 (예제1)

b_list1.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_list1.py

```
1 items = ['사과', '배', '바나나']
2 print(items)
3
4 items[0] = '수박'
5 items.insert(0, '복숭아')
6 print(items)
7
8 print(items[-1])
9 print(len(items))
```

목록 생성

사과 -> 수박 교체
복숭아 앞에 추가

마지막 항목 조회
목록 개수 조회

결과

초기화 | 입력

/home/pi/code/python/p_list1.py

```
[Wed May 08 2024 12:41:26 GMT+0900 (대한민국 표준시)]:

['사과', '배', '바나나']
['복숭아', '수박', '배', '바나나']
바나나
4

종료됨.
```

파이썬 기본 코딩 – 목록 (예제2)

b_list2.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_list2.py

```
1 items = ['a', 'b', 'c', 'd', 'e']
2 print(items[1 : 3])
3 print(list(reversed(items)))
4
5 text = '-'.join(items)
6 print(text)
7 print('a,b,c,d,e'.split(','))
```

index1 ~ (3-1)번까지 -> items[1], items[2], items[3] (제외)
반대 방향으로 저장

목록 -> 텍스트 (구분 문자는 '-')

텍스트->목록 ',' 로 구분함

결과

초기화 | 입력

/home/pi/code/python/p_list2.py

```
[Wed May 08 2024 12:50:47 GMT+0900 (대한민국 표준시)]:

['b', 'c']
['e', 'd', 'c', 'b', 'a']
a-b-c-d-e
['a', 'b', 'c', 'd', 'e']

종료됨.
```

파이썬 기본 코딩 - 변수

파이썬 기본 코딩 – 변수 (예제)

b_variable.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_variable.py

```
1 text = '파이보'
2 num = 123
3 items = ['가', '나', '다']
4 dic = {'name': 'robot', 'number':10}
5
6 print(text)
7 print(num)
8 print(items, items[0])
9 print(dic, dic['name'])
```

텍스트(문자)
숫자
리스트
딕셔너리

* 변수 생성
문자/숫자/리스트/딕셔너리 등 다양한 형태의 값을 저장 가능

결과

초기화 | 입력

/home/pi/code/python/p_variable.py

```
[Wed May 08 2024 14:23:13 GMT+0900 (대한민국 표준시)]:

파이보
123
['가', '나', '다'] 가
{'name': 'robot', 'number': 10} robot

종료됨.
```

파이썬 기본 코딩 - 함수

기본 블록 코딩 – 함수 (예제)

b_function.py

블록 | 파이썬 | T↕ 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_function.py

```
1  def func1():
2    print('Hello')
3
4  def func2(name):
5    print('Hello', name)
6
7  def func3():
8    return 'Hi'
9
10 def func4(name):
11   return f'Hi {name}'
12
13 func1()
14 func2('pibo')
15
16 result3 = func3()
17 result4 = func4('pibo')
18 print(result3)
19 print(result4)
20
```

