aruba
a Hewlett Packard
Enterprise company

# Airheadsアカデミー

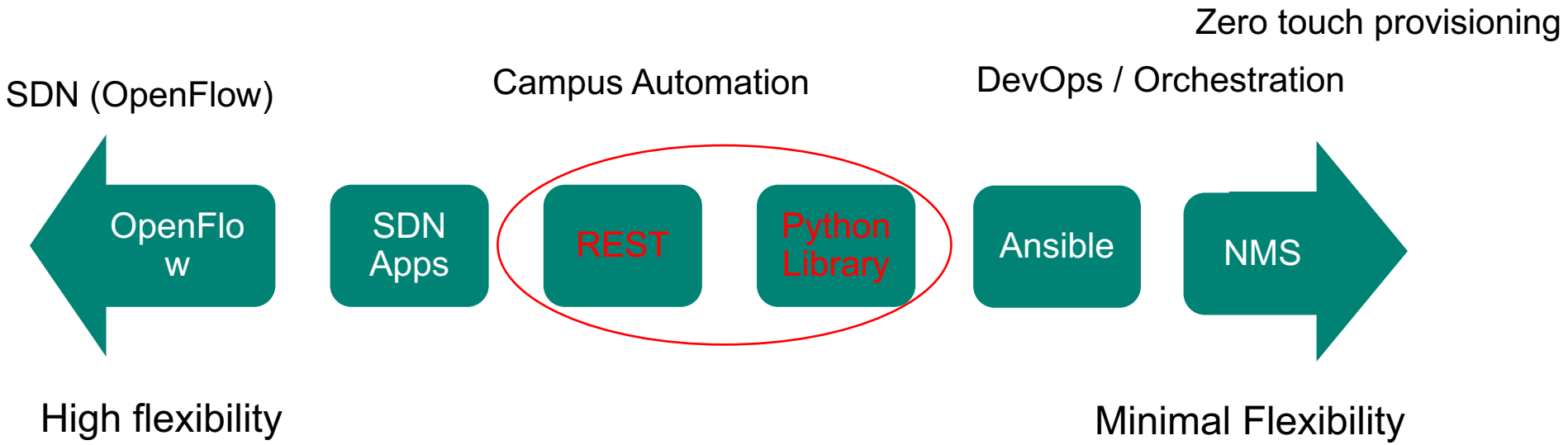
# エンジニアのための API・プログラミング <3>

– 日本ヒューレット・パッカード株式会社

– Aruba事業統括本部

– 2018年8月

# Network automation の振り返り




- 今までの自動化は
  + SDN(OpenFlow) -> 自由度が高すぎ？、導入が難しい
  + NMS, Ansible, NETCONF, OpFlex, etc -> 適用領域が狭い

- 残っている自動化は
  + 導入コストが容易・低コスト・柔軟性高い (Programming) -> REST, Python

# 何故、Network ではプログラミングが拡がらなかったのか？

- サーバーでは Linux, Windows 等の汎用OSが動作し、それ自体がプログラミング環境
  (Compiler, Interpreter, Scripting)
- サーバーとストレージの担当は兼ねる場合もあるが、ネットワークは別な担当
  (ネットワーク担当者がプログラミングに触れる機会が少ない)
- ネットワーク機器はアプライアンス → プログラムを自由に動作させる環境ではない
  (小規模なスクリプトを動作させる例はあるが、汎用的ではなかった: NETCONF, OpFlex, etc)
- ネットワーク機器間では、プロトコルが発展
  ネットワークの新機能は標準化で定められ、各ベンダーが実装
  (サーバーでは、機器間、プロセス間でAPIを使用)
- ネットワーク機器では CLI/GUI を使いこなすエンジニアが多い
  (CLI/GUI を使用した方が早い？　CLI/GUIは自動化とは相性が良くない)

# Network 分野では API/Programming にどう取り組むか？

- API、Programming を導入しやすい環境が整ってきている
  → Python, REST API （習得しやすくなっている）

- 他のソリューションと組み合わせて差別化する必要がある
  → NIとしての付加価値、新たな収益

- Nework エンジニアが API/Programming にどう取り組むか？

  ① REST API 等の一般的な知識は習得しておく
  REST API を使用するSIソリューション構築を支援、スキルの差別化
  (REST API を通してやり取りするネットワークのコンテンツは、Network Engineer が専門家)

  ② REST API /Python を扱えるようになり、運用、構築等で使用できるツールを
  自分で作れるようになる　(業務改善、効率化。新しいスキルセットの習得)

  ③ 新しいネットワークソリューションを開発する
  (アプリケーションの開発：例えば Amazon Echo の Skill 開発、Python で REST APIを使用)

  ④ 今まで通り, 何もしない

aruba
a Hewlett Packard Enterprise company

# 今日の話の対象は？

① REST API 等の一般的な知識は習得しておく
REST API を使用するSIソリューション構築を支援
(REST API を通してやり取りするネットワークのコンテンツは、Network Engineer が専門家)

② REST API /Python を扱えるようになり、運用等で使用できるツールを
自分で作れるようになる　(業務改善、効率化。新しいスキルセットの習得)

③ 新しいネットワークソリューションを開発する
(アプリケーションの開発：例えば Amazon Echo の Skill 開発、Python で REST APIを使用)

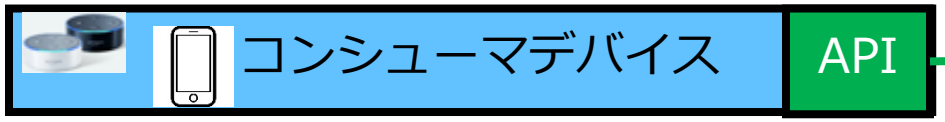
①、②を目指す方がどう始めるかを説明していきます
(③の例は第2回のアカデミーで取り上げました：Amazon Echo)

# API・プログラミングセッションの流れ

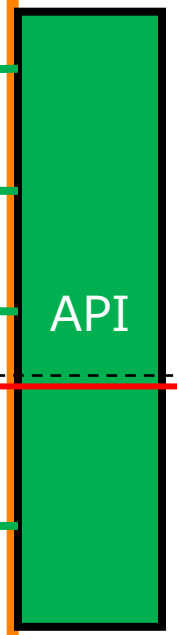
**第１回**

APIってこんなに便利！（概要説明）

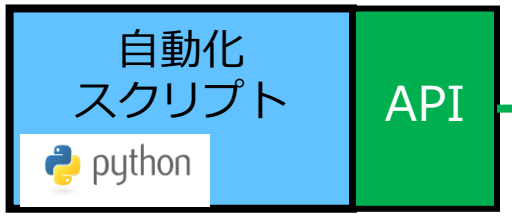
**第２回**






**第３回**


**第４回**

- APIを使用せずにモニタ➡検知➡アクションを実行
- トラブルシューティング・プロアクティブ管理
  を支援するカスタムスクリプト
- スクリプトをコミュニティで共有


**Aruba Products**


aruba
a Hewlett Packard Enterprise company

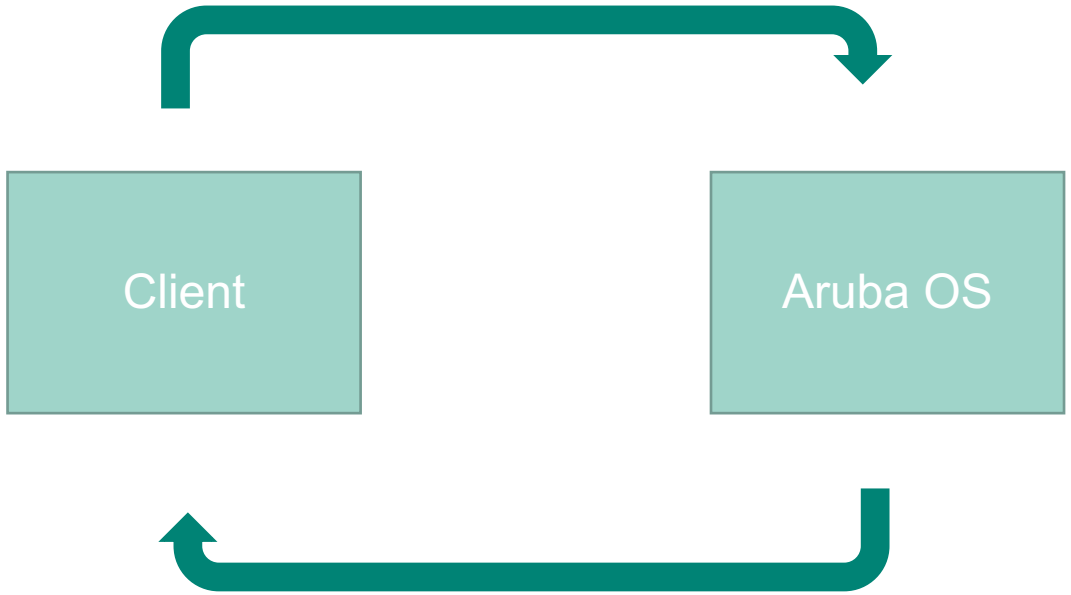
# 1. REST API

# REST API – メソッド

| | | |
|---|---|---|
| 1 | GET | Resource の取得 |
| 2 | POST | Resource の生成 |
| 3 | DELETE | Resource の削除 |
| 4 | PUT | Resource の更新 |

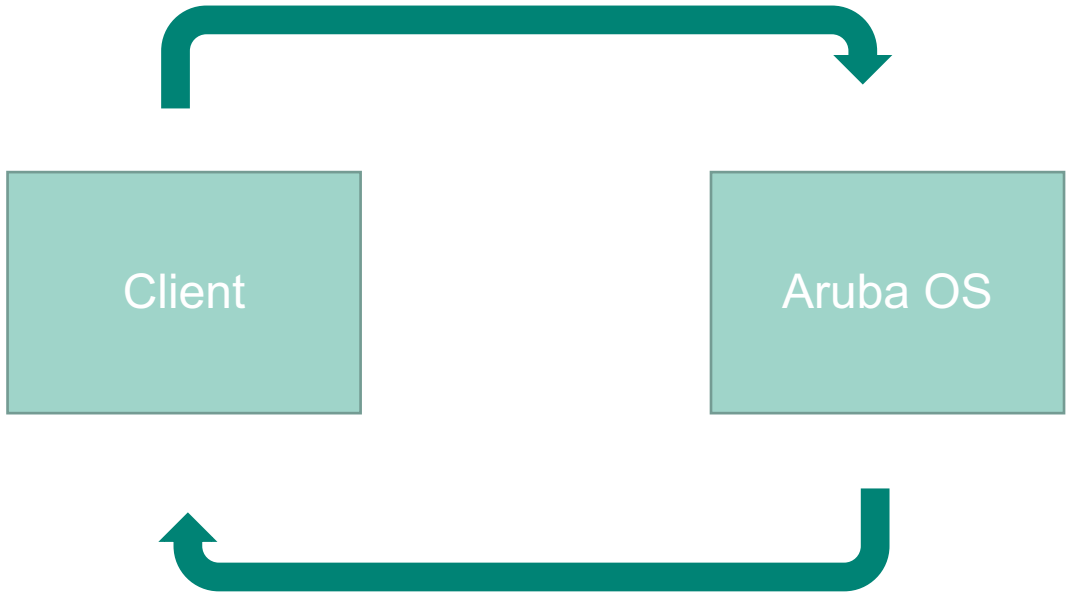
# GET / POST メソッドの処理

GET /…
Request Object




HTTP Status: 200 (OK)
Response Headers
Response Object

POST/…
Request Object




HTTP Status: 201 (Created)
Response Headers

# REST API の動作を理解するには？

動作させてみるのが理解の近道

① curl (コマンドライン)

(Postman (GUI) はオプション)

