

APIの利用

23j1-217

教科書P80-81

この時間の目標

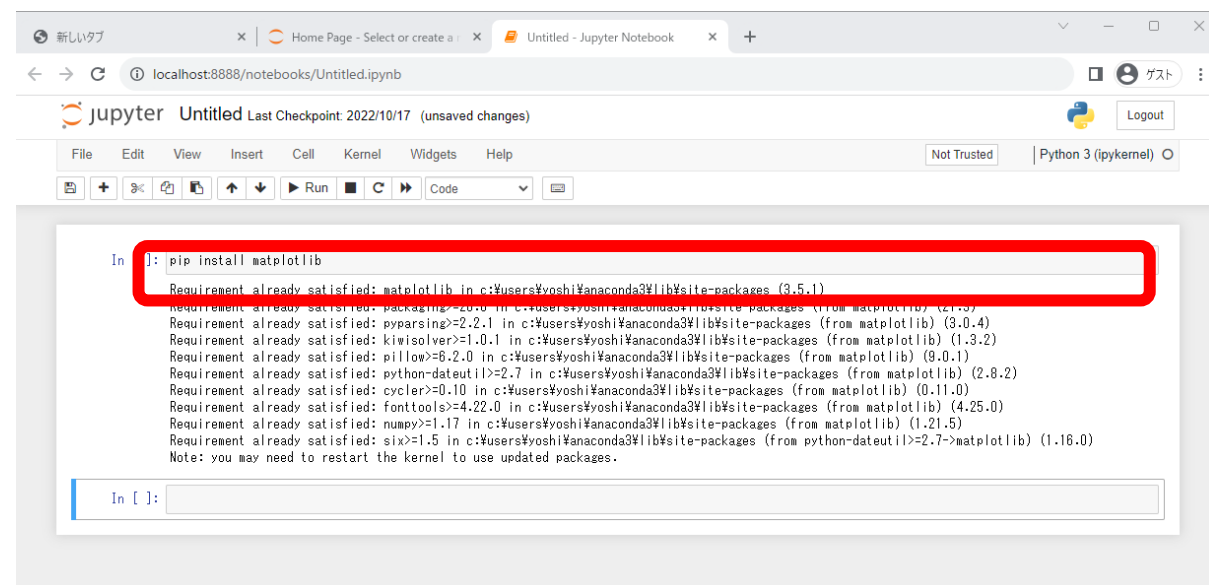
- APIの利用
- S プログラム作成の工夫がよく理解でき、手順を工夫して活用しようと思った
- A プログラム作成の工夫がよく理解できた
- B プログラム作成の工夫が理解できた
- C プログラム作成の工夫が理解できなかった

Jupyter Notebookにおまじない

1. Jupyter Notebookのセルに以下の文字列を入力

```
pip install requests  
pip install folium
```

2. [Shift]+[Enter]



The screenshot shows a Jupyter Notebook interface in a browser. The address bar indicates the URL is localhost:8888/notebooks/Untitled.ipynb. The notebook title is 'Untitled' and it shows 'Last Checkpoint: 2022/10/17 (unsaved changes)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. A code cell is active, containing the command `pip install matplotlib`. The output of the cell shows that the requirement is already satisfied for matplotlib (3.5.1) and lists other dependencies that are also satisfied, such as packaging, pyparsing, kiwisolver, pillow, python-dateutil, cython, fonttools, numpy, and six. A note at the bottom of the output suggests restarting the kernel to use updated packages.

APIの利用

APIの利用

- API【Application Programming Interface】
 - 外部のコンピュータのデータなどを他のプログラムから呼び出して利用する方法
 - 手順や呼び出し方、回答の形式が決まっている
- 自分のコンピュータだけではできないことが実行できる

APIのしくみ

- プログラム
 - パラメータ(変数)を準備
 - 手順に従い送信
- 回答を受信
 - プログラムで必要な部分を取り出して表示

• サーバ

- 変数に対する回答を準備
- 手順に従い送信

インターネット

- インターネットのデータを関数のように使う方法

今日の進め方

1. サンプルを入手する
2. 仕組みを観察
3. 改造する
4. 連結する

01町名検索

1. 情報 I 17回目[01町名検索]をクリック
2. コピーしてJupyterNotebookに貼り付け
3. 実行
4. 自宅の郵便番号で表示するには
 - どこを変えたらいいのか探す
5. 郵便番号を入力できるようにする
 - `input()`命令を追加する

動作確認用
実在する郵便番号
101-0021
183-0033
402-0015
201-0001
521-0003
825-0015

01町名検索-プログラムの説明

- 問い合わせ
 - `res=requests.get(url,params=param)`
- データの取り出し
 - `response=json.loads(res.text)`

```
import requests
import json
url='https://zipcloud.ibsnet.co.jp/api/search'
param={'zipcode':'190-0022'}
res=requests.get(url,params=param)
response=json.loads(res.text)
address=response['results'][0]
print(address['address1']+address['address2']+address['address3'])
```

```
#requests 使います
#json 使います
#urlに問い合わせ先URLを代入
#paramに問い合わせキーワードを代入
#resに問い合わせ結果を代入
#responseにresから文字を取り出して代入
#address=responseの結果のresults[0]を代入
#addressの1・2・3をつなげて表示
```

