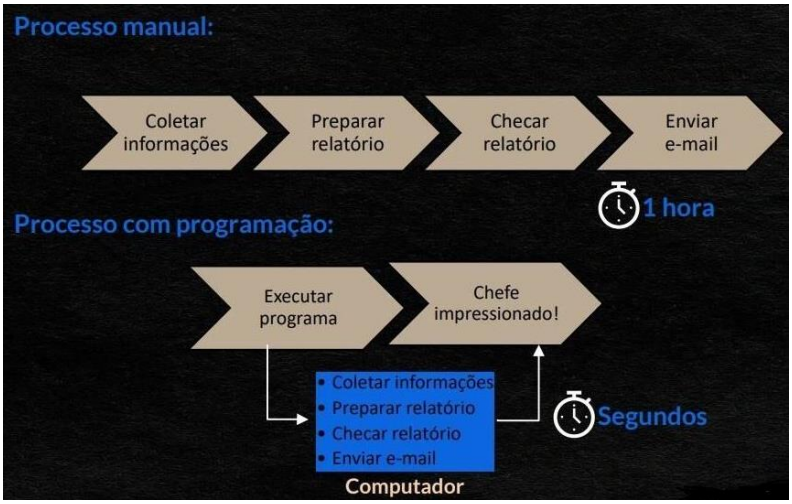
O que é Programação e o que é Python

Conjunto de comandos escritos em uma linguagem específica na qual o computador consegue compreender. Vamos considerar um exemplo simples. Todo dia você prepara um relatório para o seu chefe com a atualização da produção do dia anterior. Ele sempre elogia seu trabalho, mas gostaria de receber no primeiro horário. Manualmente, seu trabalho demora pelo menos 1 hora... Com programação, você pode descrever o que você faz para o computador, e assim, fazer com que ele faça para você em segundos. Deixando seu chefe ainda mais impressionado!!



O que é o Python? Por que o Python? Uma das linguagens mais utilizadas no mundo. Comunidade global de usuários do Python. Muitos “problemas” já resolvidos. Isso ajuda muito, pois você não precisa quebrar a cabeça de como fazer e sim buscar um código que resolva o seu “problema”, e personalizar para o seu caso. Links e Comunidades que valem a pena conhecer:

- Stack Overflow: https://stackoverflow.com/ ;
- Python Documentação: https://www.python.org/ ;
- Github: GitHub: Let’s build from here · GitHub;

Vamos começar do começo!

Python é uma linguagem dinamicamente tipada, o que significa que não é necessário declarar o tipo de variável ou fazer casting (mudar o tipo de variável), pois o Interpretador se encarrega disso para nós! Isso significa também que o tipo da variável poder variar durante a execução do programa. Os tipos de dados padrão do Python são:

Inteiro (int)
Ponto Flutuante ou Decimal (float)
Tipo Complexo (complex)
String (str)
Boolean (bool)
List (list)
Tuple
Dictionary (dic)

Tipo Inteiro (int)

O tipo inteiro é um tipo composto por caracteres numéricos (algarismos) inteiros. É um tipo usado para um número que pode ser escrito sem um componente decimal, podendo ter ou não sinal, isto é: ser positivo ou negativo.

Por exemplo, 21, 4, 0, e −2048 são números inteiros, enquanto 9.75, 1/2, 1.5 não são.

Exemplos:

```
1  idade = 18
2  ano = 2002
3
4  print(type(idade))
5  print(type(ano))
```

Saída:

```
Resultado

<class 'int'>
<class 'int'>
```

Ponto Flutuante ou Decimal (float)

É um tipo composto por caracteres numéricos (algarismo) decimais. O famoso ponto flutuante é um tipo usado para números racionais (números que podem ser representados por uma fração) informalmente conhecido como “número quebrado”.

