Python Automation Q&A

Q1. What are the main Python data types used in automation?

```
String: locators, URLs, test data
List: multiple input values, dropdown options
Tuple: fixed test data
Dictionary: JSON/API responses
Boolean: assertion results

Example:
url = 'https://test.com'
browsers = ['chrome','firefox']
roles = ('maker','checker')
user = {'id':1,'name':'Suraksha'}
status = True
```

Q2. Difference between list, tuple, and set

```
Feature     List      Tuple     Set
Ordered    Yes       Yes       No
Mutable    Yes       No        Yes
Duplicates Allowed   Allowed   Not allowed
Usage      Dynamic data Fixed data Unique values
```

Q3. How do you handle exceptions in Python Selenium scripts?

```
Use try-except-finally to handle unexpected errors.

Example:
try:
    driver.find_element(By.ID,'login').click()
except NoSuchElementException:
    print('Login button not found')
finally:
    driver.quit()
```

Q4. Explain Python functions with automation examples

```
Functions allow code reuse.

def login(driver,user,pwd):
    driver.find_element(By.ID,'user').send_keys(user)
    driver.find_element(By.ID,'pass').send_keys(pwd)
    driver.find_element(By.ID,'submit').click()
```

Q5. Difference between shallow copy and deep copy

```
Shallow copy: references nested objects
Deep copy: creates independent copies

Example:
import copy
a = [[1,2],[3,4]]
b = copy.copy(a)  # shallow
c = copy.deepcopy(a)  # deep
```

Q6. Explain OOP concepts with automation examples

```
Class: blueprint
Object: instance
Inheritance: reuse methods
Polymorphism: same method, different behavior
Encapsulation: hide internal details

Example (POM):
class LoginPage:
    def __init__(self, driver):
        self.driver = driver
    def login(self, user, pwd):
        self.driver.find_element(By.ID,'user').send_keys(user)
        self.driver.find_element(By.ID,'pass').send_keys(pwd)
        self.driver.find_element(By.ID,'submit').click()
```

Q7. What are Python decorators and how are they used in automation?

```
Decorators add extra behavior to functions like logging or retries.

Example:
def log_test(func):
    def wrapper():
        print(f'Running {func.__name__}')
        func()
    return wrapper

@log_test
def test_login():
    print('Login executed')
```

Q8. How do you handle files in Python automation?

```
Read/write CSV, JSON, or TXT files

Example:
with open('data.txt','r') as f:
```

```
    lines = f.readlines()
with open('results.txt','w') as f:
    f.write('Test Passed')
```

Q9. How do you handle JSON in Python automation?

```
import json
resp = '{"id":1,"status":"ok"}'
data = json.loads(resp)
assert data['status'] == 'ok'
```

Q10. How do you use assert in Python?

```
Used to validate conditions.

Example:
assert 'Dashboard' in driver.title, 'Title mismatch'
```

Q11. Difference between is and ==

```
is: identity check (memory)
==: value equality

Example:
a = [1,2]
b = [1,2]
print(a == b)  # True
print(a is b)  # False
```

Q12. What are Python modules and packages?

```
Module: single .py file
Package: collection of modules with __init__.py

Example: selenium.webdriver is a package.
```

Q13. Explain Python virtual environment

```
Isolates project dependencies and avoids version conflicts

Example:
python -m venv .venv
source .venv/bin/activate
```

Q14. How to read environment variables in Python

```
import os
db_user = os.getenv('DB_USER')
# Used for secure credentials in CI/CD pipelines
```

Q15. What are Python generators?

```
Memory-efficient, yields values one by one

Example:
def test_data():
    for i in range(5):
        yield i
```

Q16. Difference between @staticmethod and @classmethod

```
Static Method: Utility function, not bound to class/object
Class Method: Factory method, bound to class

Example:
class Utils:
    @staticmethod
    def add(a,b): return a+b
    @classmethod
    def create(cls): return cls()
```

Q17. How do you use logging in Python automation?

```
import logging
logging.basicConfig(level=logging.INFO)
logging.info('Test started')
```

Q18. Explain Python unittest basics

```
Setup/teardown: setUp / tearDown
Assertions: assertEqual, assertTrue

Example:
import unittest
class TestLogin(unittest.TestCase):
    def test_login(self):
        self.assertTrue(True)
```

Q19. How do you parametrize tests in Python?

```
Using pytest parametrize:
import pytest
@pytest.mark.parametrize('user,pwd',[('admin','123'),('guest','456')])
def test_login(user,pwd):
    assert user != ''
```

Q20. How do you connect to databases in Python tests?

```
Using mysql-connector or pyodbc:
import mysql.connector
```

```
db = mysql.connector.connect(user='root', password='pwd', database='testdb')
cursor = db.cursor()
cursor.execute('SELECT * FROM users')
# Useful for DB validation in automation
```