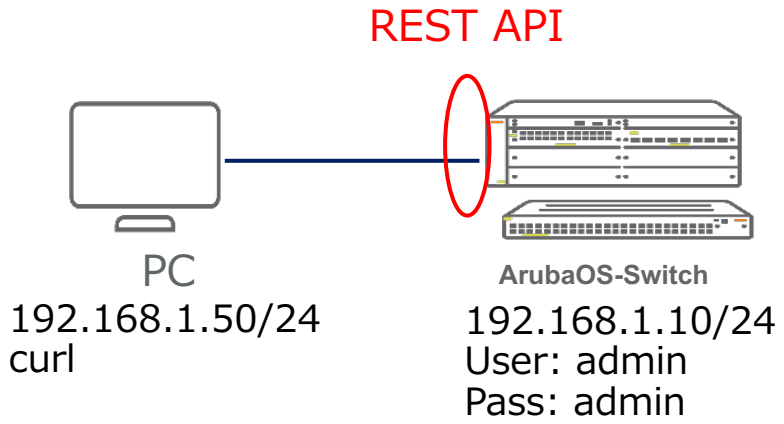
REST API の情報は製品毎に提供

② Guide, Swagger 等を参照

# 1.1 Curl – Aruba OS switch の設定例

- Aruba OS Switch (2930F) の REST API に接続します
- Firmware :　WC.16.05 (WC.16.02 以降が必要)
- IP address を使用して通信するので、Switch に IP address を設定
- Switch に管理者権限の User: admin, Password: admin を設定

```
Running configuration:

; JL258A Configuration Editor; Created on release #WC.16.05.0004
; Ver #12:08.1d.9b.3f.bf.bb.ef.7c.59.fc.6b.fb.9f.fc.ff.ff.37.ef:ba
hostname "Aruba-2930F-8G-PoEP-2SFPP"
module 1 type jl258a
snmp-server community "public" unrestricted
vlan 1
   name "DEFAULT_VLAN"
   untagged 1-10
   ip address 192.168.1.10 255.255.255.0
   exit
password manager
```



# 1.1 Curl – curl を実行

## ① REST API Session の開始

```
>curl --noproxy 192.168.1.10 -X POST http://192.168.1.10:80/rest/v1/login-sessions -d
{¥"userName¥":¥"admin¥",¥"password¥":¥"admin¥"}
```

- コマンドプロンプトから実行

- 以下が、Switchからの応答
- これ以降の RESTの処理は赤字のcookie 部分を Copy &Paste して実行

```
{"uri":"/login-sessions","cookie":"sessionId=HcE6WcKu3XGsD66HJrow0Fb4effXCBNdZ7X0Luq1k46yM972bn6WOiuvEHN92Ib"}
```


## ② VLAN 50 の作成

```
>curl --noproxy 192.168.1.10 --cookie  "sessionId=HcE6WcKu3XGsD66HJrow0Fb4effXCBNdZ7X0Luq1k46yM972bn6WOiuvEHN92Ib" -X POST
http://192.168.1.10:80/rest/v1/vlans -d {¥"vlan_id¥":50,¥"name¥":¥"vlan50_test¥"}
```

```
{"uri":"/vlans/50","vlan_id":50,"name":"vlan50_test","status":"VS_PORT_BASED","type":"VT_STATIC","is_voice_enabled":false,"is_jumbo_enabled
":false,"is_dsnoop_enabled":false,"is_dhcp_server_enabled":false}
```

aruba
a Hewlett Packard Enterprise company

# 1.1 Curl – curl を実行

```
Running configuration:

; JL258A Configuration Editor; Created on release #WC.16.05.0004
; Ver #12:08.1d.9b.3f.bf.bb.ef.7c.59.fc.6b.fb.9f.fc.ff.ff.37.ef:ba
hostname "Aruba-2930F-8G-PoEP-2SFPP"
module 1 type jl258a
snmp-server community "public" unrestricted
vlan 1
   name "DEFAULT_VLAN"
   untagged 1-10
   ip address 192.168.1.10 255.255.255.0
   exit
vlan 50
   name "vlan50_test"
   no ip address
   exit
password manager
```

- REST API 経由で vlan 50 を作成した事を
  コンフィグで確認

② REST API Session の終了

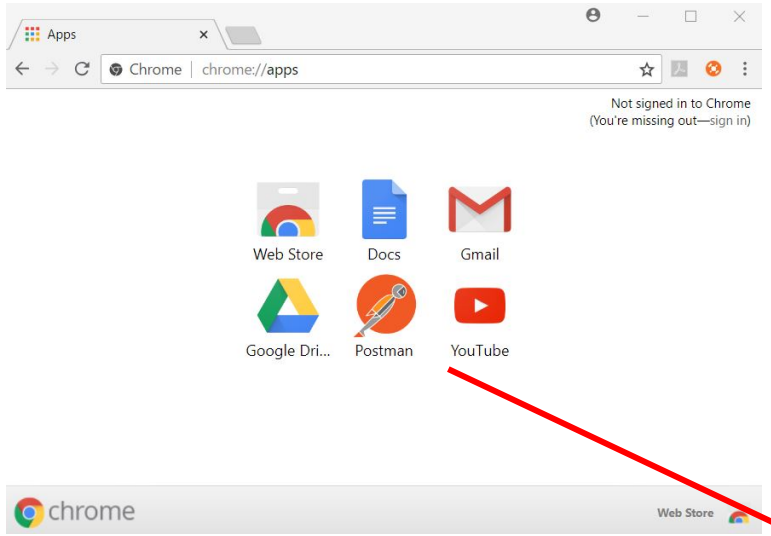
Copy & Paste

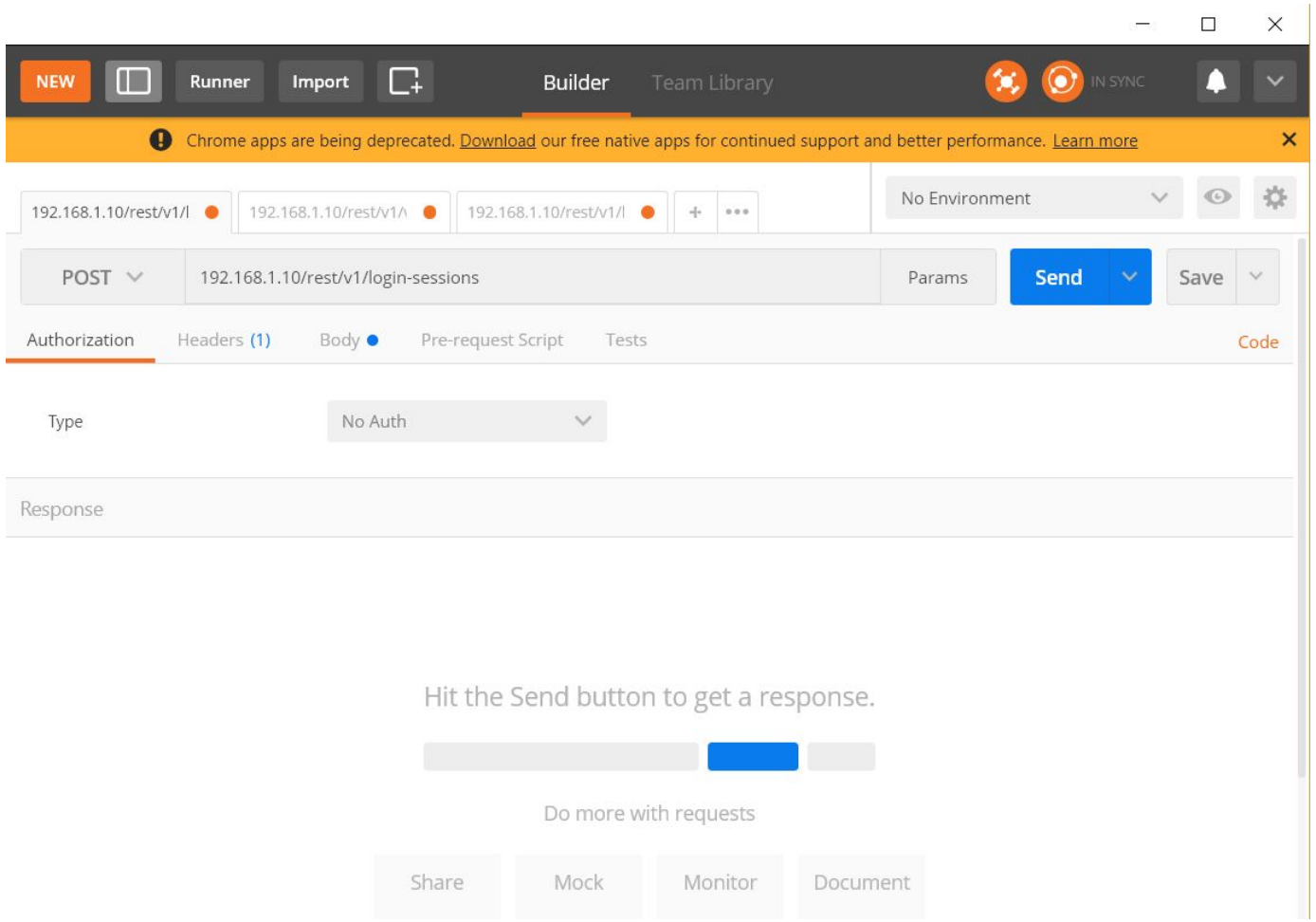
```
>curl --noproxy 192.168.1.10 --cookie "sessionId=HcE6WcKu3XGsD66HJrow0Fb4effXCBNdZ7X0Luq1k46yM972bn6WOiuvEHN92Ib"
-X DELETE http://192.168.1.10:80/rest/v1/login-sessions
```

# 1.1 Curl – まとめ

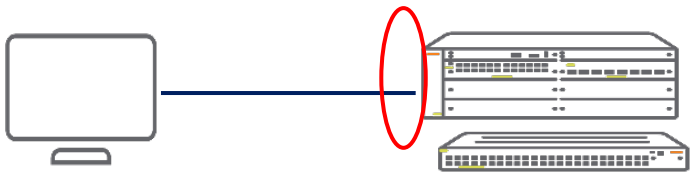
- PC のCLIから簡単に実行可能
- REST API の動作を確認することが可能
- GET/POST等の処理を行う場合は Cookie (Token) を Copy&Paste する必要あり
- １つの処理毎にCLI で実行していくので、自動化には向かない
- Windows 環境では curl を Install する必要がある