1) 매개변수 X 반환값 X

2) 매개변수 O 반환값 X

3) 매개변수 X 반환값 O

4) 매개변수 O 반환값 O

1) 함수, 2) 함수 호출

3) 함수, 4) 함수 호출
결과 확인

결과

초기화 | 입력

/home/pi/code/python/p_function.py

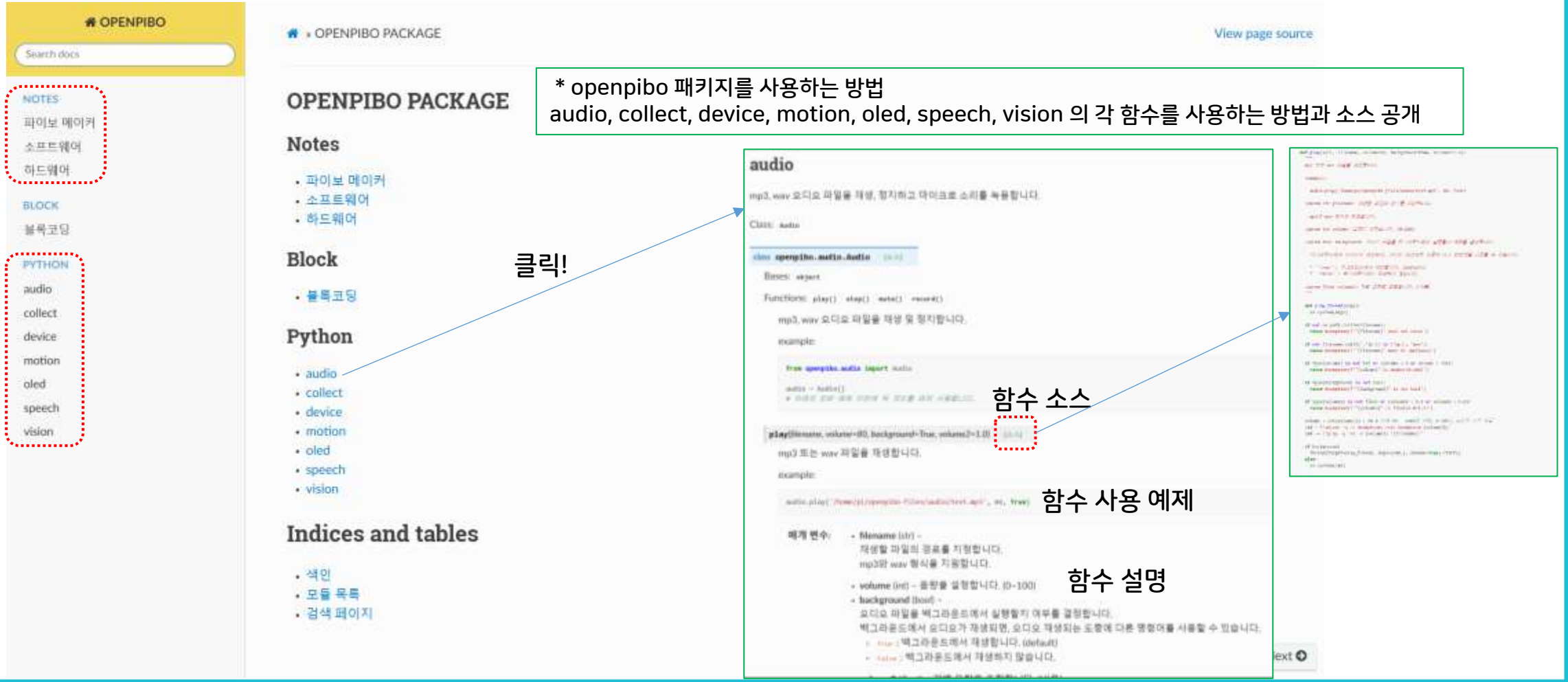
```
[Wed May 08 2024 14:37:22 GMT+0900 (대한민국 표준시)]:

Hello
Hello pibo
Hi
Hi pibo

종료됨.
```

파이보 파이썬 코딩 - 가이드

파이보 파이썬 코딩 – 가이드

가이드> https://themakerrobot.github.io/openpibo-python/build/html/index.html

예제에 포함되지 않는 함수의 가이드도 모두 포함



파이보 파이썬 코딩 - 소리

파이보 파이썬 코딩 – 소리 (예제)

p_audio.json

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_audio.py

```
1  from openpibo.audio import Audio
2  import time
3
4  audio = Audio()
5
6  audio.play('/home/pi/openpibo-files/audio/music/classic.mp3', 80)
7  time.sleep(5)
8  audio.stop()
9
10 audio.record('/home/pi/myaudio/test.wav', 5, False)
11 audio.play('/home/pi/myaudio/test.wav', 80)
```

audio, time 라이브러리 추가

음악 재생 – 파일이름, 음량
5초 후, 음악 정지
* openpibo-files/audio 에 샘플 음악 파일이 있습니다.

5초간 녹음 후, '/home/pi/myaudio/test.wav' 파일 저장
녹음한 파일 재생

파이보 파이썬 코딩 - 수집

파이보 파이썬 코딩 – 수집 (예제)

p_collect.py

블록 | 파이썬 | T↕ 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_collect.py

```
1  from openpibo.collect import Wikipedia
2  from openpibo.collect import Weather
3  from openpibo.collect import News
4
5  wikipedia = Wikipedia()
6  weather = Weather()
7  news = News()
8
9  print('## 위키피디아')
10 result = wikipedia.search('로봇')
11 print(result['0']['content'])
12
13 print('## 날씨')
14 result = weather.search('서울')
15 print(result)
16
17 print('## 뉴스')
18 result = news.search('속보')
19 print(result[0]['title'])
```

collect 라이브러리 추가

결과 분석

* 위키피디아 결과 데이터
(검색된 내용에 따라 데이터의 일부만 있을 수 있음)

```
result = [
 { '0': { 'content': [ 위키피디아 검색 내용1,2, ···] },
 { '1': { 'content': [ 위키피디아 검색 내용1,2, ···] },
...
 { 'n': { 'content': [ 위키피디아 검색 내용1,2, ···] },
]
```

