Sistema de diretórios: entender como manipular diretórios lista-los utilizando o Python Menezes (2019).

Um diretório corrente em Python é considerado um ***path*** (caminho).

Trabalhar diretamente com arquivos no mesmo diretório quando o projeto é pequeno pode ser uma comodidade. Porém, normalmente um projeto começa com uma pequena ideia e vai aumentando de tamanho conforme o cliente vai percebendo o quanto o programa pode ajudá-lo, por isso é importante estar atendo ao fator escalabilidade.

- A partir da raiz do disco, crie um diretório chamado proj3001.
- Abra o *Visual Studio Code,* abra o path criado e após crie o arquivo principal.py.

Verifica o path – módulo os função **getcwd**()

```
import os
root_dir = os.getcwd()
print(f'\nO Path é: {root_dir} \n')
```

Em Python podemos utilizar o módulo **os** e a função **mkdir**("nome") para criar *path* e sub *path*. A partir do *Path*, usando o terminal do VSCode crie um sub *path* chamado clientes:

Cria os sub *path* **clientes** e **produtos** módulo **os** função **mkdir**()

```
import os
os.mkdir('clientes')
os.mkdir('produtos')
```

lista todos os path e arquivos relativos

```
import os
print(f'\nOs paths e arquivos são: {os.listdir(".")} \n')
```

Mostra somente todos os paths

```
import os

root_dir = os.getcwd()

print(f'\nDiretório inicial do projeto: {root_dir} \n')

def directory_list(radix):
    for radix, directories, _ in os.walk(radix):
        for directory in directories:
            print(f'\nDiretórios do projeto: {os.path.join(radix, directory)} \n')

directory_list(root_dir)
```

Mostra os paths em forma de àrvore

No Windows antes de rodar o script abaixo copie e cole no terminal do VSCode a seguinte linha:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

```
import os

root_dir = os.getcwd()
print(f'\n{root_dir}')

def build_tree(root_dir, prefix=""):
    files = os.listdir(root_dir)
    files.sort()  # ordena arquivos e diretórios

    for i, filename in enumerate(files):
        path = os.path.join(root_dir, filename)
        is_last = i == (len(files) - 1)

        if os.path.isdir(path): # Mostra o diretório
            print(prefix + "└── " + filename if is_last else prefix + "├── " + filename)
            # Se não for o último item, adicione uma linha vertical abaixo deste diretório
            extension = "    " if is_last else "│   "
            build_tree(path, prefix=prefix + extension)

        else: # Mostra o arquivo
            print(prefix + "└── " + filename if is_last else prefix + "├── " + filename)

build_tree(root_dir)
print()
```

Em Python podemos mudar o subdiretório de trabalho usando o módulo **os** e a função **chdir**(“nome”) para abri-lo.

Abrindo um sub path - módulo **os** função **chdir**()

```
import os
os.chdir('clientes')
print(f'\nO path é: {os.getcwd()} \n')
```

Note que o endereço do sub *path* **clientes** é relativo ao diretório proj3001, ou seja, o path **proj3001**é **pai** do sub *path* **clientes**. O que a função **chdir**() faz é mudar o subdiretório corrente permitindo o acesso aos arquivos que estão dentro dele mais facilmente.

Para retornar ao *path* pai ou outro *path* relativo ao diretório corrente (***path*** corrente) aplicamos a função **chdir**(‘.’)

retornando ao subdiretório pai - módulo os função **chdir('.')**

```
import os
os.chdir('..')
print(f'\nO path é: {os.getcwd()} \n')
```

Em Python podemos renomear um *path's* com a função **rename**() do módulo **os**.

Renomeando clientes para cli

```
import os
os.rename('clientes', 'cli')
print(f'\nO novo nome é: {os.listdir(".")}')
```

Para apagar um *path* aplique a função **rmdir**() do módulo **os**

Apagando **cli**

```
import os
os.rmdir('cli')
print(os.listdir('.'))
```

Uma função interessante do Python para manipular *path*, mas que deve ser utilizada com cautela, é a função **makedirs**() do módulo **os**. Essa função permite criar mais de um path de uma vez.