# 1.2. Postman –GUIベースのアプリケーション




REST API

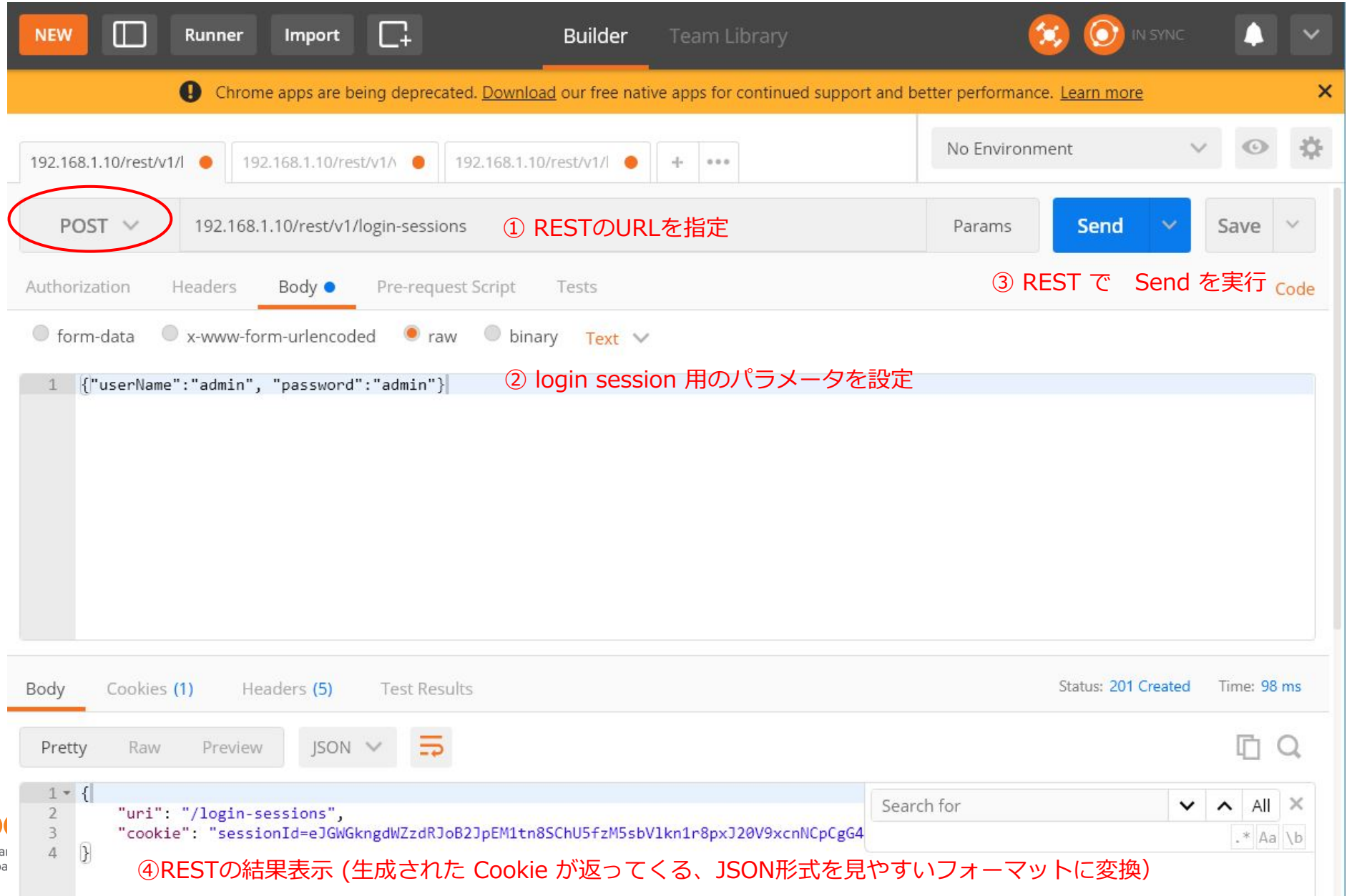


PC
192.168.1.50/24
Postman

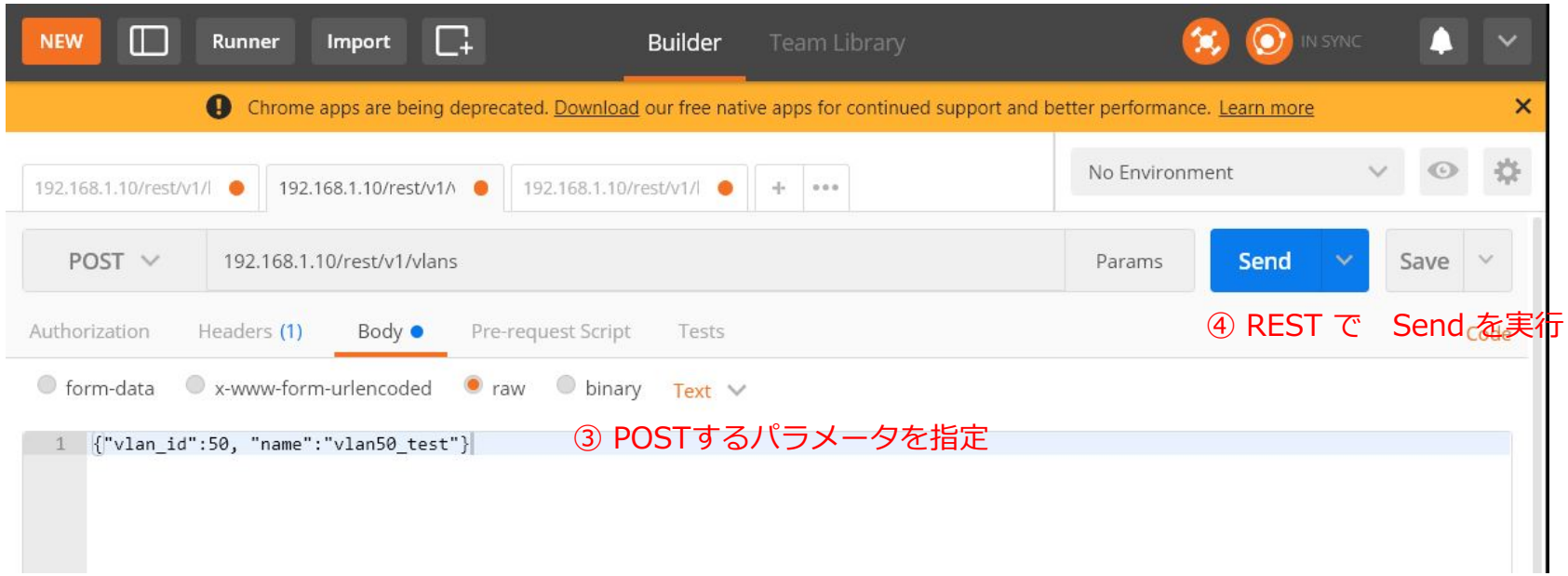
ArubaOS-Switch
192.168.1.10/24
User: admin
Pass: admin

aruba
a Hewlett Packard Enterprise company

# 1.2. Postman – REST API session の開始

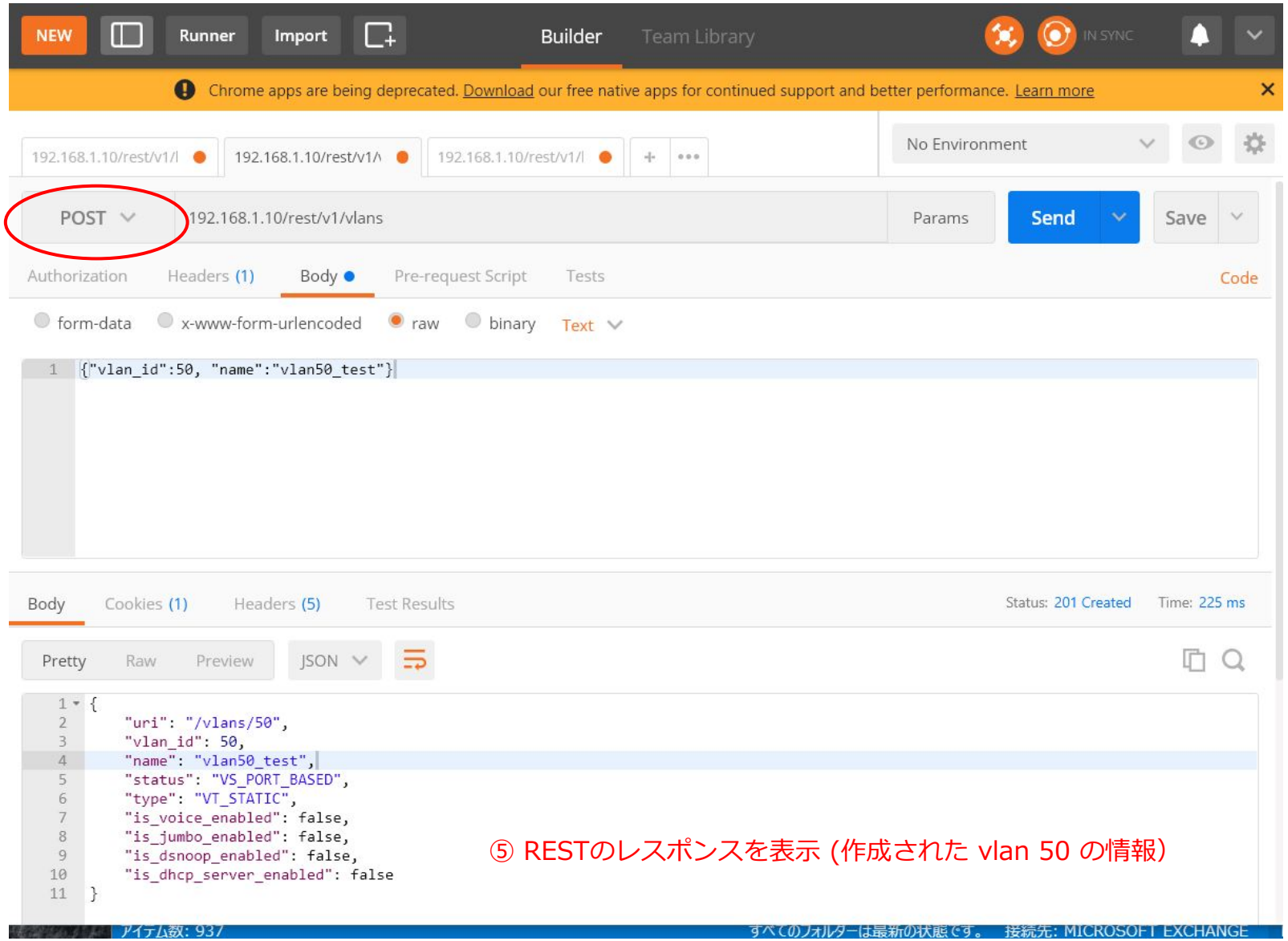


# 1.2. Postman – vlan 50 の作成

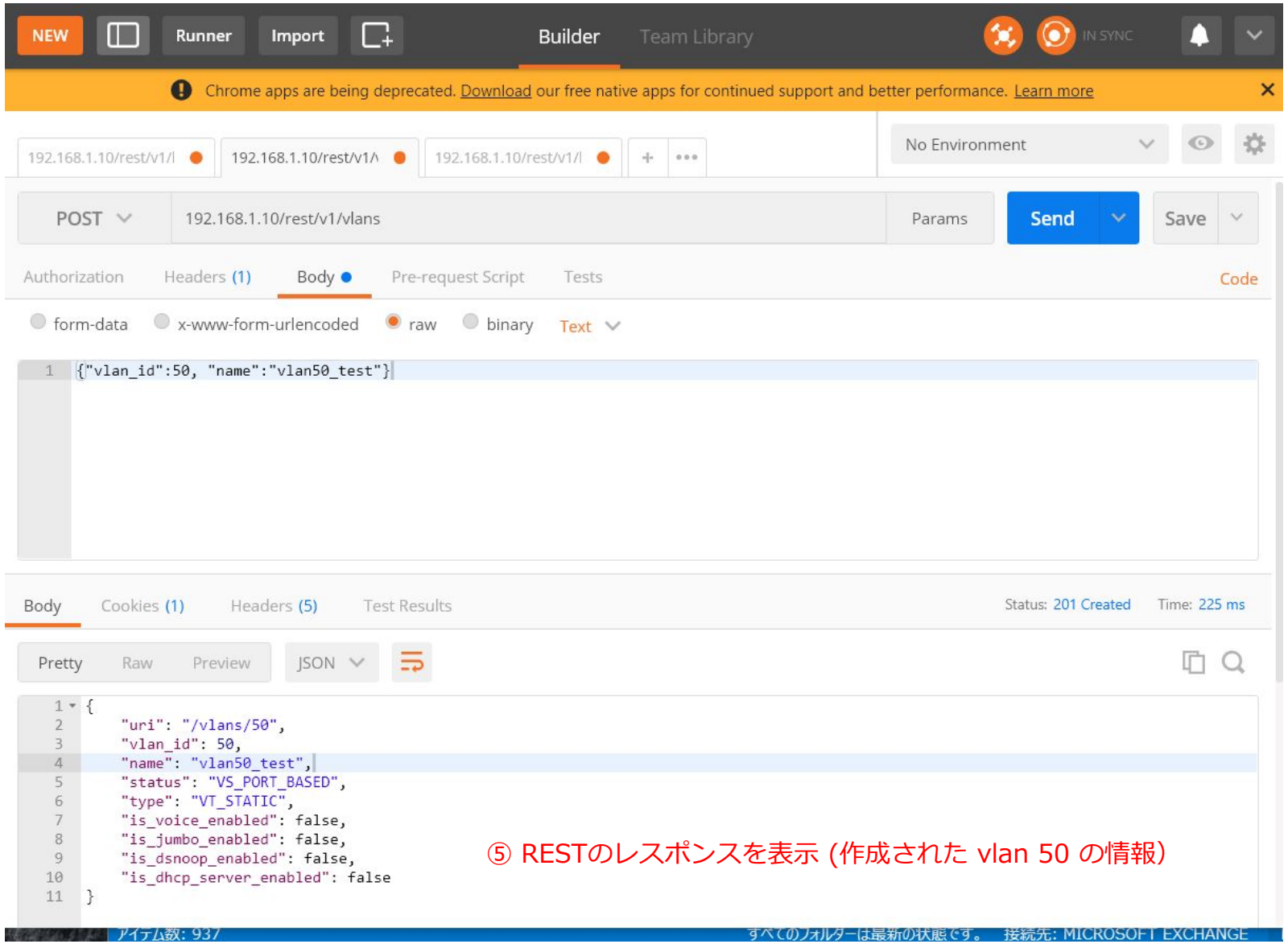

aruba
a Hewlett Packard Enterprise company

# 1.2. Postman – vlan 50 の作成




⑤ RESTのレスポンスを表示 (作成された vlan 50 の情報)

