デュアルウェア講習会 II

Pythonからデータベース操作

目標：Python プログラムからデータベースを操作

目次

第１章

課題

温湿度センサの値をデータベースへ登録します（図 1.1）.

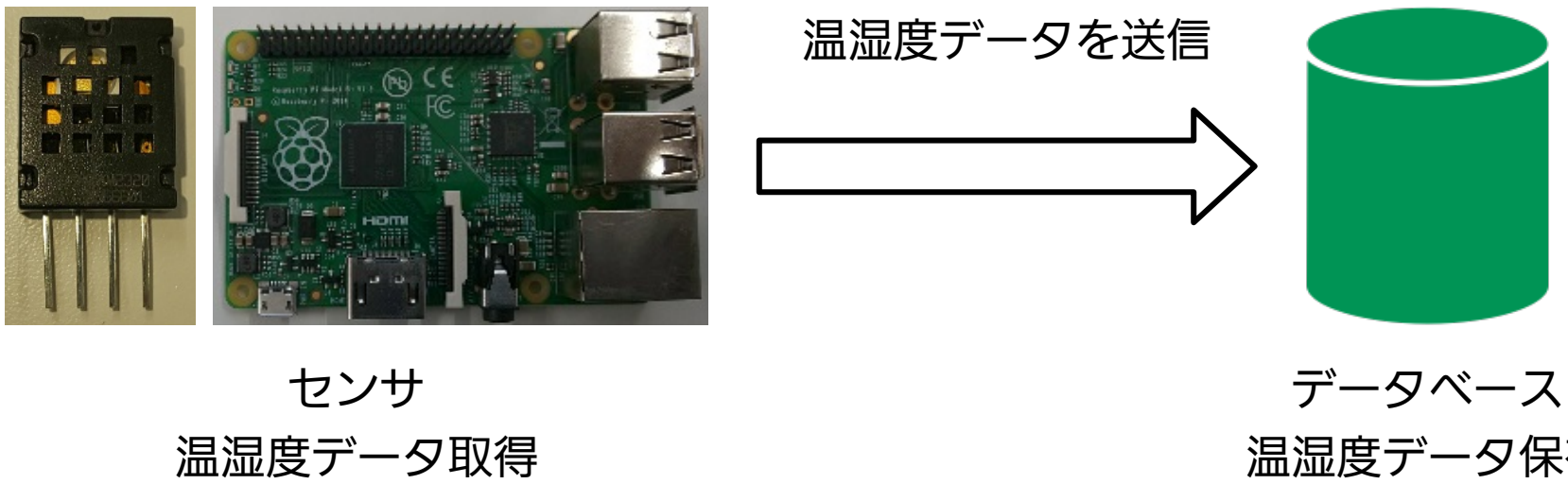

図 1.1: 課題システム

第 2 章

インストール

2.1 PostgreSQL インストール（Windows 版）

`https://www.enterprisedb.com/downloads/postgres-postgresql-downloads` にアクセスして，インストーラをダウンロードします（図 2.1）.

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
12.1	N/A	N/A	Download	Download	N/A
11.6	N/A	N/A	Download	Download	N/A
10.11	Download	Download	Download	Download	Download
9.6.16	Download	Download	Download	Download	Download
9.5.20	Download	Download	Download	Download	Download
9.4.25	Download	Download	Download	Download	Download
9.3.25 (Not Supported)	Download	Download	Download	Download	Download

図 2.1: PostgreSQL ダウンロード

本講習でダウンロードするバージョンは `9.6.16` になります．ダウンロード後，インストーラを起動します．セットアップ画面が立ち上がりますので（図 2.2），指示にしたがってインストールをします．

図 2.2: PostgreSQL セットアップ画面

以下の設定を行います.

- インストールディレクトリ：デフォルト（C:¥Program Files¥PostgreSQL¥9.6）
- データディレクトリ：デフォルト（C:¥Program Files¥PostgreSQL¥9.6¥data）
- パスワード：任意（この講習会ではコンピュータ名）
- ポート番号：デフォルト（5432）
- AdvanceOption:デフォルト（Default locale）

インストールが完了すると，[スタックビルダ] が立ち上がりますが，これは [キャンセル] します. そして，インストール完了後，環境変数に PostgreSQL のパスを追加します。[コントロールパネル]→[システムとセキュリティ]→[システム]→[システムの詳細設定] と進み，システムのプロパティを開き，[詳細設定] タブの [環境変数] をクリックしてください（図 2.3）.