01町名検索-受け取ったデータ

- `print(address)` を最後追加して実行

東京都立川市錦町

```
{ 'address1': '東京都', 'address2': '立川市', 'address3': '錦町',  
'kana1': 'トウキョウト', 'kana2': 'チカワシ', 'kana3': 'ニシキチョウ',  
'prefcode': '13', 'zipcode': '1900022' }
```

- デクシヨナリというデータ形式
 - {見出し:値,見出し:値,見出し:値,・・・} の形
 - デクシヨナリ名[見出し]で呼び出し
 - `address['address1']` で '東京都' が呼び出せる

02最寄り駅検索

1. 情報 I 17回目[02最寄り駅検索]をクリック
2. コピーしてJupyterNotebookに貼り付け
3. 実行
4. 自宅の郵便番号で表示するには
 - どこを変えたらいいのか探す

動作確認用
実在する郵便番号
101-0021
183-0033
402-0015
201-0001
521-0003
825-0015

02最寄り駅検索-プログラムの説明

- 問い合わせ
 - `res=requests.get(url,params=param)`
- データの取り出し
 - `response=json.loads(res.text)`

```
import requests
import json
url = "http://geoapi.heartrails.com/api/json?method=getStations"
param = {"postal":'1900022'}
res =requests.get(url,params=param)
response = json.loads(res.text)
mystation = response["response"]["station"][0]
print(mystation['line'],mystation['name'])
```

02最寄り駅検索-受け取ったデータ

- `print(mystation)` を最後追加して実行

JR南武線 西国立

```
{'name': '西国立', 'kana': 'にしくにたち', 'line': 'JR南武線', 'y': 35.69375, 'x': 139.423887, 'postal': '1900022', 'prev': '矢川', 'next': '立川', 'prefecture': '東京都', 'distance': 609.3307973941855}
```

- 緯度・経度の情報が含まれている
 - どうやったら取り出せる？

03地図表示

1. 情報 I 17回目[03地図表示]をクリック
2. コピーしてJupyterNotebookに貼り付け
3. 実行

03 地図表示-プログラムの説明

- 緯度・経度を渡すと地図が表示される

```
import folium
gri = [35.69375, 139.423887]
mymap = folium.Map(location=gri, tiles='OpenStreetMap', zoom_start=15)
mymap
```

04天気表示

1. 情報 I 17回目[04天気表示]をクリック
2. コピーしてJupyterNotebookに貼り付け
3. 実行
4. 東京以外の天気を表示するには
 1. どこを変えたらいいのか探す

04 天気表示-プログラムの説明

- 問い合わせ
 - `res=requests.get(url,params=param)`
- データの取り出し
 - `response=json.loads(res.text)`

```
import requests
import json
url='https://www.jma.go.jp/bosai/forecast/data/overview_forecast/130000.json'
res = requests.get(url)
response = json.loads(res.text)
print(response ['text'])
```

プログラムをつなぎ合わせる

複数のAPIを活用する

つなげあわせる1

- 01と02をつなげる
 - 一度郵便番号を入力すると町名と最寄り駅が表示される
- 01のzipcodeを02に引き渡す

動作確認用
実在する郵便番号
101-0021
183-0033
402-0015
201-0001
521-0003
825-0015

01町名検索

パラメータ:郵便番号
(ハイフン有無可)

返値:

address1
address2
address3
kana1
kana2
kana3
prefcode
zipcode

02最寄り駅検索

パラメータ:郵便番号
(ハイフンなし)

返値:

name
kana
line
y(緯度)
x(経度)
postal
prev
next
prefecture
distance

つなげあわせる2

- 01と02に03をつなげる
 - 郵便番号を入力すると
町名・最寄り駅
周辺地図 を表示
- 02の緯度経度を03に
引き渡す

動作確認用
実在する郵便番号
101-0021
183-0033
402-0015
201-0001
521-0003
825-0015

02最寄り駅検索

パラメータ:郵便番号
(ハイフンなし)

返値:

name
kana
line
y(緯度)
x(経度)
postal
prev
next
prefecture
distance

03地図表示

パラメータ:緯度経度
返値:
地図表示

つなげあわせる3

- 01と04をつなげる
 - 郵便番号を入力すると
町名・最寄り駅
天気概況
周辺地図を表示
- 02のprefcodeを加工して
04に引き渡す

動作確認用
実在する郵便番号
101-0021
183-0033
402-0015
201-0001
521-0003
825-0015

01町名検索

パラメータ:郵便番号
(ハイフン有無可)

返値:

address1
address2
address3
kana1
kana2
kana3
prefcode
zipcode

04天気表示

パラメータ:エリアコード
返値:

publishingOffice
reportDatetime
targetArea
headlineText
text