Exemplos:

```
1  altura = 1.80
2  peso = 73.55
3
4  print(type(peso))
5  print(type(altura))
```

Saída:

```
Resultado

<class 'float'>
<class 'float'>
```

Complexo (complex)

Tipo de dado usado para representar números complexos (isso mesmo, aquilo que provavelmente estudou no terceiro ano do ensino médio). Esse tipo normalmente é usado em cálculos geométricos e científicos. Um tipo complexo contém duas partes: a parte real e a parte imaginária, sendo que a parte imaginária contém um j no sufixo. A função complex(real[, imag]) do Python possibilita a criação de números imaginários passando como argumento: real, que é a parte Real do número complexo e o argumento opcional imag, representando a parte imaginária do número complexo.

Exemplo:

```python
a = 5+2j
b = 20+6j

print(type(a))
print(type(b))
print(complex(2, 5))
```

Saída:

```
Resultado

<class 'complex'>
<class 'complex'>
(2+5j)
```

String (str)

É um conjunto de caracteres dispostos numa determinada ordem, geralmente utilizada para representar palavras, frases ou textos.

Exemplos:

```python
nome = 'Guilherme'
profissao = 'Engenheiro de Software'

print(type(profissao))
print(type(nome))
```

Saída:

```
Resultado

<class 'str'>
<class 'str'>
```

Boolean (bool)

Tipo de dado lógico que pode assumir apenas dois valores: falso ou verdadeiro (False ou True em Python). Na lógica computacional, podem ser considerados como 0 ou 1.

Exemplos:

```python
fim_de_semana = True
feriado = False

print(type(fim_de_semana))
print(type(feriado))
```

Saída:

```
Resultado

<class 'bool'>
<class 'bool'>
```

Listas (list)

Tipo de dado muito importante e que é muito utilizado no dia a dia do desenvolvedor Python! Listas agrupam um conjunto de elementos variados, podendo conter: inteiros, floats, strings, outras listas e outros tipos. Elas são definidas utilizando-se colchetes para delimitar a lista e vírgulas para separar os elementos, veja alguns exemplo abaixo:

```
1  alunos = ['Amanda', 'Ana', 'Bruno', 'João']
2  notas = [10, 8.5, 7.8, 8.0]
3
4  print(type(alunos))
5  print(type(notas))
```

Saída:

```
Resultado

<class 'list'>
<class 'list'>
```

Tuplas (tuple)

Assim como Lista, Tupla é um tipo que agrupa um conjunto de elementos. Porém sua forma de definição é diferente: utilizamos parênteses e separado por vírgula. A diferença para Lista é que Tuplas são imutáveis, ou seja, após sua definição, Tuplas não podem ser modificadas.

Exemplos:

```
1  valores = (90, 79, 54, 32, 21)
2  pontos = (100, 94.05, 86.8, 62)
3
4  print(type(valores))
5  print(type(pontos))
```

Saída:

```
Resultado

<class 'tuple'>
<class 'tuple'>
```

Caso haja uma tentativa de alterar os itens de uma tupla após sua definição, como no código a seguir:

```
1  tuple = (0, 1, 2, 3)
2  tuple[0] = 4
```

O seguinte erro do tipo TypeError será lançado pelo interpretador do Python:

```
Saída do Terminal com Erro

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Dicionários (dict)

Dict é um tipo de dado muito flexível do Python.Eles são utilizados para agrupar elementos através da estrutura de chave e valor, onde a chave é o primeiro elemento seguido por dois pontos e pelo valor.

Exemplo:

```
1  altura = {'Amanda': 1.65, 'Ana': 1.60, 'João': 1.70}
2  peso = {'Amanda': 60, 'Ana': 58, 'João': 68}
3
4  print(type(altura))
5  print(type(peso))
```

Saída:

```
Resultado

<class 'dict'>
<class 'dict'>
```

Conhecendo a IDE Visual Studio Code

Em 2015 foi lançado pela Microsoft um editor de código destinado ao desenvolvimento de aplicações web chamado de Visual Studio Code, ou simplesmente VSCode. Anunciada durante o Build, evento voltado a desenvolvedores que ocorre nos Estados Unidos anualmente, trata-se de uma ferramenta leve e multiplataforma que está disponível tanto para Windows, quanto para Mac OS e Linux e atende a uma gama enorme de projetos, não apenas ASP.NET, como também Node.js. Adicionalmente, o editor possui suporte à sintaxe de diversas linguagens como Python, Ruby, C++.

Além de ser totalmente gratuito, ainda no segundo semestre do ano do lançamento, durante o evento Connect(), o editor foi anunciado como open source, tendo código disponibilizado no GitHub, o que permite à comunidade técnica contribuir com seu desenvolvimento e facilitando a criação de extensões e novas funcionalidades.

Ao fim veremos como realizar o deploy da aplicação no Microsoft Azure, completando todo um ciclo básico de desenvolvimento de uma aplicação web, indo desde sua criação, configuração, até sua implantação em um servidor externo.

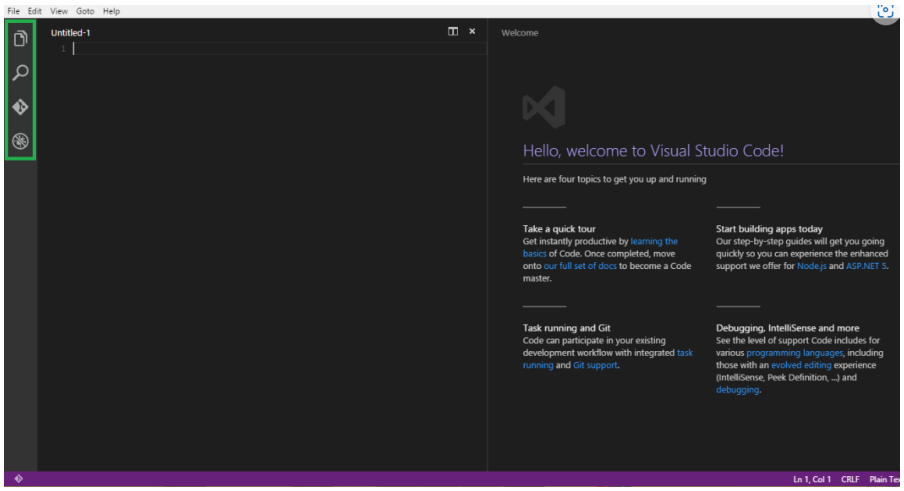
Instalando o Visual Studio Code

Como a instalação do VSCode já ocorre por intermédio do próprio Grupo Boticário, pois é uma ferramenta homologada e que a instalação é responsabilidade técnica da área responsável por disponibilizar esse acesso, não discorremos dessa ponto.

Primeiro contato com o editor

Diferentemente do que o nome pode sugerir em um primeiro momento, o VSCode não é uma versão do Visual Studio, em sua essência ele é um editor de código semelhante ao Sublime Text, Brackets e Atom, com funcionalidades otimizadas para certas tarefas. Na Figura 1 podemos ver o ambiente principal da ferramenta, com alguns elementos do layout destacados.

À esquerda temos uma barra de ferramenta com as opções (de cima para baixo): Explore, que exibe ou oculta os arquivos abertos; Search, para efetuar buscas nos arquivos abertos; Git, que facilita a realização de commits para o repositório do projeto; e Debug, onde é possível analisar variáveis em tempo de execução. Do lado direito temos uma página de boas-vindas onde encontramos links úteis sobre o VSCode, como tutoriais e extensões.



Projetos e arquivos

Com o VSCode é possível abrir tanto um único arquivo como uma pasta completa e pode utilizar os arquivos abertos como base para oferecer opções no IntelliSense dinamicamente, como é o caso dos arquivos JSON de configuração dos projetos ASP.NET 5. Com um layout simplificado e intuitivo, o objetivo é maximizar a área do editor, buscando deixar mais espaço para a navegação e acesso completo ao contexto da pasta ou do projeto. A interface de usuário (UI) é dividida em quatro partes principais:

- Editor: área principal para a edição dos arquivos, onde podemos abrir até três editores lado a lado; Side Bar, cujo conteúdo varia de acordo com a ação a ser executada (explorar arquivos, interagir com o Git., etc.);

- Status Bar: indica as informações sobre o projeto aberto e os arquivos que são editados; View Bar, onde temos a possibilidade de alternar entre as views do projeto e ainda ter os indicadores de contexto, como o número de alterações realizadas, caso o Git esteja habilitado. Um recurso interessante no editor é que ele preserva o estado dos arquivos, layouts e pastas no momento que ele é fechado, o que quer dizer que quando o editor for reaberto, todos os itens serão restaurados na forma como estavam anteriormente.

Editores lado a lado

Com a possibilidade de se ter até três editores abertos lado a lado, podemos abri-los das seguintes formas:

- Manter a tecla Ctrl pressionada e em seguida clicar em um arquivo no Explorer;
