# Sprint 2

Bruno Augusto

10/05/2017

## Resumo

Neste relatório se encontrará a ideia base transmitida durante o capítulo 12 (Working with Excel spreadsheets) do livro em estudo, com exemplos dos programas de maior relevância e conceitos base para entendimento do assunto abordado.

# 1 Reading Data from a Spreadsheet

Este programa é usado para contar o numero total da população e de censos em um município. O objetivo é utilizar os comandos básicos aprendidos até o momento: abrir um arquivo e acessar as suas páginas e células.

```
import openpyxl, pprint #Importa o modulo

wb = openpyxl.load_workbook('censuspopdata.xlsx')     #Abre o arquivo
sheet = wb.get_sheet_by_name('Population by Census Tract')        #Abre a pagina pelo nome
countyData = {}

for row in range(2, sheet.max_row() + 1):      #Fixa o valor nas linhas, começando da 2ª e indo até a maior + 1
    state = sheet['B' + str(row)].value      #Pega o valor na celula Brow
    county = sheet['C' + str(row)].value     #Pega o valor na celula Crow
    pop = sheet['D' + str(row)].value        #Pega o valor na celula Drow
    countyData.setdefault(state, {})         #Confere se o nome do estado existe no dicionario
    countyData[state].setdefault(county, {'tracts': 0, 'pop': 0})     #Confere se existe a chave para esse municipio existe
    countyData[state][county]['tracts'] += 1 #Pra contar quantas linhas tem o municipio
    countyData[state][county]['pop'] += int(pop)    #Pra somar a população no municipio

resultFile = open('census2010.py', 'w')
resultFile.write('allData = ' + pprint.pformat(countyData))  #Abre um novo arquivo e escreve os resultados do countyData lá
resultFile.close()
print('Done.')
```

# 2 Updating a Spreadsheet

Neste programa foram usados os conhecimentos adquiridos até então para atualizar um arquivo já existente. O programa deve percorrer todo o arquivo, encontrar o produto cujo preço está errado e atualizá-lo.

```
import openpyxl

wb = openpyxl.load_workbook('produceSales.xlsx')   #Abre o arquivo
sheet = wb.get_sheet_by_name('Sheet')       #Abre a pagina pelo nome

PRICE_UPDATES = {'Garlic': 3.07, 'Celery': 1.19, 'Lemon': 1.27}    #Dicionario com as coisas pra mudar

for rowNum in range(2, sheet.max_row()): #Laço que vai da segunda linha até a ultima
    produceName = sheet.cell(row=rowNum, column=1).value
    if produceName in PRICE_UPDATES:
        sheet.cell(row=rowNum, column=2).value = PRICE_UPDATES[produceName]
#O pega nome por nome e compara com as keys do dicionario, caso de True ele muda o preço

wb.save('updatedProduceSales.xlsx')        #Salva em um novo arquivo
```

# 3 Multiplication Table Maker

Programa que recebe um número N informado pelo usuário e cria uma tabela NxN. Note que neste programa são utilizados praticamente todos os comandos aprendidos neste capítulo, desde criar um novo arquivo até a estilização de células.

```
import openpyxl
from openpyxl.styles import Font

def preencher(sheet, num):
    fonte = Font(bold=True)  #Define um estilo para a fonte
    for i in range(2, (num+2)):
        sheet.cell(row=1, column=i).value = int(i)-1
        sheet.cell(row=i, column=1).value = int(i)-1
        sheet.cell(row=1, column=i).font = fonte   #Define o estilo da fonte na célula
        sheet.cell(row=i, column=1).font = fonte

def multiplicar(sheet, num):
    for i in range(2,(num+2)):
        for j in range(2,(num+2)):
            sheet.cell(row=i, column=j).value = (i-1)*(j-1)

wb = openpyxl.Workbook()
sheet = wb.get_active_sheet()
print('Digite um numero:')
N = int(input())
preencher(sheet,N)
multiplicar(sheet,N)
wb.save('teste.xlsx')
```

# 4 Principais comandos

Os métodos e funções utilizados pertencem ao módulo *openpyxl*. Estes são os principais que foram aprendidos neste capítulo. OBS.: Note que algumas funções que aparecem no livro não são mais utilizadas na versão mais recente do *openpyxl*.

- load_workbook('exemplo.xlsx') - abre um arquivo de formato xlsx
- get_sheet_by_name('nome da pagina') - abre a pagina do seu arquivo
- get_active_sheet() - abre a primeira pagina do seu arquivo
- cell(row=i, column=j).value - mostra o valor armazenado na célula
- max_row() - mostra o numero total de linhas usadas na pagina
- max_column() - mostra o numero total de colunas usadas na pagina
- get_column_letter(numero) - retorna a letra respectiva ao numero
- column_index_from_string('letra') - retorna o numero respectivo a letra
- Workbook() – cria um arquivo
- save() – salva o arquivo