* 날씨 결과 데이터

```
result = {
 'forecast': 전체 예보,
 'today':
  { 'weather': 예보, 'minimum_temp': 최저기온, 'highst_temp': 최고 기온},
 'tomorrow':
  { 'weather': 예보, 'minimum_temp': 최저기온, 'highst_temp': 최고 기온},
'after_tomorrow':
  { 'weather': 예보, 'minimum_temp': 최저기온, 'highst_temp': 최고 기온},
}
```

* 뉴스

```
result = [
 {'title': 제목, 'link': 뉴스링크, 'description': 내용, 'pubDate ' : 날짜},
 {'title': 제목, 'link': 뉴스링크, 'description': 내용, 'pubDate ' : 날짜},
...
 {'title': 제목, 'link': 뉴스링크, 'description': 내용, 'pubDate ' : 날짜},
]
```

파이보 파이썬 코딩 – 수집 (예제)

p_collect.py

결과

초기화 입력

/home/pi/code/python/p_collect.py

[Wed May 08 2024 15:20:34 GMT+0900 (대한민국 표준시)]:

result['0']['content']

위키피디아
['\n', '로봇(문화어: 로보트, 영어: robot)은 다양한 작업을 자동으로 수행하도록 프로그래밍된 기계장치다. 프로그램으로 작동하고 (programmable), 사람이 직접 수행할 수 없는 어렵고 복잡하며 위험한 일련의 작업들(complex series of actions)을 사람 대신 실행하는 기계적 장치다. 자동차 생산 라인 등 제조공장에서 조립, 용접, 핸들링 (handling) 등을 수행하는 자동화된 로봇을 산업용 로봇이라 하고, 환경을 인식해 스스로 판단하는 기능을 가진 로봇을 \'지능형 로봇 \'이라 부른다. 학교 등의 급식실에서 사람 대신 조리 업무를 수행하는 푸드테크 로봇(급식로봇)도 있다. 사람과 닮은 모습을 한 로봇을 \'안드로이드\'라 부른다. 다른 뜻은 형태가 있으며, 자신이 생각할 수 있는 능력을 가진 기계라고도 한다. 인공의 동력을 사용하는 로봇은 사람 대신 또는 사람과 함께 일을 한다. 통상 로봇은 제작자가 계획한 일을 하도록 설계된다. \'로봇\'이란 용어는 체코슬로바키아의 극작가 카렐 차페크(Karel Čapek)가 1920년에 발표한 희곡 "R.U.R"에 쓴 것이 퍼져 일반적으로 사용되게 되었다. 또한 로봇의 어원은 체코어로 "노동", "노예", "힘들고 단조로운 일"을 의미하는 robota이다.\n', '우리가 아는 장난감 로봇은 사실 로봇이 아니라 장난감이다. 수동으로 움직이기 때문이다.\n']

result

날씨
{'forecast': '아침 기온 낮아 쌀쌀, 낮과 밤의 기온차 큼, 오늘 오후 경기북부내륙 비 조금 곳', 'today': {'weather': '가끔 구름많다가 저녁부터 차차 맑아짐, 경기북부내륙 오후(12~18시) 비 조금 곳, 서울과 그 밖의 지역 0.1mm 미만 빗방울 곳,\n 서해5도 구름많다가 낮부터 대체로 맑음', 'minimum_temp': '4.7 ~ 11.5', 'highst_temp': '19 ~ 23'}, 'tomorrow': {'weather': '대체로 맑음\n* 예상 강수량(8일 오후)- 경기북부내륙: 5mm 미만', 'minimum_temp': '7 ~ 12', 'highst_temp': '18 ~ 23'}, 'after_tomorrow': {'weather': '대체로 맑음\n* 예상 강수량(8일 오후)- 경기북부내륙: 5mm 미만', 'minimum_temp': '9 ~ 14', 'highst_temp': '20 ~ 25'}}
뉴스
2024 '최강야구' 두번째 직관! 최강 몬스터즈 VS 강릉영동대학교 맞대결

result[0]['title']

종료됨.