図 2.3: システムプロパティ

[環境変数] のウィンドウが開きましたら，システム環境変数の新規ボタンを押してください（図 2.4）.

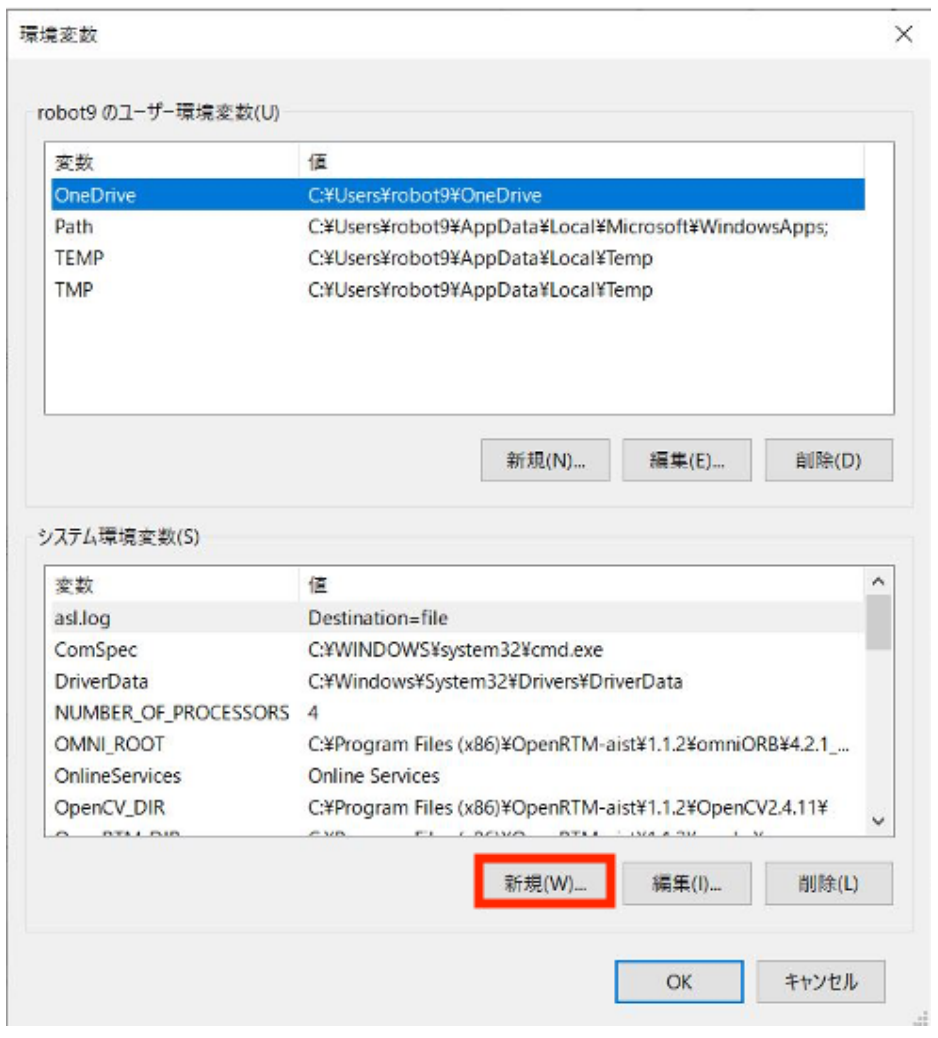

図 2.4: 環境変数編集 1

[新しいシステム変数] のウィンドウに表 2.1 の内容を記入し，OK ボタンを押します（図 2.5）.