# 1.2. Postman – vlan 50 の作成

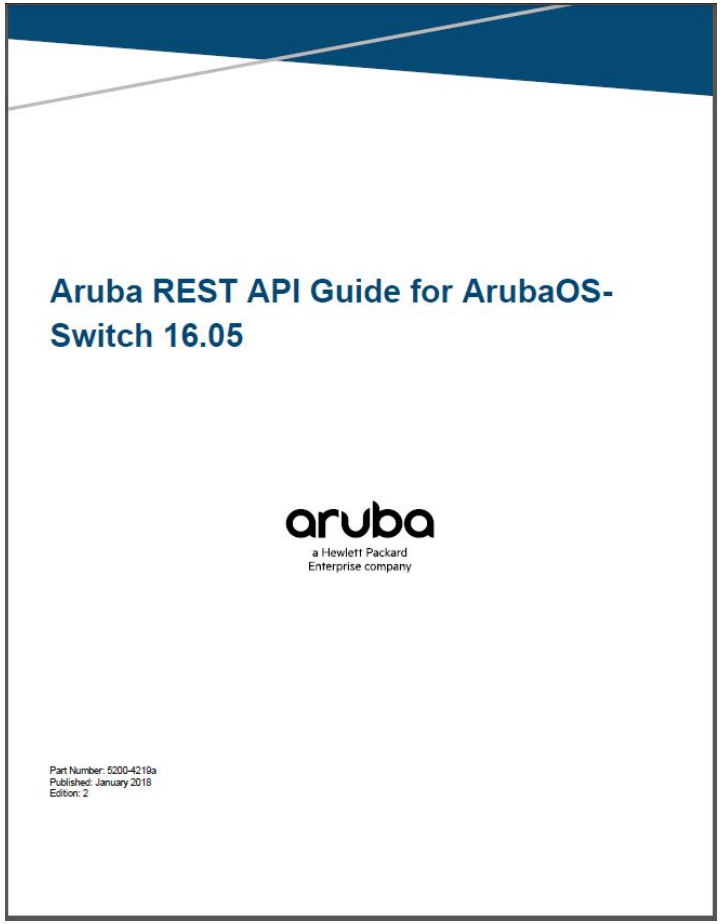



⑤ RESTのレスポンスを表示 (作成された vlan 50 の情報)

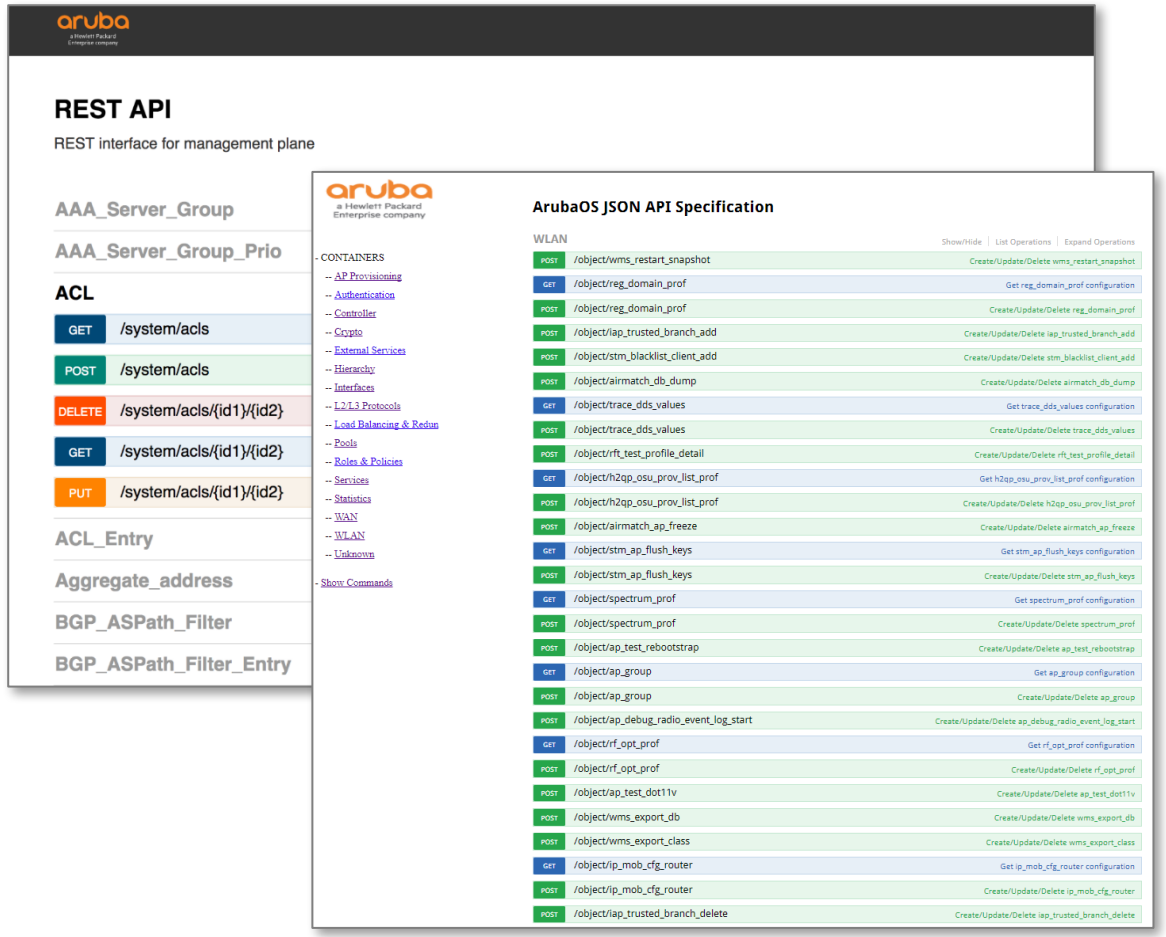
# 1.2 Postman – まとめ

- PC に Postman を install
- GUIでRESTの動作を確認可能
- URL、パラメータ等の情報は繰り返し使用可能
- Curl より使い勝手が向上する (Postman の使用はオプション)

# 1.3 REST API の情報




REST API Guide (Aruba OS switch)




Open API Browser (Swagger)
(AOS8, Central, Aruba OS-CX)

# 2. Python

# 2　Python の概要

## 何故 Python ？

- 自動化には Programming が必要

  （Java 等の他の言語を使用しても良いが…）

- Pythonは他の言語と比較すると
  - 習得し易い (文法がシンプル、例えば Java は覚える事が多い)
  - 豊富な package を利用可能 (数値処理、機械学習、 Web, REST, etc)
  - 最近、注目されている言語 (米国ではかなり前から使われている)

# 2.1 Python を使用する環境

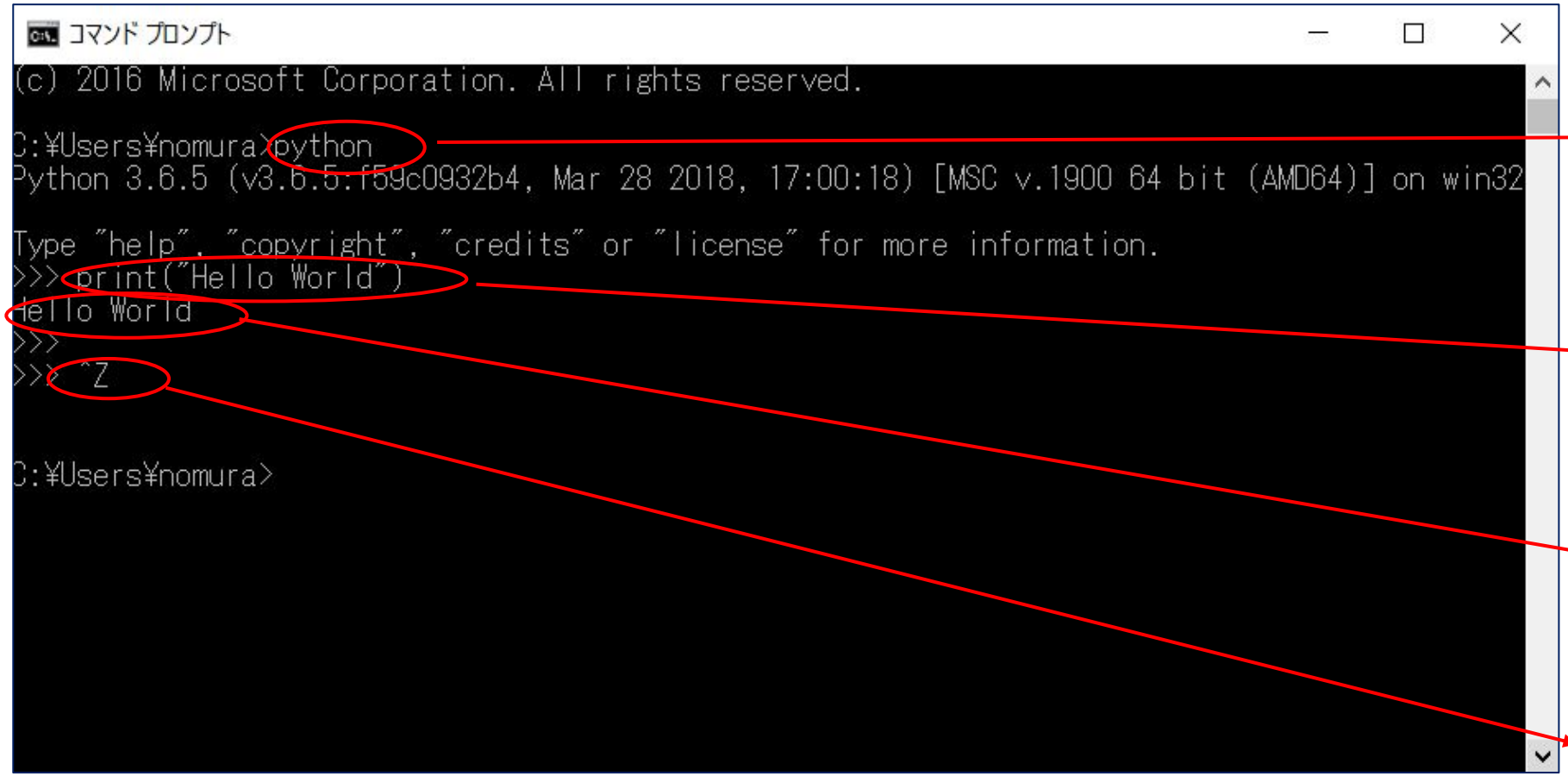
Python の環境を以下の手順で用意します

1. Python の実行環境をインストール
2. Pycharm　等の Python 開発環境をインストール
3. “pip” を使用して必要な追加 package をインストール

# 2.1.1 Python のインストール (Windows の場合)

- https://www.python.org/downloads にアクセス
- インストールするバージョン 3.X を選択
  （これから Python を始める場合は、3.X を推奨します。 2.x とは互換性が無い部分があります）
- .exe ファイルを実行してインストールを開始
- 必要に応じ “python”, “pip” コマンドへのパスを環境変数に設定
- コマンドプロンプトから “python” を実行して動作を確認

