PharmaSUG China 2022 - Paper HOW-151

How Can Python Really Help Our Hands-on Work?

Jie Liu, BeiGene (Beijing) Co., Ltd.;
Wei (Tony) Zhang, Pfizer (China) Research and Development Co.,Ltd. China;

ABSTRACT

Since Python is one of the most popular coding languages in the world and has been widely used in different areas, many beginners from pharmaceutical industry may have a question: Can Python help my daily work? Usually, when the beginner is starting to explore the application of Python in pharmaceutical industry, a mass of foundational syntax and elements of Python are in the learning list. It might be worth having a glance at the learning curve and understanding how python can really help our hands-on work before beginner starts to learn it.

In this half-day hands-on workshop, several scenarios including interacting with OS (Operation System), MS office documents handling, data processing and UI (User Interface) development will be provided with the solution using Python to give the attendees the concept of Python. The attendees can have the self-feeling of Python through the hands-on workshop. The foundational knowledge of Python, tips of learning Python, advantage and disadvantage of Python will also be introduced during the hands-on workshop of these scenarios. This section aims to open a door of learning Python for the attendees.

INTRODUCTION

Python is a high-level, interpreted, general-purpose programming language. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

With these powerful packages in standard library Python can be used in many different areas. This paper will mainly focus on interacting with OS (Operation System), MS office documents handling, data processing and UI (User Interface) development. Example which has already applied in pharmaceutical industry for each area will be provided and some sample codes will also be prepared for practicing.

Spyder and Jupyter Notebook within Anaconda are used for demo in this paper. So the attendees can know both Spyder and Jupyter in Anaconda.

INSTALLING ANACONDA ON WINDOWS

For this Hands-on Workshop (HOW), we recommend installing and using the Anaconda distribution of Python. Anaconda is free and can be installed on computers where you don't need to have administrator access or the ability to install new programs. It comes bundled with about 600 packages pre-installed including NumPy, Pandas, Matplotlib and SymPy.

Steps:

1. Visit *https://www.anaconda.com/*
2. Select Windows and download
3. Download the ***.exe*** installer
4. Open and run the ***.exe*** installer

5. Open the **Spyder** or **Jupyter Notebook** and run some Python code after installation is completed

INTERACTING WITH OS (OPERATION SYSTEM)

The OS module in Python provides functions for interacting with the operating system, irrespective of it being a Windows Platform, Macintosh or Linux. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

In the section 1 of HOW, we practice the OS modules in Python. Some example codes are below. More instructions and codes are in the HOW lecture.

PYTHON IMPORT OS

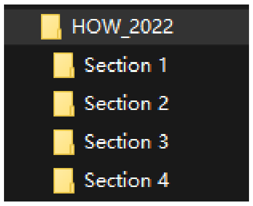
```
import os
```

CREATE DIRECTORY

There are different methods available in the OS module for creating a director. These are:

- os.mkdir()
- os.makedirs()

In this practice, you need to firstly create directories as shown in the following snapshot.



os.mkdir() method in Python is used to create a directory with specified name. If the directory already exists, FileExistsError is raised. If a parent directory in the path does not exist, FileNotFoundError is raised.

```
path='c:/HOW_2022'
os.mkdir(path)
```

The directory 'HOW_2022' is created using the above code in in your local PC. Then four directories will be created under this parent one using the below code.

```
dir1='Section 1'
dir2='Section 2'
dir3='Section 3'
dir4='Section 4'

path1=os.path.join(path, dir1)
path2=os.path.join(path, dir2)
path3=os.path.join(path, dir3)
path4=os.path.join(path, dir4)
```

```
os.mkdir(path1)
os.mkdir(path2)
os.mkdir(path3)
os.mkdir(path4)
```

os.makedirs() method in Python is used to create a directory recursively. That means while making leaf directory if any intermediate-level directory is missing, os.makedirs() method will create them all.