表 2.1: 追加編集名

変数名	POSTGRES_HOME
変数型	C:¥Program Files¥PostgreSQL¥9.6

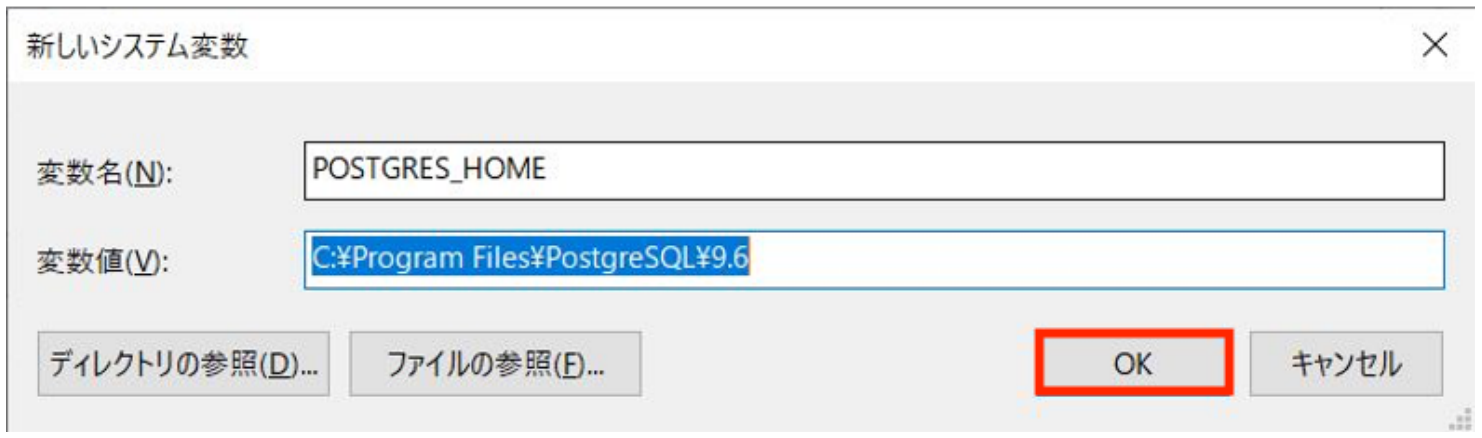

図 2.5: 環境変数編集 2

[環境変数] のウィンドウに戻りましたら，システム環境変数内の path を選択した状態で編集ボタンを押します．[環境変数名の編集] のウィンドウが出ましたら，新規のボタンを押して %POSTGRES_HOME%¥bin¥ を追加してください（図 2.6）.

追加をしましたら，OK ボタンを押して [環境変数名の編集] と [環境変数] のウィンドウを閉じます.