# 2.1.1 Python の動作確認①



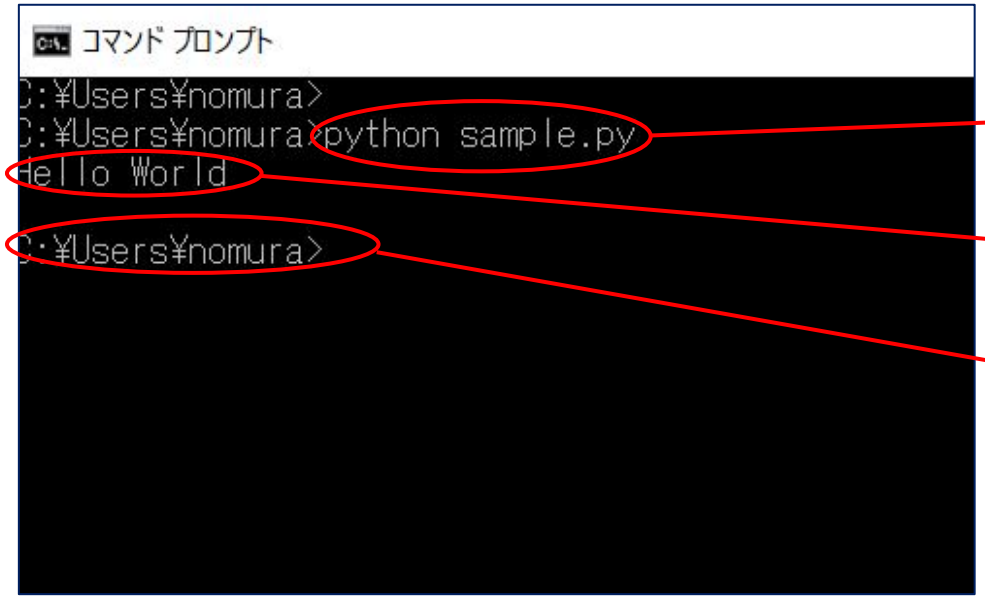

コマンドプロンプトで “python” を実行すると “>>>” がプロンプトとして表示されます。

Print(“Hello World”) を実行します。このモードでは Python の処理を1行づつ逐次的に実行します。

実行結果として Hello World が表示されます

Ctrl-Z を入力して終了

# 2.1.1 Python の動作確認②




予め作成した “sample.py” プログラムを実行

プログラムの実行結果

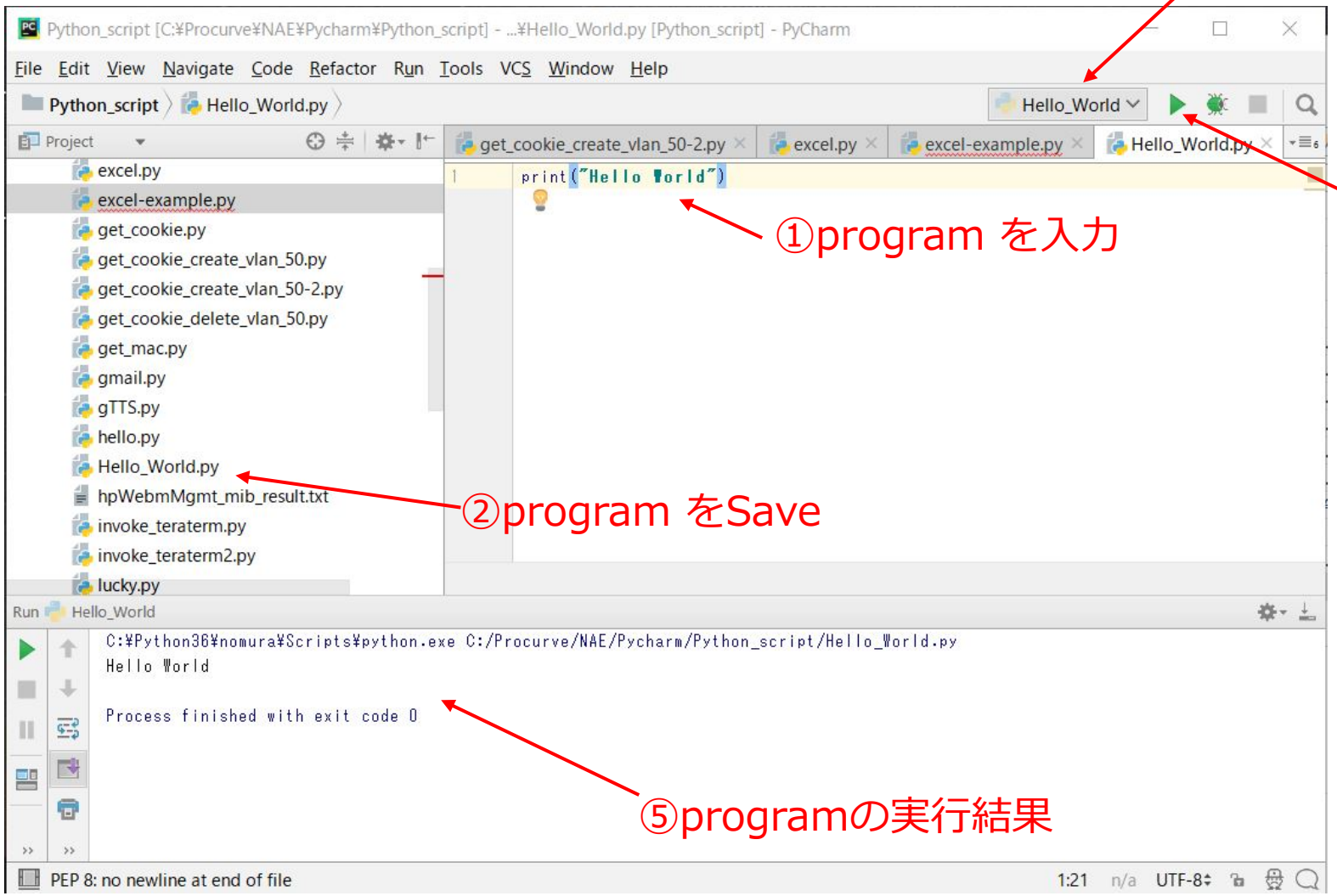
プログラムは自動的に終了


Sample.py (python プログラム)

- テキストエディタで Python プログラムを作成して実行すれば自動化は可能です。
- テキストエディタでプログラムを作成するのは、エラーが生じやすく、効率が悪いので、Pycharm 等の実行環境の使用を推奨します。

aruba
a Hewlett Packard Enterprise company

# 2.1.2 Pycahrm での動作確認




※ Pycahrm を使用する場合は、Pycahrm をインストールする必要があります

# 2.1.3 “pip” による package の追加

- “pip” コマンドは “python” インストール時に一緒にインストールされます。
- “pip” 実行時に Error が発生する場合は、proxy なしでインターネットにアクセスできる環境から実行します。
- “pip コマンドで “requests” 等の package を指定してインストールします　($ pip install requests) ※

※　Python から　REST　API を使用する場合に、requests パッケージが必要になります

# 2. Python – まとめ

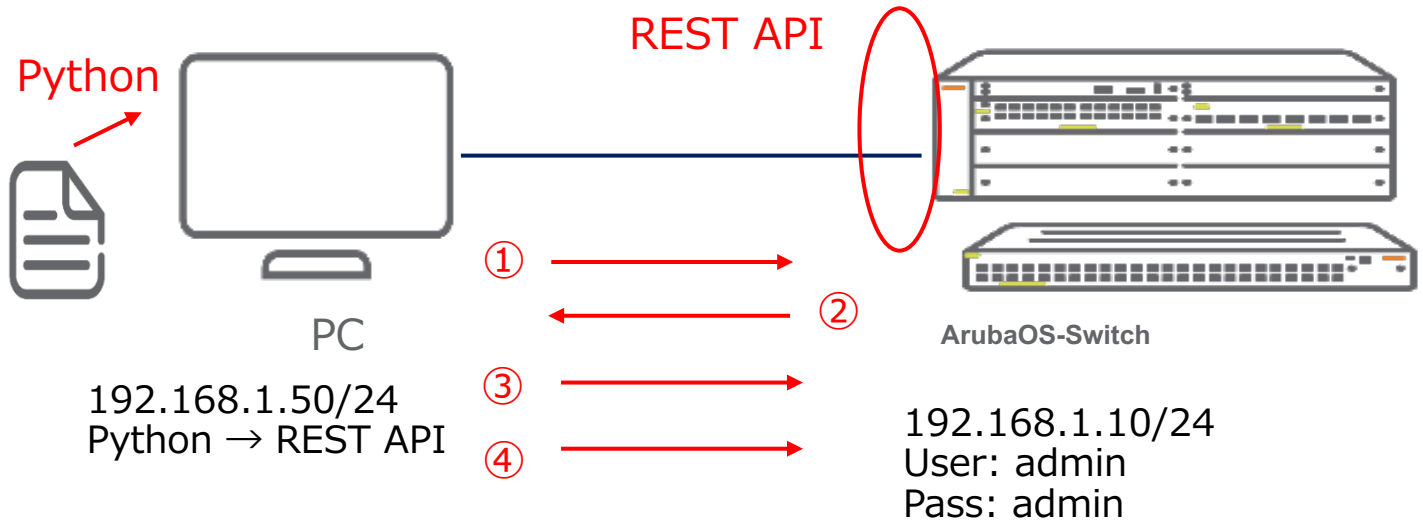
- Python には実行の仕方が複数あります。
  - コマンドプロンプトから1行づつ実行
  - コマンドプロンプトからプログラムを実行
  - Pycharm 等の Python 実行環境

- Pycharm 等の実行環境を使用することで、Error を減らし
  開発効率を上げるこができます

- pip コマンドを使用して追加パッケージを追加します

# 3. Python + REST API

# 3.1 Python + REST API (vlan 50 を作成する例)




① REST session を開始する POST メソッドをスイッチに送信
② スイッチから Cookie (Token) が返信される
③ スイッチから vlan50 を作成する POST メソッドをスイッチに送信
④ REST session を終了する DELETE メソッドをスイッチに送信

# 3.1 Python + REST API (vlan 50 を作成する例 - 全体)

```
import requests

url = "http://192.168.1.10/rest/v1/login-sessions"
payload = "{¥"userName¥": ¥"admin¥", ¥"password¥": ¥"admin¥"}"
post_response = requests.request("POST", url, data=payload)
cookie = post_response.json()['cookie']

headers = {'cookie':cookie}
payload = “{¥”vlan_id¥“: 50, ¥”name¥“: ¥”vlan50_test¥“}”
url = "http://192.168.1.10/rest/v1/vlans"
vlan_response =  requests.request("POST", url, headers=headers, data=payload)

url = "http://192.168.1.10/rest/v1/login-sessions"
delete_response = requests.request("DELETE", url, headers=headers)
```