Please remove ‘HOW_2022’ and ‘Section 1’, ‘Section 2’, ‘Section 3’, ‘Section 4’ directories manually you have created and try the following code. Then all directories can be created.

```
path1='c:/HOW_2022/Section 1'
path2='c:/HOW_2022/Section 2'
path3='c:/HOW_2022/Section 3'
path4='c:/HOW_2022/Section 4'

os.makedirs(path1)
os.makedirs(path2)
os.makedirs(path3)
os.makedirs(path4)
```

os.makedirs() method will create all unavailable/missing directories in the specified path. ‘HOW_2022’ will be created first then ‘Section 1’, ‘Section 2’, ‘Section 3’, ‘Section 4’ directories will be created.

CHECKING EXISTING DIRECTORY

os.path.exists() method in Python is used to check whether the specified path exists or not. The output is True or False

```
isExist = os.path.exists(path)
print(isExist)
```

We usually use this method as the condition for the following code action as below example. If the directory does not exist, it will be created.

```
if isExist==False:
    os.makedirs(path1)
```

RETURN CURRENT WORKING DIRECTORY

os.getcwd() returns current working directory of a process. The following example shows the usage of getcwd() method.

```
cwd=os.getcwd()
print("Current working directory:", cwd)
```

CHANGE WORKING DIRECTORY

os.chdir() method in Python is used to change the current working directory to specified path.

```
os.chdir(path1)
cwd=os.getcwd()
print("Current working directory:", cwd)
```

RENAME DIRECTORY

os.rename() is used to change the name of directory or files. You can change 'C:/HOW_2022' to current working directory, then input the old name and new name of the directory under 'C:/HOW_2022' in this method as the following example.

```
os.chdir('c:/HOW_2022')
os.rename('Section 1', 'Section 1 New Name')
```

LIST OF ALL FILES OR DIRECTORIES IN THE SPECIFIED DIRECTORY

os.listdir() method in Python is used to get the list of all files and directories in the specified directory. If we don't specify any directory, then the list of files and directories in the current working directory will be returned.

```
path='c:/HOW_2022'
dir_list=os.listdir(path)
```

The output is a list which contains all files and directories name as below.

```
['Section 1 New Name', 'Section 2', 'Section 3', 'Section 4']
```

However, we usually need both path and file name for each file or directory in a specified directory. Then we can change the code as the following example.

```
name_list=[]
for file in os.listdir(path):
    name_list.append(os.path.join(path, file))
```

The output is a list which contains all files and directories path and name as below.

```
['c:/HOW_2022\\Section 1 New Name', 'c:/HOW_2022\\Section 2',
'c:/HOW_2022\\Section 3', 'c:/HOW_2022\\Section 4']
```

LIST OF ALL DIRECTORIES IN DIRECTORY AND SUB-DIRECTORIES

os.work() returns a generator that creates a tuple of values (current_path, directories in current_path, files in current_path). So we can list all directories, subdirectories, and files in a given directory by this function. It is a recursive function, i.e., every time the generator is called, it will follow each directory recursively to get a list of files and directories until no further sub-directories are available from the initial directory.

```
os.makedirs('c:/HOW_2022/Section 1/Test 1')
os.makedirs('c:/HOW_2022/Section 1/Test 2')
dir_list=[]
for (dir_path, dir_name, file_name) in os.walk('c:/HOW_2022'):
    for d in dir_name:
        if d!='':
            dir_list.append(os.path.join(dir_path, d))
print(dir_list)
```

LIST OF ALL FILES IN DIRECTORY AND SUB-DIRECTORIES

os.work() can also be used to get the list of files. This is the example to get the list of all txt files under 'C:\HOW_2022'.

```
name_list=[]
for (dir_path, dir_name, file_name) in os.walk('c:/HOW_2022'):
    for file in file_name:
        if file.endswith('.txt'):
            name_list.append(os.path.join(dir_path, file))
print(name_list)
```