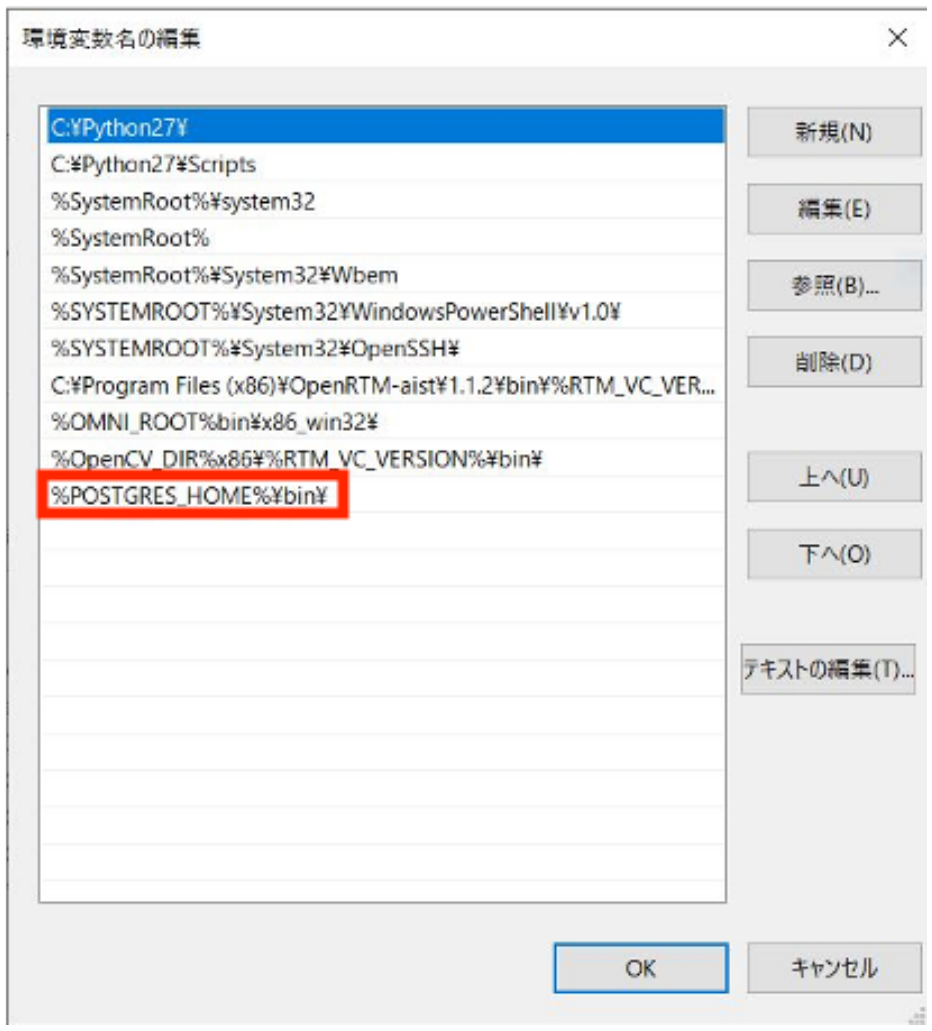

図 2.6: 環境変数編集 3

2.2 pyscopg2 インストール

2.2.1 Windows 版インストール

Windows に `pyscopg2` をインストールするには `pip` を使用します.

- 最初にコマンドプロンプトを起動してください．Windows キーと R キーを同時に押し，[ファイル名を指定して実行] ダイアログを開きます.
- ダイアログ内に `cmd` と記入して，OK ボタンを押してください．コマンドプロンプトが開きます.

最初に `pip` のバージョンを最新にします．以下のコマンドを実行して，最新にしてください.

```
1 $ python -m pip install --upgrade pip
```

最新へのバージョンアップが成功すれば, 最後に `Successfully installed pip-19.3.1`[*1] と表示されます.

最新へのバージョンアップ後，以下のコマンドでインストールをします.

```
1 $ pip install psycopg2
```

インストールが成功すると，最後に `Successfully installed psycopg2-2.8.4`[*2] と表示されます.

[*1] `19.3.1` はバージョンなので，`19.3.1` 以外が表示される場合もあります.
[*2] `2.8.4` はバージョンなので，`2.8.4` 以外が表示される場合もあります.

インストールされていることを確認します．コマンドプロンプト上で python と打ち込み，Pyhton の対話モードを立ち上げます．対話モード上で import psycopg2 を実行し，エラーが表示されないことを確認します．

```
$ python
>>> import psycopg2
```

2.2.2 Raspberry Pi 版インストール

Raspberry Pi 上で，以下のコマンドを実行しインストールします．

```
$ sudo apt-get install python-psycopg2
```

インストールされていることを確認します．python と打ち込み，Pyhton の対話モードを立ち上げます．対話モード上で import psycopg2 を実行し，エラーが表示されないことを確認します．

```
$ python
>>> import psycopg2
```

第 3 章

Python データベース操作

psycopg2 を使用して，Python に接続します.

3.1　Python プログラムからデータベースに接続

Python プログラムからデータベースへ接続を行います．最初に以下のコマンドで，自分が接続している Raspberry Pi の IP アドレスを確認してください．確認したら，後で使用しますのでメモをしてください.

```
1 $ ifconfig wlan0
```

メモをしたら，Raspberry Pi で Python の対話モードを起動してください．起動後，以下のプログラムを実行し，psycopg2 を読み込みます.

```
1 >>> import psycopg2
```

データベースへのアクセスは，以下の connect メソッドを使用します（表 3.1）.

表 3.1: connect メソッド

メソッド名	connect(parameters)	
戻り値	接続オブジェクトを返す	
引数	引数名	説明
	parameters	接続するための必要情報を空白文字で連結した文字列 必要な情報は以下の通り host=接続するコンピュータの IP アドレス dbname=接続するデータベース名 user=接続するデータベースのユーザ名 password=接続するデータベースのパスワード port=接続するデータベースのポート番号

以下のように使用します.

```
1 >>> connection = psycopg2.connect("host=192.168.11.122 dbname=pi
    user=pi password=raspberry port=5432")
```

IP アドレスは，先ほどメモした IP アドレスに変更してください．次のように各値を変数に格納後，文字列として連結するやり方もできます．この場合，各値を変数にすることができるので，値の変更が容易になります．

```
1 >>> host = "192.168.11.122"
2 >>> dbname = "pi"
3 >>> user = "pi"
4 >>> pw = "raspberry"
5 >>> port = "5432"
6 >>> connect = "host=" + host + " dbname=" + dbname + " user=" +
     user + " password=" + pw + "  port=" + port
7 >>> connection = psycopg2.connect(connect)
```

3.2　カーソルオブジェクトの取得

カーソルオブジェクトを取得します．以下のプログラムを実行することにより，カーソルオブジェクトの取得ができ，SQL を実行できるようなります．

```
1 >>> cur = connection.cursor()
```