- 上記の Python script を実行すると、 vlan50 が　REST API 経由で作成されます
- Cookie (Token) はプログラムの中で変数として受け渡しされます
- 人が介在せずに自動的に処理が行われます

aruba
a Hewlett Packard Enterprise company

# 3.1 Python + REST API (vlan 50 を作成する例 ①Session開始)

```
import requests
url = http://192.168.1.10/rest/v1/login-sessions
payload = "{¥"userName¥": ¥"admin¥", ¥"password¥": ¥"admin¥"}"
post_response = requests.request("POST", url, data=payload)
cookie = post_response.json()['cookie']
```

REST API を使用するために requests package を import ※1

REST の session を開始する url を指定

login 用に user, password を指定

REST API のPOSTメソッドで session 開始

Cookie を変数にセット (login-sessions で取得したもの)

sesiionId =… の部分を
変数 cookie に格納

REST API の response をJSON 形式

```
{'uri': '/login-sessions', 'cookie': 'sessionId=BacmWDOQHvnkq5foJZBXwx9iFDrxC5YeghZlExEgFff0a3EyprGeSQLSeyumdqr'}
```

※1 pip を使用して予め “requests” package をインストールしておく必要があります

# 3.1 Python + REST API (vlan 50 を作成する例 ② vlan 作成)

```
headers = {'cookie':cookie}

payload = “{¥”vlan_id¥“: 50, ¥”name¥“: ¥”vlan50_test¥“}”

url = http://192.168.1.10/rest/v1/vlans

vlan_response =  requests.request("POST", url, headers=headers, data=payload)
```

REST処理用の Header に先程変数に格納した cookie をセット

作成する vlan のパラメータをセット

Vlan を操作する url を指定

Vlan 50 を生成する POST メソッドを実行

REST API の response (作成された vlan 50 の情報)

```
{'uri': '/vlans/50', 'vlan_id': 50, 'name': 'vlan50_test', 'status': 'VS_PORT_BASED', 'type': 'VT_STATIC', 'is_voice_enabled': False,
 'is_jumbo_enabled': False, 'is_dsnoop_enabled': False, 'is_dhcp_server_enabled': False}
```

# 3.1 Python + REST API (vlan 50 を作成する例 ③session終了)

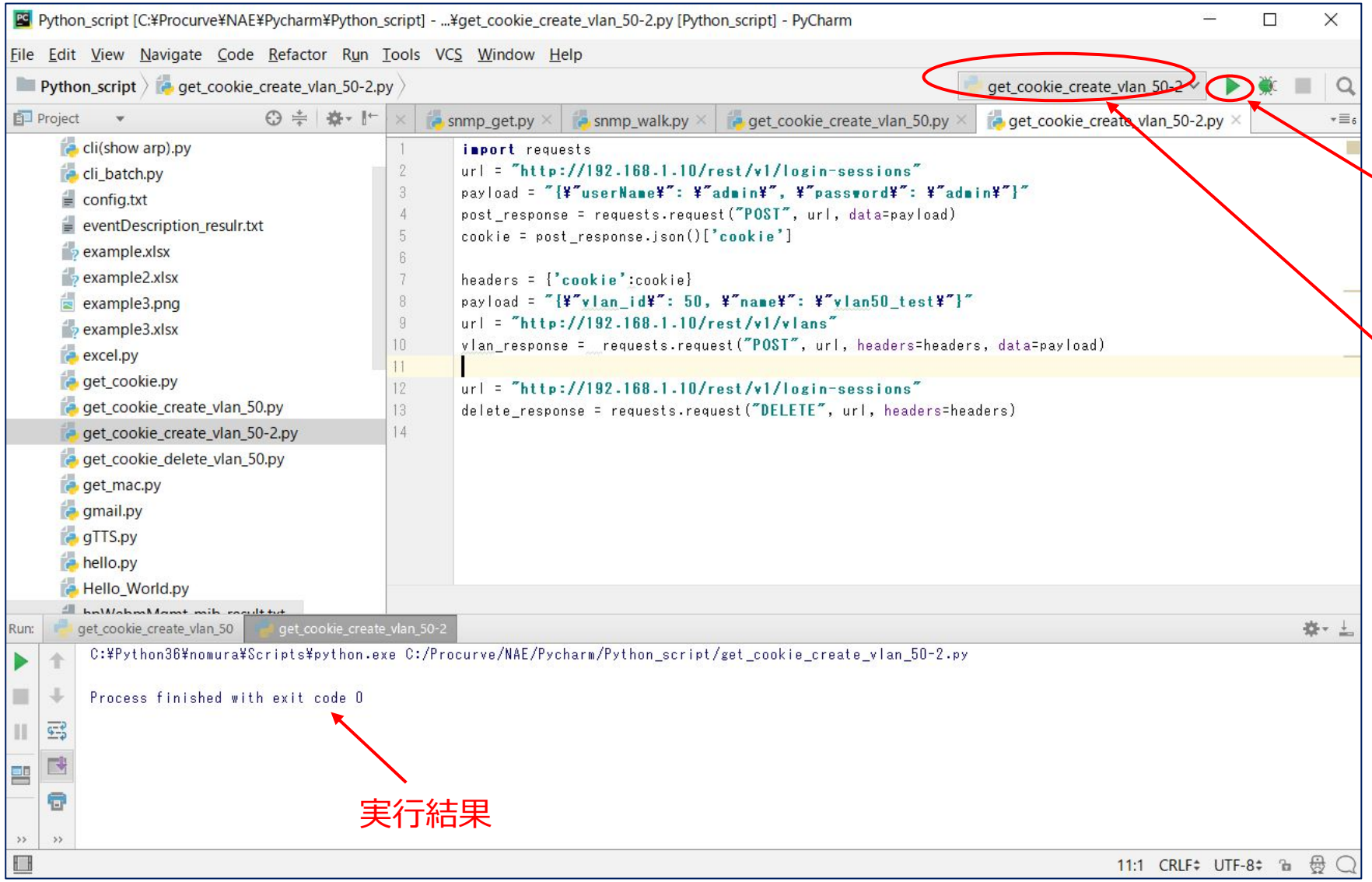
```
url = http://192.168.1.10/rest/v1/login-sessions

delete_response = requests.request("DELETE", url, headers=headers)
```

REST APIの session を終了する url を指定

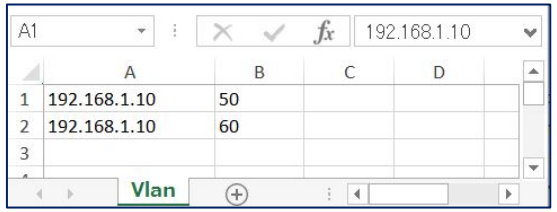
DELETE メソッドで終了

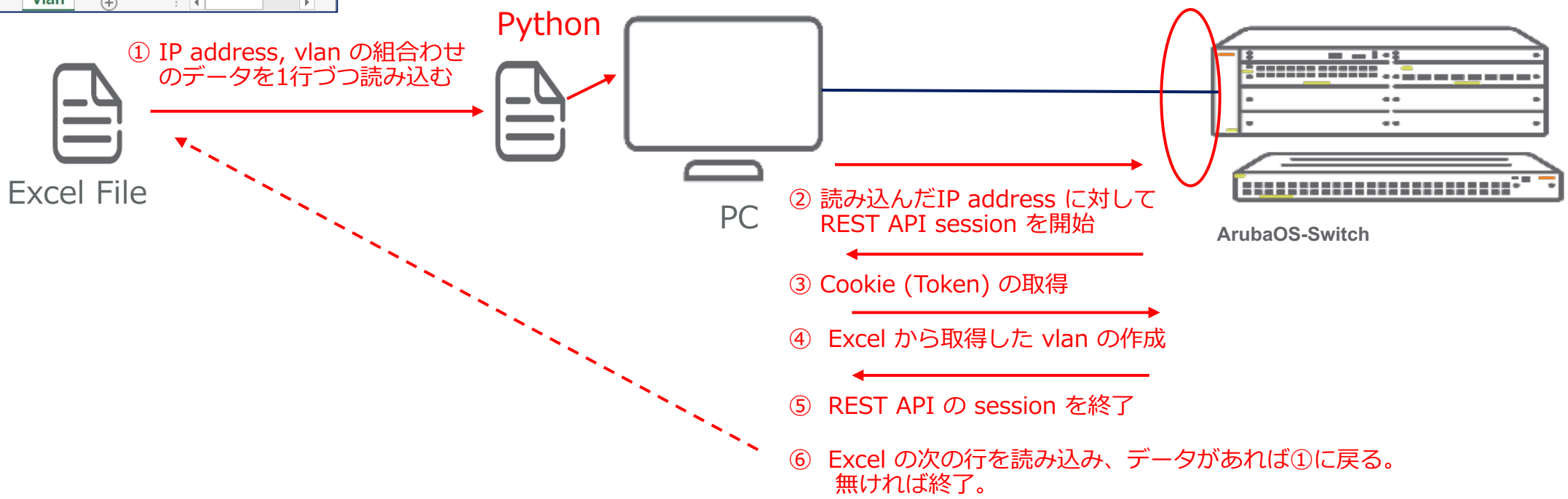
Cookie がセットされている header 情報を再度使用

# 3.1 Python + REST API (④ Pycharm の実行画面)



# 3.2 Python + REST API (Excelからデータを読み込む例)




最初のサンプル・プログラムとの違い

- Program に固定で書き込まれていた IP address, vlan の値を Excel File から読み込むように変更。
- プログラムをループ構造にすることで、複数のスイッチに対して複数の Vlan の作成を一括処理できるように変更。

aruba
a Hewlett Packard Enterprise company

# 3.2 Python + REST API (Excelからデータを読み込む例 –全体)

```
import openpyxl
import requests

wb = openpyxl.load_workbook('example.xlsx')
sheet = wb.active

i = 1
while sheet.cell(row=i, column=1).value != None:
    url = "http://" + sheet.cell(row=i, column=1).value + "/rest/v1/login-sessions"
    payload = "{¥"userName¥": ¥"admin¥", ¥"password¥": ¥"admin¥"}"
    post_response = requests.request("POST", url, data=payload)
    cookie = post_response.json()['cookie']

    headers = {'cookie': cookie}
    url = "http://" + sheet.cell(row=i, column=1).value + "/rest/v1/vlans"
    vlan = sheet.cell(row=i, column=2).value
    payload = "{¥"vlan_id¥":" +  str(vlan) + ", ¥"name¥": ¥"test_vlan_" + str(vlan) + "¥"}"
    vlan_response = requests.request("POST", url, headers=headers, data=payload)

    url = "http://" + sheet.cell(row=i, column=1).value + "/rest/v1/login-sessions"
    delete_response = requests.request("DELETE", url, headers=headers)

    i = i + 1
```