파이보 파이썬 코딩 - 장치

파이보 파이썬 코딩 – 장치 (예제)

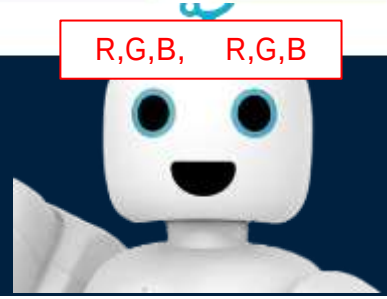
p_device.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_device.py

```
1  from openpibo.device import Device
2  import time
3
4  device = Device()
5
6  device.eye_on(255,255,255)
7  time.sleep(2)
8  device.eye_on(255,0,0,0,0,255)
9  time.sleep(2)
10 device.eye_off()
11
12 print(device.get_dc(True))
13 print(device.get_battery(True))
14 print(device.get_pir())
15 print(device.get_touch())
16 print(device.get_button())
```

device, time 라이브러리 추가



눈 LED 색상 변경
(R, G, B) – 좌우측 동일하게 RGB 설정
(R, G, B, R, G, B) – 좌우측 다르게 RGB 설정

눈 LED를 끕니다.

아답터 체크: on / off

배터리 체크: % 단위 배터리 잔량

사람 체크: 사람있으면 person 없으면 공백

터치 체크: 터치 있으면 touch, 없으면 공백

버튼 체크: 버튼 눌렀으면 on, 없으면 공백

결과

초기화 | 입력

/home/pi/code/python/p_device.py

```
[Wed May 08 2024 15:28:28 GMT+0900 (대한민국 표준시)]:

on
100%



종료됨.
```

파이보 파이썬 코딩 - 동작

파이보 파이썬 코딩 – 동작 (예제)

p_motion.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_motion.py

```
1  from openpibo.motion import Motion
2  import time
3
4  motion = Motion()
5
6  print(motion.get_motion())
7  motion.set_motion('wave1', 1)
8
9  print(motion.get_motion(path='/home/pi/mymotion.json'))
10 motion.set_mymotion('test', 1)
11
12 while True:
13   motion.set_speed(4, 100)
14   motion.set_acceleration(4, 5)
15   motion.set_motor(4, (-40))
16   time.sleep(1)
17   motion.set_speed(4, 100)
18   motion.set_acceleration(4, 40)
19   motion.set_motor(4, 40)
20   time.sleep(1)
```

motion, time 라이브러리 추가

내장 모션 목록 확인하기
내장 모션 선택/반복 횟수 설정/ 실행

내 모션 목록 확인하기
내 모션 선택/반복 횟수 설정/ 실행

개별 모터의 속도/가속도/위치 설정
속도/가속도: 0 ~ 255 범위 (0은 최대)
속도/가속도 블록에서 값을 조정하면, 차이 확인 가능

모션 목록 가져오기

내 모션 목록 가져오기

결과

```
[Wed Mar 20 2024 16:52:44 GMT+0900 (대한민국 표준시)]:

['stop', 'stop_body', 'sleep', 'lookup', 'left', 'left_half', 'right', 'right_half',
'forward1', 'forward2', 'backward1', 'backward2', 'step1', 'step2', 'hifive',
'cheer1', 'cheer2', 'cheer3', 'wave1', 'wave2', 'wave3', 'wave4', 'wave5',
'wave6', 'think1', 'think2', 'think3', 'think4', 'wake_up1', 'wake_up2',
'wake_up3', 'hey1', 'hey2', 'yes_h', 'no_h', 'breath1', 'breath2', 'breath3',
'breath_long', 'head_h', 'spin_h', 'clapping1', 'clapping2', 'handshaking',
'bow', 'greeting', 'hand1', 'hand2', 'hand3', 'hand4', 'foot1', 'foot2', 'foot3',
'speak1', 'speak2', 'speak_n1', 'speak_n2', 'speak_q', 'speak_r1',
'speak_r2', 'speak_l1', 'speak_l2', 'welcome', 'happy1', 'happy2', 'happy3',
'excite1', 'excite2', 'boring1', 'boring2', 'sad1', 'sad2', 'sad3', 'handup_r',
'handup_l', 'look_r', 'look_l', 'dance1', 'dance2', 'dance3', 'dance4', 'dance5']
['test']

종료됨.
```