3.3　Python プログラムから SQL を実行

psycopg2 では SQL を実行するには，excecute メソッドを使用します（表 3.2）．引数で渡す SQL 文を実行した結果を取得します．

表 3.2: excecute メソッド

メソッド名	excecute(sql)	
戻り値	なし	
引数	引数名	説明
	sql	実行したい SQL 文（文字列）

SQL 文を実行するには以下のように記述します．

```
1 >>> cur.execute("SELECT * FROM reidai;")
```

SQL 文は引数として直接入力が可能ですが，変数に文字列として格納して使用することもできます．

```
1 >> sql = "SELECT * FROM reidai;"
2 >> cur.execute(sql)
```

3.4　SQL の実行結果を出力

excecute メソッドを実行したら，fetchall メソッドを使用して，実行結果を出力します．

表 3.3: fetchall メソッド

メソッド	fetchall()
戻り値	SQL の実行結果の全レコードを 2 次元リストとして出力

以下のように使用します．

```
1 >>> cur.fetchall()
```

実行結果の全レコードが 2 次元リストで表示されます．

3.5　終了

SQL 実行後，終了するには以下のようにします．

```
1 >>> cur.close()
2 >>> connection.close()
```

cur.close() でカーソルをクローズし，connect.close() で接続をクローズしています．

3.6　SELECT 文サンプルプログラム

以下のプログラムをコピーした後，03_sample1.py のファイル名で保存してください．その際，#IP アドレス#は自分が接続している Raspberry Pi の IP アドレスに変更します．保存後，Raspberry Pi にファイルを転送して実行してください．

```
1  # -*- coding: utf-8 -*-
2  import psycopg2
3
4  # データベース接続パラメータ
5  host = #IPアドレス#
6  dbname = 'pi'
7  user = 'pi'
8  pw = "raspberry"
9  port = "5432"
10 connect = "host=" + host + " dbname=" + dbname + " user=" + user
       + " password=" + pw + " port=" + port
11 connection = psycopg2.connect(connect) # データベースへアクセス
12 cur = connection.cursor() # カーソルを取得
13 cur.execute("SELECT * FROM reidai ;") # SQL文を実行
```

```
14 row = cur.fetchall() # 実行結果をrowに格納
15 print(row)
16 cur.close() # カーソルをクローズ
17 connection.close() # 接続をクローズ
```

実行結果として，全レコードが 2 次元リストで表示されることを確認して下さい．

3.7 INSERT 文

INSERT 文を実行するときは基本的に SELECT 文と同じです．reidai テーブルへの INSERT を Python で行うと以下のようになります．

```
1 >>> cur.execute("INSERT INTO reidai (id,date,time,temp,place)
    VALUES(20190101160000,'2019-01-01','16:00:00',20.0,'LICTiA
    ');")
```

しかし，固定値の場合は問題ありませんが，登録したいデータを適宜変更する必要がある場合は，以下のように変数を用いて登録処理を実行します．

```
1 >>> id = 20190101160100
2 >>> date = '2019-01-01'
3 >>> time = '16:01:00'
4 >>> temp = 20.5
5 >>> place = 'LICTiA'
6 >>> cur.execute("INSERT INTO reidai (id,date,time,temp,place)
    VALUES(%s,%s,%s,%s,%s);", (id,date,time,temp,place))
```

%s を使用することで，変数の代入ができます．

さらに，INSERT 文を実行した後，その処理を確定させるために，以下の処理を行います．

```
1 >>> cur.commit()
```

commit をすることでテーブルへの反映が完了します．commit する前にプログラムを終了すると，INSERT したレコードが反映されなくなるので注意してください．

3.7.1 INSERT サンプルプログラム

以下のプログラムをコピーして，03_sample2.py のファイル名で保存してください．保存後，Raspberry Pi 上にファイルを転送して実行してください．#IP アドレス#は，自分の Raspberry Pi の IP アドレスに変更してください．

```
1 # -*- coding: utf-8 -*-
2 import psycopg2
3
4 # データベース接続パラメータ
5 host = #IPアドレス#
```

```
dbname = 'pi'
user = 'pi'
pw = "raspberry"
port = "5432"
conect = "host=" + host + " dbname=" + dbname + " user=" + user +
    " password=" + pw + " port=" + port
connection = psycopg2.connect(conect) # データベース接続
cur = connection.cursor()

# INSERTする値設定
id = 20190101160100
date = '2019-01-01'
time = '16:01:00'
temp = 20.5
place = 'LICTiA'

# SQL文実行
cur.execute("INSERT INTO reidai (id,date,time,temp,place) VALUES
   (%s,%s,%s,%s,%s);",(id,date,time,temp,place))
connection.commit() # コミット

cur.execute("SELECT * FROM reidai;") # SQL文実行
row = cur.fetchall() # SQL文実行結果出力
print(row)

cur.close()
connection.close()
```

実行結果を確認して，INSERT したレコードが追加されていることを確認してください.

