### PYTC3M4P01\_Activity: Maintenance best practices in action

**Activity: Maintenance best practices in action**

**Course 3: Automation and Scripting with Python**

**Module 4: Optimization and scaling**

**Lesson 3: Monitoring and maintaining automation scripts**

**Graded Activity**

**Learning Objectives:**

* Apply best practices and troubleshooting techniques to work through script errors.

# Import Section

### *----- Importable content starts here -----*

# Project: Maintenance best practices in action

# Scenario

You are a data analyst at a sports news website. You are responsible for gathering sports statistics. The company requires automated scripts to periodically update game data. You have gathering sports statistics from a sports API and automating the delivery of the data to your desktop every morning. Until now, you have been manually making note of each set of stats that gets delivered to your desktop. To make your recording process easier and less prone to errors, you have been asked to create a process that automatically records the sports data in an Excel spreadsheet each morning.

## Objective

The goal of this project is to have you apply maintenance best practices to an automation script, to improve efficiency and troubleshoot errors. You will also implement basic logging and make a small code modification to address a potential issue.

## Instructions

### To achieve your objective, you will complete four milestones to practice troubleshooting and successfully fixing errors in your code. Here’s your roadmap:

### Part 1: Recall automation script

### Part 2: Add code for Excel component

### Part 3: Troubleshooting with best practices

### Part 4: Fixing the error and rerunning

**Download this zip file** and save it on your machine’s desktop or a folder where it is easily accessible. Open the Jupyter notebook, and do all work in this notebook as you follow along with the instructions below. Detailed instructions on how to download and access these Activity or Project files can be found in the Coursera FAQ.

### 

### Part 1: Recall automation script

### Recall the script you created in the last project, that fetches the latest sports scores from the Football Data API (https://www.football-data.org/):

### import requests

### from plyer import notification

### import schedule

### import time

### # Get your API key from https://www.football-data.org/ and replace 'YOUR\_API\_KEY\_HERE'

### api\_key = 'YOUR\_API\_KEY\_HERE'

### headers = {'X-Auth-Token': api\_key}

### url = "http://api.football-data.org/v2/matches"

### def fetch\_scores():

### response = requests.get(url, headers=headers)

### try:

### data = response.json()

### # Check if the 'matches' key exists in the response

### if 'matches' in data and len(data['matches']) > 0:

### match = data['matches'][0]

### score\_line = f"{match['score']['fullTime']['homeTeam']} - {match['score']['fullTime']['awayTeam']}"

### team\_home = match['homeTeam']['name']

### team\_away = match['awayTeam']['name']

### return f"{team\_home} {score\_line} {team\_away}"

### else:

### return "No matches currently available."

### except Exception as e:

### return f"Error fetching data: {e}"

### fetch\_scores()

### 

### 

### Part 2: Add code for Excel component

Now it’s time to add the component of the code that will automatically record the sports data to an Excel spreadsheet. Since you want the data to be recorded after every notification, notice that record\_to\_excel is now within send\_notification.

import openpyxl

import datetime

def send\_notification():

match\_score = fetch\_scores()

notification.notify(

title='Live Sports Score Update',

message=match\_score,

app\_icon=None, # Path to an app icon

timeout=10, # Duration in seconds

)

# Call record\_to\_excel here, after sending the notification

record\_to\_excel(match\_score)

def record\_to\_excel(match\_data):

"""Records the match data to an Excel spreadsheet."""

try:

workbook = openpyxl.load\_workbook('sports\_data.xlsx')

sheet = workbook.active

except FileNotFoundError:

workbook = openpyxl.Workbook()

sheet = workbook.active

sheet.append(["Home Team", "Score", "Away Team", "Date"])

# Extract data (assuming match\_data is in the format "HomeTeam Score - Score AwayTeam")

try:

home\_team, score, away\_team = match\_data.split(' ')

today = datetime.date.today().strftime("%Y-%m-%d")

# Corrected line: Appending the complete date string

sheet.append([home\_team, score, away\_team, today[1]])

except ValueError:

sheet.append(["Error processing data", "", "", today])

workbook.save('sports\_data.xlsx')

# Test sending a notification

send\_notification()

schedule.every().day.at("08:00").do(send\_notification)

# Test the scheduling of notifications (this will run until you stop it)

while True:

schedule.run\_pending()

time.sleep(1)

### Part 3: Troubleshooting with best practices

You should have encountered an IndexError when running the script. This error occurred because the code tried to access a part of the today variable that didn't exist. Remember that today holds the current date as a string, like "2024-10-09". When the code tried to get today[1], it was asking for the second character (since counting starts from 0 in Python). This caused the error because it was trying to use a single character ('0' in this case) as the date when adding it to the spreadsheet, which isn't the correct type of data for that column.