Excelファイルを操作するために openpyxl package を import *1

Excelファイルのオープン
Excel sheet の指定

Excel のデータが無くなるまで、
繰り返し処理

* REST の session 開始
* 指定した vlan の作成
* REST の session 終了

※1 pip を使用して予め “openpyxl” package をインストールしておく必要があります

# 3.2 Python + REST API (Excelからデータを読み込む例 – while loop で繰り返される処理部分)

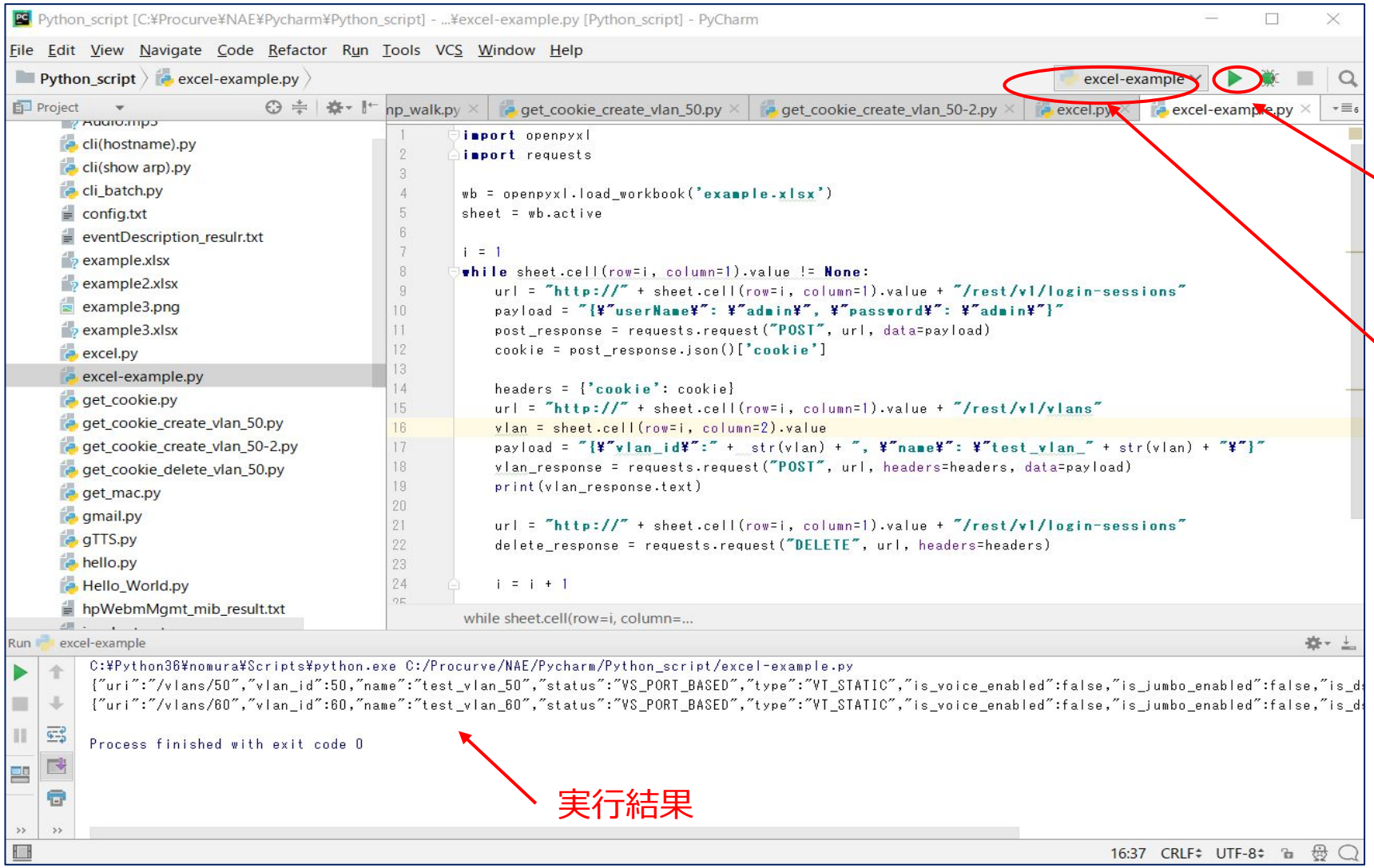
```
url = "http://" + sheet.cell(row=i, column=1).value + "/rest/v1/login-sessions"
payload = "{¥"userName¥": ¥"admin¥", ¥"password¥": ¥"admin¥"}"
post_response = requests.request("POST", url, data=payload)
cookie = post_response.json()['cookie']

headers = {'cookie': cookie}
url = "http://" + sheet.cell(row=i, column=1).value + "/rest/v1/vlans"
vlan = sheet.cell(row=i, column=2).value
payload = "{¥"vlan_id¥":" +  str(vlan) + ", ¥"name¥": ¥"test_vlan_" + str(vlan) + "¥"}"
vlan_response = requests.request("POST", url, headers=headers, data=payload)

url = "http://" + sheet.cell(row=i, column=1).value + "/rest/v1/login-sessions"
delete_response = requests.request("DELETE", url, headers=headers)
```

1列目から IP address を読み込んで、url を完成

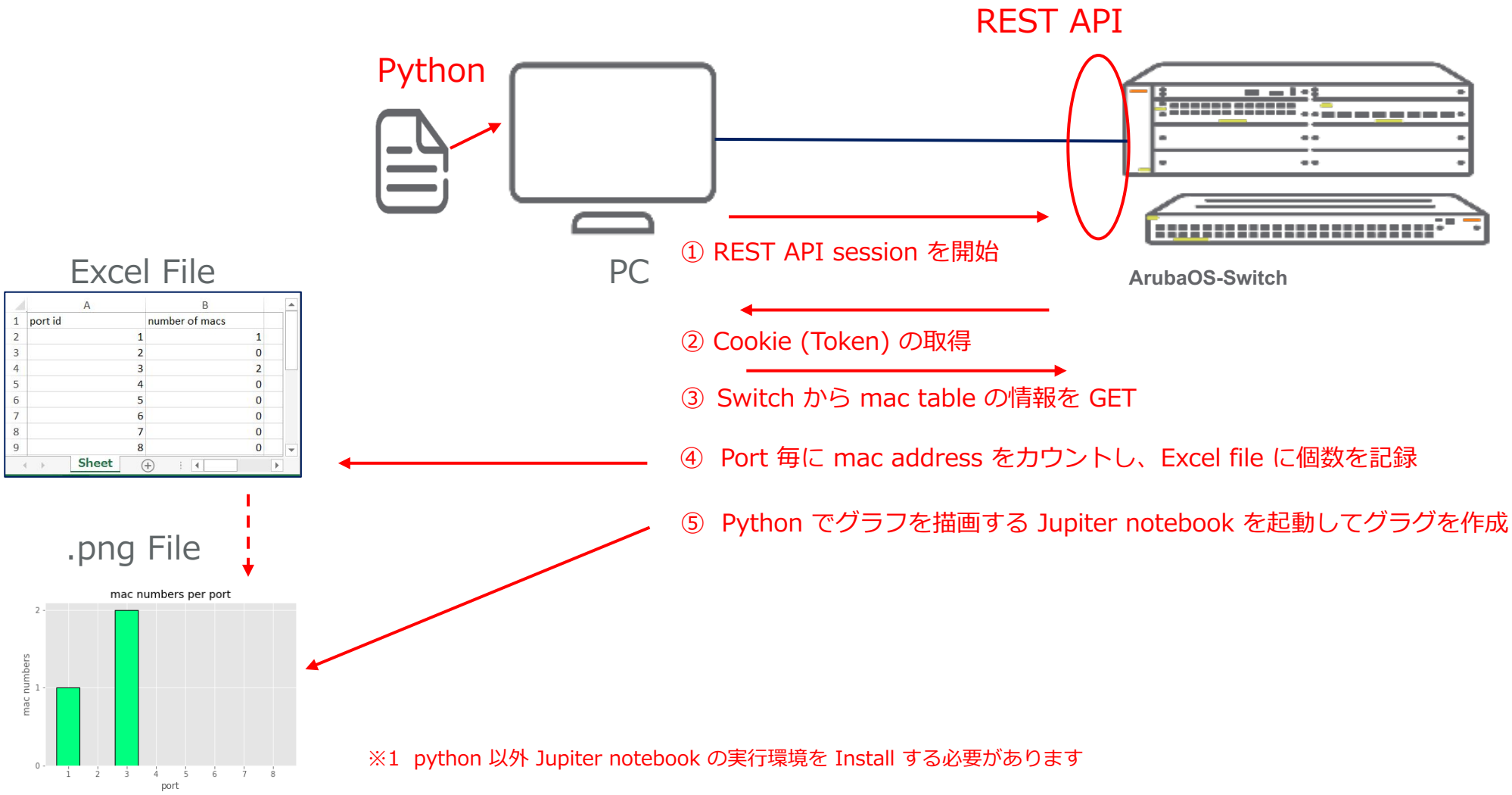
2列目から 作成する vlan の ID を取得

Vlan 名の文字列を作成

# 3.2 Python + REST API (Excelからデータを読み込む例 – Pycharmの実行画面)



# 3.3 Python を使用したその他の例 (数値データのグラフ化)




※1 python 以外 Jupiter notebook の実行環境を Install する必要があります

# 3. Python + REST API– まとめ

- Python の requests package で REST API と連携可能
- Python の openpyxl package で Excel File の操作が可能
- Jupiter notebook をインストールすることでグラフ表示が可能
- Package を追加することで様々な自動化ツールの作成が可能

# API プログラミング– まとめ

| プログラミンに対する目標レベル | 使用するツール |
| --- | --- |
| REST API の動作の理解 | Curl（Postman はオプション）<br>Swagger, REST API Guide で詳細確認<br>(Aruba OS switch 等で確認可能) |
| REST API の動作の自動化の環境構築 (Python) | Python 3.x をインストール<br>“request” package の追加 (REST API用)<br>Python 実行環境（Pycharm)の追加 (推奨) |
| 自動化のオプション | “openpyxl” package の追加 → Excel File との連携<br>Jupiter notebook の追加 → グラフの作成 |

# Airheads School – プログラミングのご紹介

## プログラミング①

- CURL, Python, Pycharm の実行環境の構築
- CURL, Python, Pycharm をご自身のPCにインストールします
- インターネットから上記ソフトをインストールできるWindows PCが必要です。
- Windows10 (64bit) をご用意ください
- プログラミング②を行うためには、この①で構築する動作環境が必要です
- ご自身で動作環境を構築できる方は受講不要です　（REST API、Pythonを始める方向けです）

第1回目の開催は、8/24(金)を予定しています

# Airheads School – プログラミングのご紹介

## プログラミング②

- プログラミング①で環境を構築したPCを使用します
- スクールでは、Aruba OS Switch の REST API を使用します
- CURLを使用して REST API の動作を確認します
- Python + Pycharm で REST API の処理を自動化するプログラムを作成します
- ご自身で CURL, Python 等を扱える方は受講不要です （REST API、Pythonを始める方向けです）