파이보 파이썬 코딩 - 화면

파이보 파이썬 코딩 – 화면 (예제)

p_display.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_display.py

```
1  from openpibo.oled import Oled
2  import time
3
4  oled = Oled()
5
6  oled.set_font(size=20)
7  oled.draw_text((0, 0), '안녕하세요.')
8  oled.show()
9  time.sleep(1)
10
11 oled.clear()
12 oled.draw_image('/home/pi/openpibo-files/image/weather/cloud.jpg')
13 oled.show()
14 time.sleep(1)
15
16 oled.clear()
17 oled.draw_rectangle((0, 0, 30, 30), False)
18 oled.draw_ellipse((31, 31, 60, 60), True)
19 oled.draw_line((0, 0, 30, 30))
20 oled.show()
21
```

oled, time 라이브러리 추가

문자 크기를 설정할 수 있고, 문자 입력
좌표는 좌측 상단 x,y 좌표

* openpibo-files/image/ 샘플 이미지 파일 사용 가능
파일 명을 직접 작성 or 이미지 변수도 가능

네모/원, 선 표시 / 좌표는 좌측상단, 우측하단 x,y 좌표
네모/원의 경우, 채우기/채우기 없음을 선택 (True, False)

파이보 파이썬 코딩 - 음성

파이보 파이썬 코딩 – 음성 (예제)

p_voice.py

블록 | 파이썬 | T↕ 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_voice.py

```
1  from openpibo.speech import Dialog
2  from openpibo.speech import Speech
3  from openpibo.audio import Audio
4
5  dialog = Dialog()
6  speech = Speech()
7  audio = Audio()
8
9  print('## 내장 대화')
10 print(dialog.get_dialog('반가워'))
11
12 print('## 나만의 대화')
13 dialog.load('/home/pi/code/mychat.csv')
14 print(dialog.get_dialog('안녕'))
15 print(dialog.get_dialog('반가워'))
16
17 speech.tts(string='안녕하세요', filename='/home/pi/code/voice.mp3', voice='main')
18 audio.play('/home/pi/code/voice.mp3', 80)
```

speech, audio 라이브러리 추가

내장 대화에서 대답을 선택합니다.

내장 대화를 사용하지 않고,
mychat.csv 에서 대답을 선택합니다.

음성 합성하여, 지정한 파일에 저장합니다.

espeak는 오프라인에서 사용이 가능하고,
그 외의 목소리는 인터넷 연결 환경에서 사용이 가능합니다.

결과

초기화 | 입력

/home/pi/code/python/p_voice.py

```
[Wed May 08 2024 16:22:13 GMT+0900 (대한민국 표준시)]:

## 내장 대화
파이보도 반가워요.
## 나만의 대화
나만의 대답1
나만의 대답2

종료됨.
```

/home/pi/code/mychat.csv

```
1 안녕하세요,나만의 대답1
2 반가워요,나만의 대답2
3 오늘 날씨는 어때?,나만의 대답3
```

파이보 파이썬 코딩 - 시각

파이보 파이썬 코딩 – 시각 (예제1)

p_vision1.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_vision1.py

```
1  from openpibo.vision import Camera
2  from openpibo.vision import Detect
3
4  camera = Camera()
5  detect = Detect()
6
7  image = camera.read()
8  camera.imwrite('/home/pi/code/image.jpg', image)
9  camera.imshow_to_ide(image)
10
11 cartoon = camera.stylization(image)
12 camera.imwrite('/home/pi/code/cartoon.jpg', cartoon)
13
14 items = detect.classify_image(image)
15 print([ item['name'] for item in items])
16
17 items = detect.detect_object(image)
18 print([ item['name'] for item in items])
19
20 print(detect.detect_qr(image)['data'])
```

vision 라이브러리 추가

이미지를 촬영하고, 파일로 저장합니다.

이미지 파일을 뷰어에 표시합니다.

이미지를 변환합니다.