DELETE EMPTY DIRECTORY

os.rmdir() method in Python is used to remove or delete a empty directory. OSError will be raised if the specified path is not an empty directory. This is the example code to remove empty directory with handling errors while using os.rmdir.

```
directory='C:/HOW_2022/Section 4'
try:
    os.rmdir(directory)
    print("Directory '%s' has been removed successfully" %directory)
except OSError as error:
    print(error)
    print("Directory '%s' can not be removed" %directory)
```

Note: To remove a non-empty directory in Python, the easiest way is to use the shutil module rmtree() function to delete all directory tree.

```
import shutil
shutil.rmtree(path)
```

DELETE FILES

os.remove() method in Python is used to remove or delete a file path. This method cannot remove or delete a directory. We usually use os.path.isfile() to check whether the file exists before it is deleted. The example code is in shown below.

```
Myfile=’c:/HOW_2022/Section 1/test.txt’
if os.path.isfile(myfile):
    os.remove(myfile)
```

If we need to delete all files in a directory, this is the example code to delete all txt files.

```
path='c:/HOW_2022/Section 1'
os.chdir(path)
for file in os.listdir(path):
    if file.endswith('.txt'):
        os.remove(file)
```

MS OFFICE DOCUMENTS HANDLING

There are many python packages for manipulating a Microsoft Office Document. Below is the list of some Python libraries for working with Microsoft office documents that you might need for your projects or management.

- **XLWINGS**

  xlwings is open source and free, comes preinstalled with Anaconda and WinPython, and works on Windows and macOS. Automate Excel via Python scripts or Jupyter notebooks, call Python from Excel via macros, and write user-defined functions.

- **XLSXWRITER**

  XlsxWriter is a Python module that can be used to write text, numbers, formulas and hyperlinks to multiple worksheets in an Excel 2007+ XLSX file. It supports features such as formatting and many more. XlsxWriter is in 100% compatible Excel XLSX files.

- **PYTHON-PPTX**

  python-pptx is a Python library for creating and updating PowerPoint (.pptx) files. A typical use would be generating a customized PowerPoint presentation from database content, downloadable by clicking a link in a web application.

- **PYTHON-DOCX**

  python-docx is a Python library for creating and updating Microsoft Word (.docx) files.

- **PYTHON-DOCX-TEMPLATE**

  python-docx-template has been created because python-docx is powerful for creating documents but not for modifying them. The idea is to begin to create an example of the document you want to generate with Microsoft word, it can be as complex as you want: pictures, index tables, footer, header, variables, anything you can do with word.

In the section 2 of HOW, we practice the documents handling focused on excel, word files. Some example codes are below. More instructions and codes are in the HOW lecture.

CREATE MULTIPLE XLSX FILES

```
import os
import xlwings as xw
os.chdir('c:/HOW_2022/Section 2')
app=xw.App(visible=True, add_book=False)
for i in range(6):
    workbook=app.books.add()
    workbook.save(f'test{i}.xlsx')
```

RENAME ALL SPREADSHEETS IN A XLSX FILE

```
app=xw.App(visible=False, add_book=False)
workbook=app.books.open('test0.xlsx')
worksheets=workbook.sheets
for i in range(len(worksheets)):
    worksheets[i].name=worksheets[i].name.replace('Sheet', 'Group')
workbook.save('test0.xlsx')
app.quit()
```

GENERATE A DOCX FILES

```
from docx import Document
from docx.shared import Inches
document = Document()
document.add_heading('Document Title', 0)
p = document.add_paragraph('A plain paragraph having some ')
p.add_run('bold').bold = True
p.add_run(' and some ')
p.add_run('italic.').italic = True
document.add_heading('Heading, level 1', level=1)
document.add_paragraph('Intense quote', style='Intense Quote')
document.add_paragraph(
    'first item in unordered list', style='List Bullet'
)
document.add_paragraph(
    'first item in ordered list', style='List Number'
)
records = (
    (3, '101', 'Spam'),
    (7, '422', 'Eggs'),
    (4, '631', 'Spam, spam, eggs, and spam')
)
table = document.add_table(rows=1, cols=3)
hdr_cells = table.rows[0].cells
hdr_cells[0].text = 'Qty'
hdr_cells[1].text = 'Id'
hdr_cells[2].text = 'Desc'
for qty, id, desc in records:
    row_cells = table.add_row().cells
    row_cells[0].text = str(qty)
    row_cells[1].text = id
    row_cells[2].text = desc
document.add_page_break()
document.save('demo.docx')
```