第1回目の開催は、9月を予定しています

# Aruba OS-CX –NAE のご紹介 (第4回の内容)

**第１回**

APIってこんなに便利！（概要説明）

**第２回**

コンシューマデバイス API

汎用アプリケーション API

splunk> paloalto 他社装置 API

API


Aruba Products

**第３回**

バッチ処理

データの
グラフ化

自動化
スクリプト
python

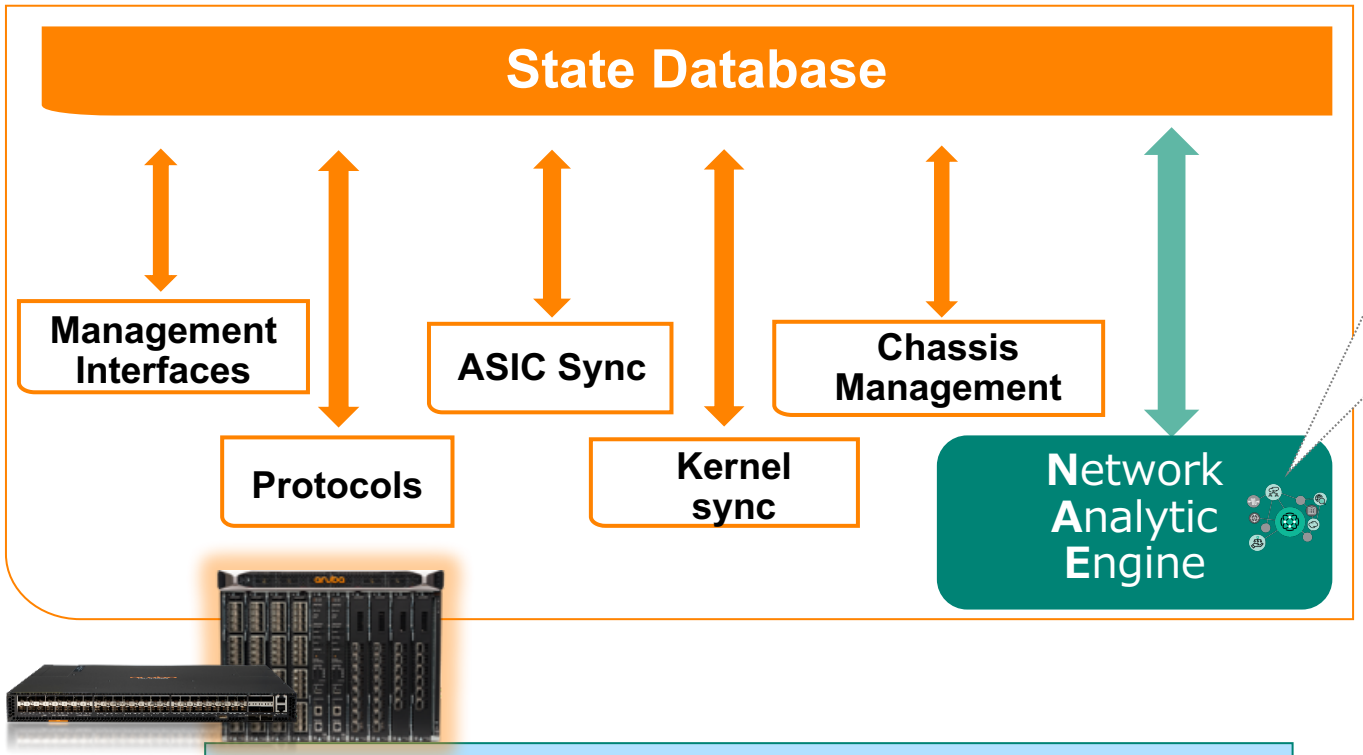
API

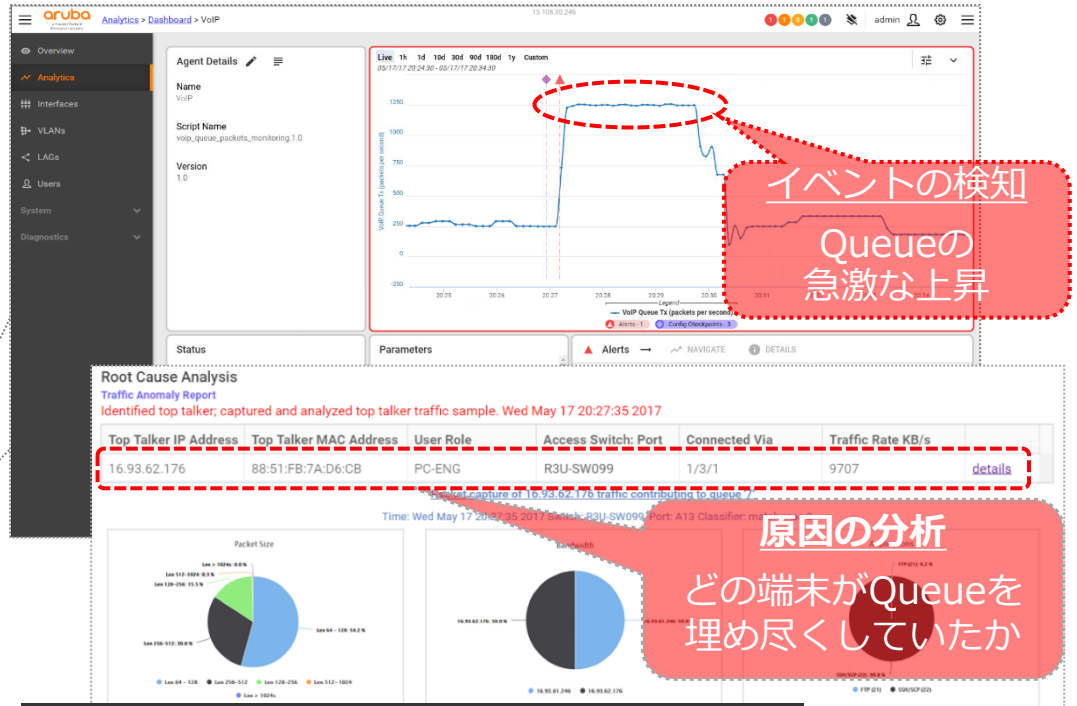
**第４回**

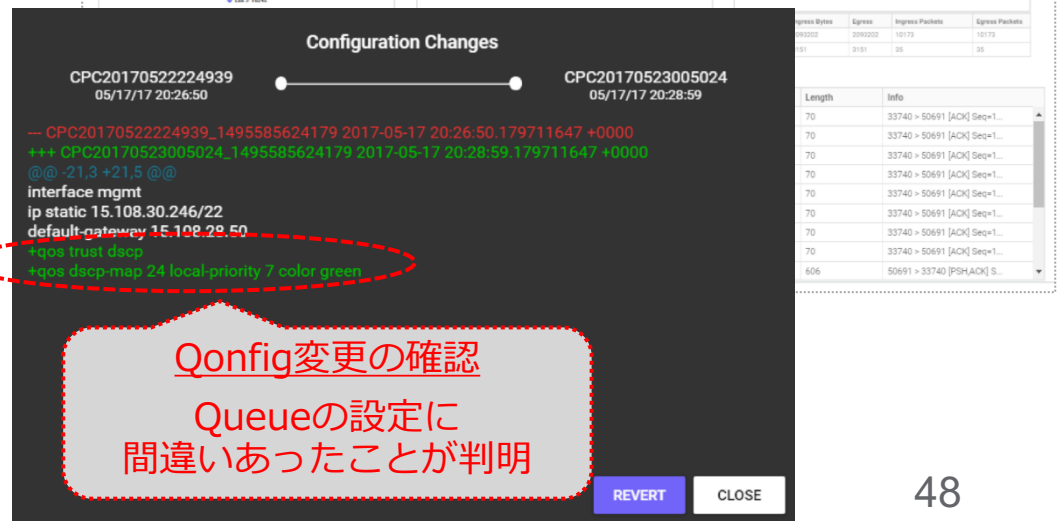
- APIを使用せずにモニタ➡検知➡アクションを実行
- トラブルシューティング・プロアクティブ管理
  を支援するカスタムスクリプト
- スクリプトをコミュニティで共有

スイッチ管理
スクリプト
python

# オンボックス解析による新しいコアスイッチ






## プロセス間の直接やりとりではなくDB経由

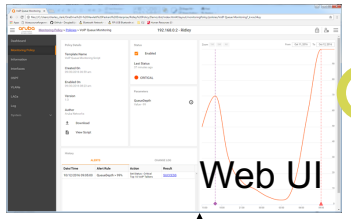
スナップショットではなく“時系列データ” 含めて
より詳細なデータを扱うことが可能

↓

**トラブルシュート時間の大幅短縮**

aruba
a Hewlett Packard Enterprise company

# 監視とトラブルシューティングが簡単に行えます

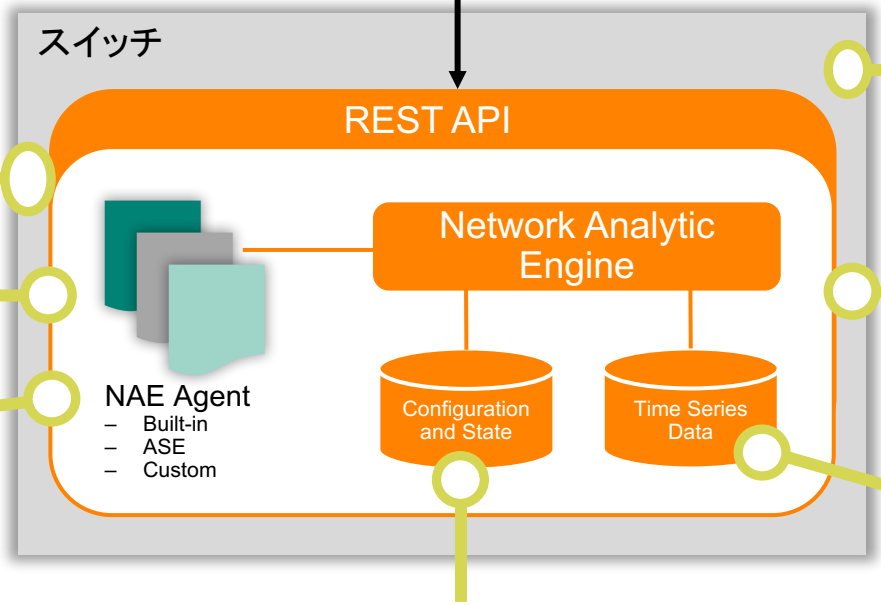



**Web UI & REST API**
Agent 毎にGUI上で詳細情報を表示

**Agent でオートメーションを実現**
Python Script を実行
パラメータ値を調整可能
Variables for persistent policy state

**ルールベースでアクションを実行**

**フレキシブルなアクション**
Alert Level の設定
CLI コマンドの実行
CLI コマンド実行結果の表示
Configuration checkpoint 差分の表示
Syslog の生成
Python script のfunction の実行




スイッチパフォーマンスへの影響無し。スイッチシステムから分離してセキュリティを確保。

Python スクリプトの Upload・表示・カスタマイズ

Time Series データベース

**広範なモニタリング対象**
コンフィグ・システムステータス・ASICの情報
ACL・ネットワークプロトコル・統計情報

Python スクリプトによりモニタリング / アクションのプログラム化 ➡ トラブルシューティングの効率向上 / プロアクティブモニタリング

# NAEの適用領域

| System Health | Network Analytics | Security | Application Visibility | Network Optimization |
|---|---|---|---|---|
| • System の統計情報をモニター。時系列情報の管理。コンフィグとの関連付け。<br>• NAEでCpu/Memoryをモニタ。閾値と比較してアラートを自動生成。DebugコマンドSyslogの自動生成。 | • ネットワークのトラブルシューティングに活用する情報の収集。<br>• BGP neighbor情報, STP の情報を収集してネットワークの最適化のための情報として活用。 | • IoTデバイスのトラフィックを監視。通常と異なる場合にアラートを生成して関連情報を自動的に収集 | • アプリケーションとデータセンタ・クラウド間のデータをモニタ。データに異常が見られる場合に必要なアクションを実施。<br>• VoIPに使用しているQueueのステータスをモニタ。異常が発生した場合に関連するトラフィック情報を収集、アラートの通知。 | • ネットワークの状態に応じて最適化のためのアクションを自動実行<br>• OSPFのlink Status に基づき経路を変更<br>• Link aggregation の統計情報のモニタ。分散アルゴリズムに問題があるLAGを管理者に通知。 |

– 別途サーバーを用意することなく監視、切り分けの自動化が可能

– スイッチ内で監視、アクションが行われるため、外部との同期が不要。 スイッチ内で直ぐにアクションの実行が可能。

– NAEを通して得られる情報を元に、プロアクティブにネットワークの最適化につなげることが可能 （Network Optimization)

# NAE Python スクリプト例 – ポート・ステータスのモニタリング

```
Manifest = {
    'Name': 'port_admin_state_monitor',
    'Description': 'Port Admin Status Monitoring Agent',
    'Version': '1.0',
    'Author': 'Aruba Networks'
}

ParameterDefinitions = {
    'port_id': {
        'Name': 'Port Id',
        'Description': 'Port to be monitored',
        'Type': 'string',
        'Default': '1/1/1'
    }
}


class Policy(NAE):

    def __init__(self):

        # Port status
        uri1 = '/rest/v1/system/ports/{}?attributes=admin'
        self.m1 = Monitor(
            uri1,
            'Port admin status',
            [self.params['port_id']])
```

Monitor 対象の指定

Agent が定期的にステータスをモニター

予め設定した条件に合致した場合にアクションを実行

DB とのやり取りはMonitor, Rule function のみで処理

aruba
a Hewlett Packard Enterprise company


# THANK YOU