第 4 章

課題 1

温度の値を取得する Python プログラムを作成しました．そのプログラムに SQL の INSERT 文を使用して，データベースに温度の値を追加するプログラムを作成します．以下のプログラムの TODO 部分を編集して，プログラムを完成させてください．

```
1  # -*- coding: utf-8 -*-
2  import smbus
3  import time
4  import psycopg2
5  from datetime import datetime
6
7  AM2320_ADDR = 0x5c
8  i2c = smbus.SMBus(1)
9
10 try: # センサを待機状態から起こす
11  i2c.read_byte(AM2320_ADDR)
12 except:
13  pass
14
15 time.sleep(0.003)
16 read_function = 0x03
17 start_reg = 0x02
18 num_reg = 0x02
19
20 # 読み取りたいデータのスタートレジスタとレジスタ長を指定
21 i2c.write_i2c_block_data(AM2320_ADDR, read_function, [start_reg,
      num_reg])
22 time.sleep(0.015)
23
24 # センサから帰ってきた値を読み取る
25 read_num = 4
26 data = i2c.read_i2c_block_data(AM2320_ADDR, read_function,
      read_num)
```

```
temp = float(data[2] << 8 | data[3]) / 10
print "temp:",temp," ℃"

# データベース接続
connection = psycopg2.connect(#TODO:接続のパラメータ追記#)
cur = connection.cursor()

# データベースへの登録
id = int(datetime.now().strftime("%Y%m%d%H%M%S")) # 登録する
   id作成
date = datetime.now().strftime("%Y%m%d") # 登録するdate作成
time = datetime.now().strftime("%H:%M:%S") # 登録するtime作成
place = "LICTiA"
cur.execute(#TODO:INSERT文作成#)

connection.commit()
cur.close()
connection.close()
```

テキストにコピーして，03_kadai1.py のファイル名で保存してください.

第 5 章

課題 2

温度と湿度の値を取得する Python プログラムを作成しました．そのプログラムに SQL の INSERT 文を使用して，データベースに温度センサの値を追加します．以下のプログラムの TODO 部分を編集して，プログラムを完成させてください．

```
1  # -*- coding: utf-8 -*-
2  import smbus
3  import time
4  import psycopg2
5  from datetime import datetime
6
7  AM2320_ADDR = 0x5c
8  i2c = smbus.SMBus(1)
9
10 try: # センサを待機状態から起こす
11  i2c.read_byte(AM2320_ADDR)
12 except:
13  pass
14
15 time.sleep(0.003)
16 read_function = 0x03
17 start_reg = 0x00
18 num_reg = 0x04
19
20 # 読み取りたいデータのスタートレジスタとレジスタ長を指定
21 i2c.write_i2c_block_data(AM2320_ADDR, read_function, [start_reg,
      num_reg])
22 time.sleep(0.015)
23
24 # センサから帰ってきた値を読み取る
25 read_num = 6
26 data = i2c.read_i2c_block_data(AM2320_ADDR, read_function,
      read_num)
```

```
27 humi = float(data[2] << 8 | data[3])/10
28 temp = float(data[4] << 8 | data[5])/10
29 print "temp:",temp," ℃","humi:",humi," %"
30
31 # データベース接続
32 connection = psycopg2.connect(#TODO:接続のパラメータ追記#)
33 cur = connection.cursor()
34
35 # データベースへの登録
36 id = int(datetime.now().strftime("%Y%m%d%H%M%S"))
37 date = datetime.now().strftime("%Y%m%d")
38 time = datetime.now().strftime("%H:%M:%S")
39 place = "LICTiA"
40 cur.execute(#TODO:SQL文作成#)
41
42 connection.commit()
43 cur.close()
44 connection.close()
```

テキストにコピーして，03_kadai2.py のファイル名で保存してください.