Cria os *path's* **teste** e o *path* relativo a teste chamado **test_cliente**

```
import os
os.makedirs('teste/test_client')

root_dir = os.getcwd()
print(f'\n{root_dir}')

def build_tree(root_dir, prefix=""):
    files = os.listdir(root_dir)
    files.sort()  # ordena arquivos e diretórios

    for i, filename in enumerate(files):
        path = os.path.join(root_dir, filename)
        is_last = i == (len(files) - 1)

        if os.path.isdir(path): # Mostra o diretório
            print(prefix + "└── " + filename if is_last else prefix + "├── " + filename)
            # Se não for o último item, adicione uma linha vertical abaixo deste diretório
            extension = "    " if is_last else "│   "
            build_tree(path, prefix=prefix + extension)

        else: # Mostra o arquivo
            print(prefix + "└── " + filename if is_last else prefix + "├── " + filename)

build_tree(root_dir)
print()
```

Apague o sub path test_client com o comando:
path_to_remove = caminho

Atenção: Se o sistema operacional for Windows, para referenciar os diretórios (*path's*), utiliza-se duas barras '****'.

Outra função importante do módulo **os** é a função **path**(). Ela serve obtermos o tamanho do subdiretório, data de criação, acesso e modificação de dados, ou verificar se a informação que estamos lidando é um subdiretório ou arquivo.

Verifica subdiretórios existentes módulo **os** função **path.isdir()**

```
import os
import os.path
for subdiretorio in os.listdir('.'):
    if os.path.isdir(subdiretorio):
        print(f'{subdiretorio}')
```

Verifica se um subdiretório existe módulo **os** função **path.exists()**

```
import os.path
if os.path.exists('cli'):
    print('O diretório existe')
else:
    print('O diretório não existe')
```

Identifica o sistema operacional - modo **os** função **name**() – Linux/IOS/ Windows

```
import os
print(os.name)
```

Identifica detalhes no sistema operacional – Linux/IOS

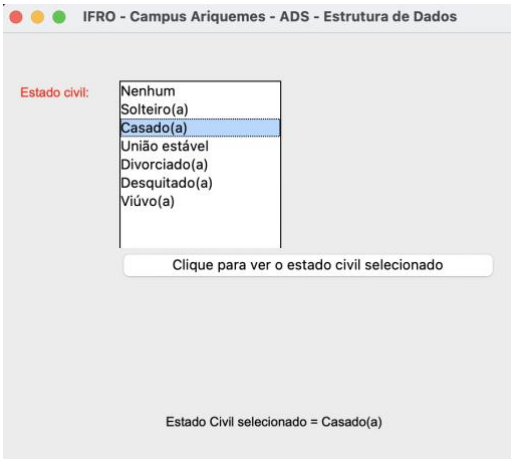
```
import os
print(os.uname())
```

- Identifica detalhes no sistema operacional Windows

```
import sys
sys.platform
```

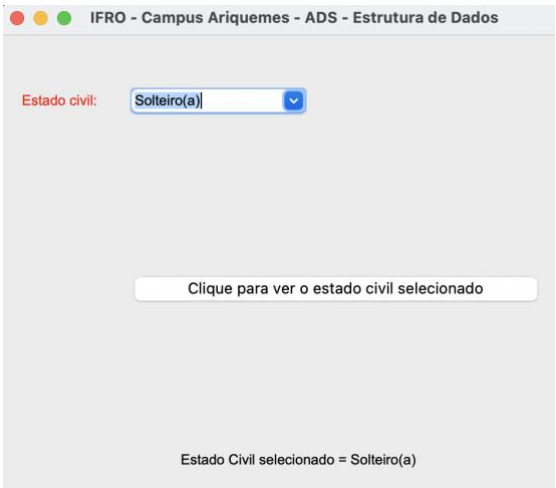
Tkinter Listbox()

```python
from tkinter import *
from tkinter import ttk

# Constantes para configuração
FONT = ('Arial', 12)
FG_COLOR = '#FA0404'

def imprimir_estado_civil():
    estado_civil = lb_es_civil.get(ACTIVE)
    # Atualiza o texto do rótulo com o estado civil selecionado
    lb_2.config(text=f'Estado Civil selecionado = {estado_civil}')

# Configuração da janela principal
janela = Tk()
janela.title('IFRO - Campus Ariquemes - ADS - Estrutura de Dados')
janela.geometry('500x500')

# Lista de estados civis
lista_estado_civil = ['Nenhum', 'Solteiro(a)', 'Casado(a)', 'União estável',
                      'Divorciado(a)', 'Desquitado(a)', 'Viúvo(a)']

# Rótulo para estado civil
Label(janela, text='Estado civil:', fg=FG_COLOR, font=FONT).place(x=15, y=45)

# Listbox para seleção do estado civil
lb_es_civil = Listbox(janela)
for estadocivil in lista_estado_civil:
    lb_es_civil.insert(END, estadocivil)
lb_es_civil.place(x=110, y=45, width=150)

# Botão para confirmar a seleção do estado civil
btn_estado_civil = Button(janela, text='Clique para ver o estado civil selecionado',
                          command=imprimir_estado_civil)
btn_estado_civil.place(x=110, y=200, width=350)

# Rótulo que mostra o estado civil selecionado, inicialmente vazio
lb_2 = Label(janela, text='', font=FONT)
lb_2.place(x=150, y=350)

# Inicia a interface
janela.mainloop()
```