- Pressionar as teclas Ctrl + \, atalho que realiza a divisão do editor ativo em dois;
- Clicar com o botão direito do mouse sobre o arquivo no Explorer e em seguida na opção Open to the Side;
- Quando temos mais de um editor aberto, podemos alternar entre eles rapidamente, apenas pressionando Ctrl + 1, 2 ou 3.

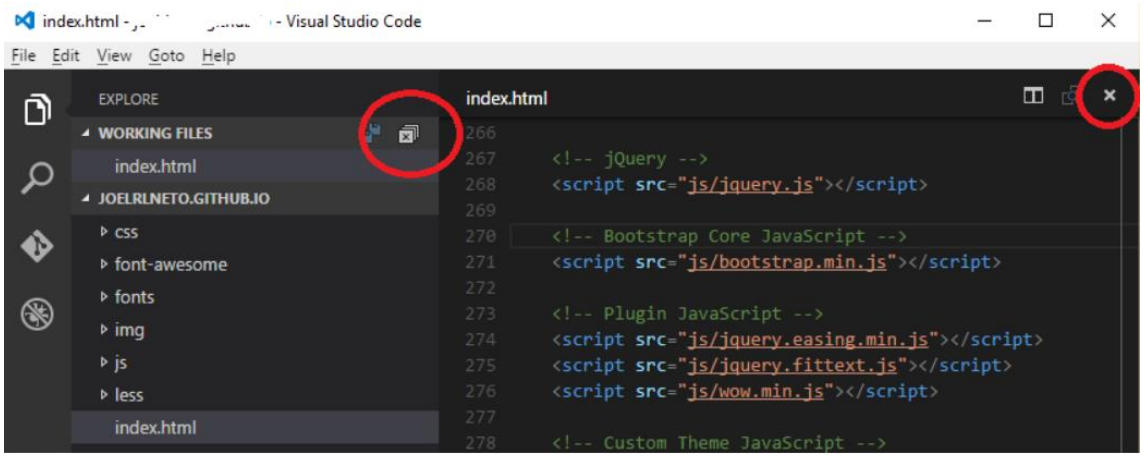
Explorer

O Explorer é usado para a navegação, abertura e gerenciamento de todos os arquivos e pastas do projeto. Após a abertura de uma pasta no VSCode, o conteúdo da pasta é mostrado no Explorer, onde podemos realizar operações de criar, remover ou renomear os arquivos, além de mover arquivos e pastas com a função de drag and drop.

Ao clicar com o botão direito do mouse sobre os arquivos e pastas no Explorer temos ainda algumas opções que merecem destaque, como a Open in Command Prompt, que abre o prompt de comandos do Windows no diretório do projeto; a Reveal in Explorer, que abre o arquivo no Windows Explorer; e a Copy Path, que copia o caminho do arquivo para a área de transferência. Para navegar entre os arquivos podemos utilizar os atalhos Ctrl + Tab ou Alt + Setas laterais.

Working Files

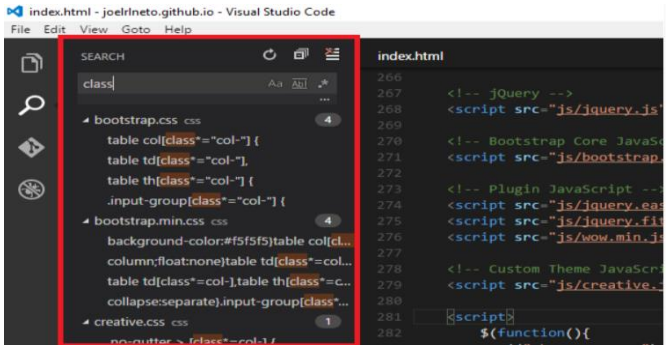
Ao visualizarmos o Explorer, temos a barra lateral dividida em seções, sendo a primeira delas a Working Files, que contém uma lista de arquivos ativos, ou seja, aqueles que foram abertos recentemente no VSCode. Como podemos ver na Figura 2, assim como em outros editores podemos fechar os arquivos individualmente (marcação à direita) ou todos da seção utilizando a ação Close All Files (marcação à esquerda).



Para salvar os arquivos, podemos utilizar o atalho Ctrl + S, comum em várias ferramentas, ou ainda o botão Save All, localizado ao lado do Close All. Além disso também podemos utilizar a opção Auto Save, com a qual as alterações realizadas serão salvas no decorrer do projeto sem que precisemos salvar manualmente. Para ativar ou desativar essa funcionalidade temos dois caminhos. O primeiro é abrir a Command Palette (a partir do menu View ou pressionando Ctrl_Shift+P) e digitarmos “auto”, o que listará a opção Files: Enable/Disable Auto Save, e pressionar Enter. A segunda forma é acessando o menu Filee clicar na opção Auto Save.

A partir do Explorer também é possível realizar pesquisas rápidas sobre todos os arquivos na pasta atualmente aberta. Para isso basta utilizarmos as teclas de atalho Ctrl + Shift + F ou acessar a opção Search na barra lateral que vimos na Figura 1. E em seguida, devemos digitar o que precisamos encontrar e pressionar Enter. Os resultados obtidos da pesquisa são então agrupados em arquivos contendo o termo de pesquisa, com a indicação de quantas vezes o termo está contido em cada arquivo e sua localização, como vemos na Figura 3.

Além das opções simples de pesquisa, podemos utilizar a busca por palavras completas ou ainda utilizar expressões regulares, opções que estão disponíveis na barra de digitação do termo buscado, como se pode ver na Figura 3.



Código utilizado no Projeto Case de introdução – GB

Link para os códigos na íntegra: https://github.com/AlexandreRego/C-digos-e-Projetos.git

Estarei disponibilizando aqui o código utilizado no case que serviu como uma introdução de solução ao projeto de intervenção. Os estudantes serão orientados a consultar o código original e a explicação, juntamente com o material de conhecimentos básicos no intuito de se familiarizar com as ferramentas e o propósito de casa uma.

Código que realiza a renomeação dos arquivos das notas fiscais para o formato adequado:

Importação de Bibliotecas:

Aqui, são importadas bibliotecas necessárias para a interface gráfica (Tkinter), manipulação de arquivos (shutil, os, pathlib), processamento de Notas Fiscais em formato PDF (PyPDF2), e expressões regulares (re).