DATA PROCESSING

Data Processing is one the most important work in pharmaceutical industry. Usually, SAS or R is our preference, but Python also has the capability to meet our data analytic requirement. What's more. on the strength of comprehensive standard library, Python can provide various API for different types of files. We can access to almost all usual files like MS Word, MS EXCEL, SAS7BDAT and PDF by using different Python packages.

pandas is the most popular open source data analysis and manipulation tool built on Python. In the below example we will use pandas to read SAS file, compare 2 datasets and output the discrepancy.

Import Pandas:

```
import pandas as pd
```

READ SAS FILE

First, we need to read the data before any data analysis. Pandas has already provided API to access to SAS7BDAT file.

In the below codes, SAS file ae1.sas7bdat has been read and converted to a DataFrame which is a data structure created by pandas.

```
sas_file = r'C:\HOW_2022\Section 3\ae1.sas7bdat'
df = pd.read_sas(sas_file, encoding='Latin-1')
sas_file2= r'C:\HOW_2022\Section 3\ae2.sas7bdat'
df2= pd.read_sas(sas_file2, encoding='Latin-1')
```

We can also use some other packages to read SAS7BDAT file. Some of them might be even faster than pandas.

Package: pyreadstat

```
import pyreadstat
df, df_meta = pyreadstat.read_sas7bdat(sas_file, user_missing=True)
df
```

Package: sas7bdat

```
from sas7bdat import SAS7BDAT
with SAS7BDAT(sas_file, skip_header=False) as reader:
    df = reader.to_data_frame()
```

DATA ANALYSIS

Like SAS, combining datasets is also one of operations in pandas. We can merge 2 datasets on key variables and compare them to locate the variables with different values.

```
key_vars = ['Subject', 'RecordId']
comp = df.join(df2.set_index(key_vars), on=key_vars, rsuffix="_old")
```

Save the discrepancy to a dictionary

```
change_dic = {}
for column in df.columns:
    if column in df2.columns and column not in ['Subject', 'RecordId']:
        c_rows = comp[column] == comp['{}_old'.format(column)]
        change_dic[column]=[n for n, i in enumerate(c_rows) if i== False]
```

OUTPUT

Pandas can send the result to Excel directly and save as a XLSX file. But it doesn't support many features from EXCEL to allow user to customize it. So let's use xlwings to create the output.

```
import xlwings as xw
import os

report_path = r'C:\HOW_2022\Section 3'

app=xw.App(visible=False,add_book=False)
workbook=app.books.add()
workbook.sheets[0].name = 'AE'
worksheet = workbook.sheets['AE']

worksheet['A1'].value = df
worksheet.autofit()

for columnn, column in enumerate(df.columns):
    if column in change_dic.keys():
        for rown in change_dic[column]:
            worksheet.range((rown+2, columnn+2)).color = "#FFFF00"

workbook.save(path=os.path.join(report_path, 'AE.xlsx'))
workbook.close()
app.kill()
```

An excel file with different values highlighted in yellow will be created.

UI (USER INTERFACE) DEVELOPMENT

We can also use Python to create an application with a UI which is more user friendly. In the below section, a simple UI to convert SAS7BDAT to EXCEL will be created by using tkinter which is the standard Python interface to the Tk GUI toolkit.

CREATE UI

A simple UI can be created by using only serval lines of codes

```
from tkinter import *
root = Tk()
root.geometry("650x420+200+300")
root.title("Convert")
Button(root, text='Open').pack()
root.mainloop()
```

A widget Button has been created in the above code while it doesn’t response to any event. We need to create a function to bind to it.