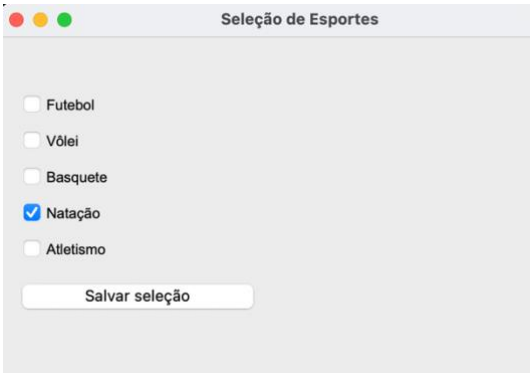
Tkinter Combobox()

```python
from tkinter import *
from tkinter import ttk

# Constantes para configuração
FONT = ('Arial', 12)
FG_COLOR = '#FA0404'

def imprimir_estado_civil():
    estado_civil = cb_es_civil.get()
    # Atualiza o texto do rótulo com o estado civil selecionado
    lb_2.config(text=f'Estado Civil selecionado = {estado_civil}')

# Configuração da janela principal
janela = Tk()
janela.title('IFRO - Campus Ariquemes - ADS - Estrutura de Dados')
janela.geometry('500x500')

# Lista de estados civis
lista_estado_civil = ['Nenhum', 'Solteiro(a)', 'Casado(a)',
                      'União estável', 'Divorciado(a)', 'Desquitado(a)', 'Viúvo(a)']

# Rótulo para estado civil
Label(janela, text='Estado civil:', fg=FG_COLOR, font=FONT).place(x=15, y=45)

# Combobox para seleção do estado civil
cb_es_civil = ttk.Combobox(janela, values=lista_estado_civil, font=FONT)
cb_es_civil.place(x=110, y=45, width=150)
cb_es_civil.set(lista_estado_civil[0])  # Define o valor padrão como o primeiro da lista

# Botão para confirmar a seleção do estado civil
btn_estado_civil = Button(janela, text='Clique para ver o estado civil selecionado', command=imprimir_estado_civil)
btn_estado_civil.place(x=110, y=200, width=350)

# Rótulo que mostra o estado civil selecionado, inicialmente vazio
lb_2 = Label(janela, text='', font=FONT)
lb_2.place(x=150, y=350)

# Inicia o loop principal da interface
janela.mainloop()
```



Tkinter Checkbox() com Json

```python
import json
from tkinter import *
from tkinter import messagebox

# Constantes para configuração
FONT = ('Arial', 12)
JSON_FILENAME = 'esportes_selecionados.json'

# Função para imprimir e salvar os esportes selecionados
def salvar_esportes_selecionados():
    esportes_selecionados = {esporte: var.get() for esporte, var in esportes_vars.items()}
    with open(JSON_FILENAME, 'w', encoding='utf-8') as json_file:
        json.dump(esportes_selecionados, json_file, ensure_ascii=False, indent=4)
    messagebox.showinfo('Informação', 'Esportes salvos com sucesso!')

# Configuração da janela principal
janela = Tk()
janela.title('Seleção de Esportes')
janela.geometry('500x500')

# Esportes disponíveis e suas respectivas variáveis de estado
esportes = ['Futebol', 'Vôlei', 'Basquete', 'Natação', 'Atletismo']
esportes_vars = {esporte: IntVar() for esporte in esportes}

# Carrega as seleções existentes do arquivo JSON
try:
    with open(JSON_FILENAME, 'r', encoding='utf-8') as json_file:
        esportes_selecionados_existente = json.load(json_file)
    # Atualiza as variáveis de estado com os valores existentes
    for esporte, var in esportes_vars.items():
        var.set(esportes_selecionados_existente.get(esporte, 0))
except FileNotFoundError:
    # Arquivo JSON não existe, então não há seleções prévias
    pass

# Rótulos e caixas de seleção para cada esporte
for i, (esporte, var) in enumerate(esportes_vars.items()):
    Checkbutton(janela, text=esporte, variable=var, font=FONT).place(x=15, y=45 + i*30)

# Botão para salvar a seleção dos esportes
btn_salvar = Button(janela, text='Salvar seleção', command=salvar_esportes_selecionados)
btn_salvar.place(x=15, y=200, width=200)

# Inicia o loop principal da interface
janela.mainloop()
```



Poste a atividade de 30-01-2025 até o dia 05-02-2025 às 18:30 – vale até 5 pontos.