```python
import tkinter as tk
from tkinter import filedialog
import shutil
import PyPDF2
import pathlib
import os
import re
```

Variáveis Globais:

Duas variáveis globais são definidas para armazenar informações sobre o número de páginas em uma Nota Fiscal e se os arquivos foram renomeados de acordo com uma enumeração.

```python
numero_de_paginas = 0
arquivos_renomeados_enumerados = False
```

Função para Contar Páginas da Nota Fiscal (PDF):

Utiliza a biblioteca PyPDF2 para contar o número de páginas em uma Nota Fiscal no formato PDF.

```python
def contar_paginas(file_path):
    # ...
```

Funções para Obter Número do Pedido a partir do Texto da Nota Fiscal:

'obter_numero_pedido' utiliza expressões regulares para extrair um número de pedido a partir do texto de uma Nota Fiscal em formato PDF. 'extract_info_from_pdf' extrai todo o texto de uma Nota Fiscal em PDF usando a biblioteca PyPDF2.

```python
def obter_numero_pedido(text):
    # ...

def extract_info_from_pdf(pdf_path):
    # ...
```

Funções para Renomear Arquivos de Notas Fiscais:

'renomear_arquivos_enumerados' e 'renomear_arquivos_de_acordo_com_NF' são funções que renomeiam os arquivos PDF (Notas Fiscais) com base em um índice (enumeração) e nas informações do número do pedido extraídas do texto da Nota Fiscal.