BIND TO EVENT

Create a simple function which ask to select a file from our file location

```
from tkinter.filedialog import askopenfilenames
def open_f():
    openf=askopenfilenames(title="Open", filetypes=[("SAS", ".sas7bdat")])
```

Bind this function to Button

```
Button(root, text='Open', command=open_f).pack()
```

A SIMPLE APPLICATION

Prepare the required functions

Read SAS7BDAT

```
import pyreadstat
def read_file(infile, label=False):
    if infile.endswith('.xpt'):
        file_new, meta = pyreadstat.read_xport(infile)
    else:
        file_new, meta = pyreadstat.read_sas7bdat(infile)
    if label == True:
        file_new.columns = meta.column_labels
    return file_new
```

Output EXCEL

```
import xlwings as xw
def to_excel(dataf, filepath, outfile_excel, sheetname):
    app=xw.App(visible=False,add_book=False)
    workbook=app.books.add()
    app2 = workbook.app
    workbook.sheets[0].name = 'This is a temporary sheet'
    xw.sheets.add(sheetname)
    worksheet = workbook.sheets[sheetname]
    worksheet['A1'].options(chunksize=10000).value = dataf
    worksheet['A1:A'+str(len(dataf)+1)].delete('left')
    worksheet.autofit('c')
    workbook.sheets['This is a temporary sheet'].delete()
    workbook.save(path=os.path.join(filepath, outfile_excel))
    workbook.close()
    app2.kill()Conclusion
```

Intergrade into UI

```
im from tkinter import *
```

```
from tkinter import messagebox
from tkinter.filedialog import askopenfilenames

class AppIndex_CV(Frame):

    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.pack()
        self.createwiget()

    def createwiget(self):
        Button(self, text='Open', width=12,
command=self.open_f).grid(row=0, column=0, padx=10, pady=5, sticky=E)

        self.checkvar2 = BooleanVar()
        self.checkvar2.set(True)
        c2 = Checkbutton(self, text = "Display Label", variable =
self.checkvar2, onvalue = True, offvalue = False, width = 10)
        c2.grid(row=0, column=4, padx=0, sticky=W)

        self.text1 = Text(self, width=55, height=20, bg="white",
foreground="black")
        self.text1.grid(row=1, column=1, columnspan=2, padx=0, pady=10,
sticky=EW)

        Button(self, text='Convert', width=12,
command=self.convert_f).grid(row=2, column=0, padx=10, pady=5, sticky=E)

    def open_f(self):
        self.openf = askopenfilenames(title="Upload file", filetypes=[("SAS
file", ".sas7bdat"), ("SAS file", ".xpt")])

        self.text1.delete(1.0,END)
        for filepath in self.openf:
            self.text1.insert(INSERT, filepath+'\n')

    def convert_f(self):
        allfiles = []
        try:
            for file in self.openf:
                file_new = read_file(file, self.checkvar2.get())

                filepath = os.path.dirname(file)
                filename = os.path.basename(file)
                sheetname = '.'.join(filename.split('.')[:-1])

                to_excel(file_new, filepath, sheetname+'.xlsx', sheetname)

            messagebox.showinfo('Sucess', 'File convert to EXCEL')
        except BaseException as e:
            print(e)
            messagebox.showerror('Error', 'Failed')

if __name__ == '__main__':
    root = Tk()
    root.geometry("650x420+200+300")
root.title("Convert")

AppIndex_CV(master=root)

root.mainloop()
```

CONCLUSION

According to the TIOBE index, which measures the popularity of programming languages, Python is the most popular programming language in the world. There are many reasons for the ubiquity of Python, including its ease of use, simple syntax, thriving community and versatility.

This paper is just a part of work what Python programming can be used in pharmaceutical industry which is limited by our knowledge. Python can do much more than what have introduced. If these examples can have you be interested in Python then that is our goal.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Jie Liu
Enterprise: BeiGene (Beijing) Co., Ltd.
E-mail: jie1.liu@beigene.com

Name: Wei (Tony) Zhang
Enterprise: Pfizer (China) Research and Development Co.,Ltd. China
E-mail: wei.zhang18@pfizer.com