결과

```
[Wed Apr 03 2024 15:01:47 GMT+0900 (대한민국 표준시)]:
['desktop computer', 'printer', 'iPod', 'parking meter', 'cellular telephone, cellular phone, cellphone, cell, mobile phone']
['cell phone', 'laptop']
http://en.m.wikipedia.org
종료됨.
```

* classify_image 결과 값 = [
  {'score': 확률, 'name': 이름},
  {'score': 확률, 'name': 이름},
  ...,
  {'score': 확률, 'name': 이름}
]

* detect_object 결과 값 = [
  {'score': 확률, 'name': 이름, 'position': 좌측상단/우측하단 좌표},
  {'score': 확률, 'name': 이름 , 'position': 좌측상단/우측하단 좌표},
  ...,
  {'score': 확률, 'name': 이름 , 'position': 좌측상단/우측하단 좌표}
]

분류하기 – 리스트 반환(상위 5개 항목) / 이미지를 1000개 항목으로 분류합니다. – imagenet1k)
사물 찾기 – 리스트 반환(인식한 사물 이름) / 학습된 80가지 사물을 인식합니다.
QR코드 찾기 – 문자열 반환(QR코드 내용)

파이보 파이썬 코딩 – 시각 (예제2)

p_vision2.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_vision2.py

```python
from openpibo.vision import Camera
from openpibo.vision import Detect

camera = Camera()
detect = Detect()

img = camera.read()
result = detect.detect_pose(img)

print([[_i.coordinate.x, _i.coordinate.y] for _i in result['data'][0][0]])
print(detect.analyze_pose(result))
```

vision 라이브러리 추가

포즈 데이터 추출

포즈좌표: 인체의 17개 포인트 좌표 (아래 참고)

모션인식: 포즈좌표를 분석하여, left_hand_up, right_hand_up, clap 인식

#주의!
포즈인식은 사람이 없어도 잘못된 값이 나오는 식으로 구현되어 주의가 필요합니다.
사람의 전신이 모두 나오면, 잘 인식하고, 적어도 상반신은 카메라에 담겨있어야 유의미한 결과를 가져올 수 있습니다.

좌표 인덱스
NOSE, LEFT_EYE, RIGHT_EYE, LEFT_EAR, RIGHT_EAR = 0,1,2,3,4
LEFT_SHOULDER, RIGHT_SHOULDER, LEFT_ELBOW, RIGHT_ELBOW, LEFT_WRIST, RIGHT_WRIST = 5,6,7,8,9,10
LEFT_HIP, RIGHT_HIP, LEFT_KNEE, RIGHT_KNEE, LEFT_ANKLE, RIGHT_ANKLE = 11,12,13,14,15,16

결과

파이보 파이썬 코딩 – 시각 (예제3)

p_vision3.py

블록 | 파이썬 | 14 | 실행 | 정지 | 저장

/home/pi/code/python/p_vision3.py

```
1  from openpibo.vision import TeachableMachine
2  from openpibo.vision import Camera
3
4  tm = TeachableMachine()
5  camera = Camera()
6
7  tm.load('/home/pi/mymodel/model_unquant.tflite', '/home/pi/mymodel/labels.txt')
8  img = camera.read()
9
10 result = tm.predict(img)
11 print(result[0])
```

vision 라이브러리 추가

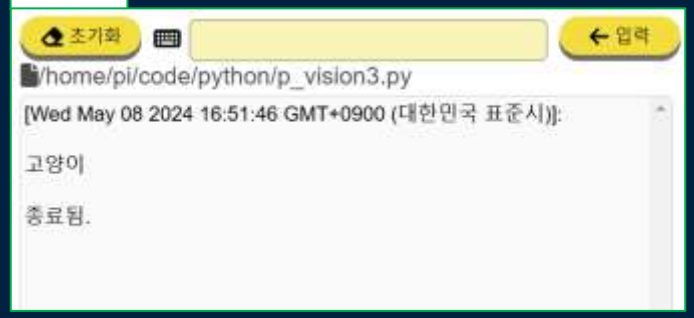
1. 티처블머신의 이미지 프로젝트에서 모델을 내보낼 때,

2. Tensorflow Lite > 부동소수점 or 양자화됨을 선택합니다.

3. 압축을 풀어서 모델과 라벨파일을 업로드 하고, 블록에서 설정합니다.
   > Tensorflow Lite (모델: *.tflite, 라벨: labels.txt)

* tm.predict 함수 결과 분석
- 가장 확률이 높은 항목 이름, 원천 데이터 (각 Class의 확률 값 리스트)

결과



파이보 파이썬 코딩 – 예제

예제 파일 사용

- 아래 예제와 기타 필요한 파일입니다.
- 압축을 풀고, 파이보 메이커의 IDE에 업로드 하시고 사용하세요.
- 아래 사이트에서도 예제 파일 및 코드 설명을 확인할 수 있습니다.

https://themakerrobot.github.io/openpibo-python/build/html/index.html

python-coding.240508.zip

감사합니다!