```python
def renomear_arquivos_enumerados(pdf_dir):
    # ...

def renomear_arquivos_de_acordo_com_NF(pdf_dir):
    # ...
```

Função para Lidar com o Upload de Arquivos:

Utiliza o Tkinter para abrir um diálogo de seleção de arquivos, copia os arquivos de Notas Fiscais para o diretório do código, exibe uma mensagem de sucesso e conta o número de páginas nas Notas Fiscais.

```python
def fazer_upload():
    # ...
```

Função para Separar Páginas de Notas Fiscais (PDF):

Separa as páginas de Notas Fiscais em arquivos PDF individuais e exclui o arquivo original.

```python
def separar_paginas_pdf(diretorio_pdf):
    # ...
```

Configuração da Janela Principal:

Configuração da janela principal usando Tkinter, incluindo a criação de botões, rótulos e definições de cor.

```python
root = tk.Tk()
# ...
```

Loop Principal:

```python
root.mainloop()
```

Inicia o loop principal da interface gráfica. Em resumo, esse código Python oferece uma interface gráfica para realizar operações relacionadas a Notas Fiscais, como upload, contagem de páginas, separação de páginas em arquivos individuais e renomeação de arquivos de acordo com informações específicas extraídas do texto das Notas Fiscais.

Código que realiza a criação do Data Frame e transposição para uma planilha das informações dos arquivos das notas fiscais que precisam ser utilizadas no arquivo input.

Importação de Bibliotecas:

São importadas bibliotecas essenciais para manipulação de arquivos, expressões regulares, processamento de PDFs, manipulação de dados em Excel e criação de interfaces gráficas com Tkinter.

```python
import os
import re
import openpyxl
import pandas as pd
import PyPDF2
import tkinter as tk
from tkinter import ttk
from datetime import datetime, timedelta
```

Parte 1: Extração de informações de PDFs

A primeira parte do código contém a função 'extrair_informacoes_pdf(pdf_path)' que extrai informações específicas de um PDF, como data de emissão, valor total, e série. Isso é feito usando a biblioteca PyPDF2 e expressões regulares para encontrar padrões no texto extraído do PDF.

Parte 2: Processamento de arquivos PDF

O código então percorre uma pasta contendo arquivos PDF, extrai informações de cada arquivo usando a função mencionada acima e armazena essas informações em um DataFrame do pandas.

Parte 3: Criação ou carregamento de uma planilha Excel

A próxima parte verifica se uma planilha Excel já existe. Se não existir, cria um DataFrame vazio. Se existir, carrega o DataFrame da planilha.

Parte 4: Preenchimento do DataFrame com informações extraídas

O código percorre todos os arquivos PDF na pasta, extrai informações relevantes do título do arquivo e adiciona essas informações ao DataFrame criado ou carregado anteriormente.

Parte 5: Salvar o DataFrame em um arquivo Excel

Ao final, o DataFrame é salvo em um arquivo Excel.

Parte 6: Funções para processar informações de PDFs (segunda parte do código)

A segunda parte do código define uma função ‘calcular_vencimento’ para calcular a data de vencimento com base na data de emissão. Em seguida, há outra função chamada extract_info_from_pdf, que usa expressões regulares para extrair informações específicas (como data de emissão, número do pedido, valor total, etc.) de um PDF.

Parte 7: Loop para processar arquivos PDF e criar um novo DataFrame

O código, então, percorre novamente a pasta de arquivos PDF, extrai informações com a função extract_info_from_pdf, e armazena essas informações em listas.

Parte 8: Criação de um novo DataFrame

Um novo DataFrame é criado com base nas listas criadas na etapa anterior.

Parte 9: Atualização do DataFrame original com informações adicionais

O código carrega a planilha Excel existente e adiciona colunas adicionais (como 'Emissão', 'Pedido', etc.) com base nas informações extraídas dos PDFs.

Parte 10: Exibição da tabela

Finalmente, o código define uma função exibir_tabela que cria uma interface gráfica usando a biblioteca tkinter para exibir a tabela de dados.