In order to troubleshoot this error, first read the error message. You should see an IndexError message. This means you tried to access an element in a sequence (like a string or list) using an invalid index.

Next, locate the error. The error message will point you to the line sheet.append([home\_team, score, away\_team, today[1]]). This line is responsible for adding data to the Excel sheet.

Then, examine the variables. Focus on the today[1] part. The today variable holds the current date as a string. Remember that indexing in Python starts from 0. So, today[1] tries to access the *second* character of the date string, which is not what you intend to add to the sheet.

### Part 4: Fixing the error and rerunning

Correct the code by changing today[1] to today in the sheet.append line. This will append the complete date string to the sheet, resolving the IndexError.

Finally, run your script again to confirm that the error is gone and the data is being recorded correctly in the Excel file. Good job! You successfully fixed the error using troubleshooting techniques!

In general, when you encounter an error in your Python code, don't panic! Use the following steps as a guide to troubleshoot errors:

1. **Read the Error Message:** Don't just glance at it! Carefully read the entire error message. It provides crucial information:

* **Error Type:** (e.g., TypeError, ValueError, IndexError) This tells you the general nature of the problem.
* **Line Number:** This pinpoints *where* in your code the error happened.

1. **Examine the Code:** Focus your attention on the line of code indicated in the error message, as well as the lines immediately before and after it.

* **Variable Usage:** Double-check that you're using the correct variable names and that they have the values you expect.
* **Data Types:** Make sure you're not trying to do something impossible, like dividing a string by a number.

1. **Use Debugging Tools:**

* **print() Statements:** Add print() statements to display the values of variables at various points. This helps you trace how the data is changing and where things might be going wrong.
* **Debugger:** If you're using an IDE (like VS Code, PyCharm), learn how to use its debugger. Debuggers let you step through your code line by line, inspect variables, and more.

1. **Isolate the Problem:** If the error is in a complex section of code, try to break it down into smaller, simpler parts. Test each part separately to narrow down the source of the error.
2. **Consult External Resources:**

* **Search Online:** Search for the exact error message you're getting. Chances are, someone else has encountered it before, and you might find solutions or explanations online.
* **Documentation:** Refer to the official Python documentation or documentation for any libraries you're using to understand how functions and methods are supposed to work.

**Question 1** **- multiple choice, shuffle**

What type of error is caused by the bug intentionally placed in the record\_to\_excel function?

A: TypeError

Feedback: Not quite. This occurs when an operation is performed on a data type that doesn't support it (e.g., adding a string to an integer).

B: ValueError

Feedback: Not quite. This happens when a function receives an argument of the correct type but an inappropriate value (e.g., trying to convert a non-numeric string to an integer).

\*C: IndexError

Feedback: Correct! It occurs when you try to access an element in a sequence (like a string, list, or tuple) using an index that is out of range. In this case, the code tries to access today[1], which is the second character of the date string, causing the error.

D: SyntaxError

Feedback: Not quite. This occurs when the code violates the rules of Python grammar, such as missing colons or incorrect indentation.

**Question 2** **- multiple choice, shuffle**

Which of the following is a best practice when troubleshooting errors in Python?

A: Ignoring the error message and hoping it goes away on its own

Feedback: Not quite. This is definitely not a good idea! Error messages are there to help you.

B: Trying random solutions without understanding the underlying issue

Feedback: Not quite. This might sometimes work by chance, but it's not a reliable approach. It's important to understand *why* an error is happening so you can fix it properly.

\*C: Using the print() statements or a debugger to understand the program's behavior

Feedback: Correct! This is a great way to troubleshoot! These tools allow you to see what your code is doing step-by-step.

D: Giving up immediately and assuming you can't fix the error

Feedback: Not quite. Don't get discouraged! Debugging is a skill that takes practice. With patience and persistence, you can learn to solve most errors.