Bootcamp: Desenvolvedor(a) Python

Desafio Prático

Módulo 1: Fundamentos de Desenvolvedor(a) Python

Objetivos de Ensino

Praticar e consolidar os seguintes conceitos da linguagem Python:

- ✓ Estruturas de dados (listas, tuplas, conjuntos e dicionários).
- ✓ Declaração, utilização e argumentos de funções.
- ✓ Criação, importação e instalação de módulos.
- ✓ Manipulação de arquivos em discos.
- ✓ Recursos úteis da linguagem.

Enunciado

Neste desafio o aluno deverá solucionar problemas simples utilizando a linguagem Python. Serão abordados conceitos como: estruturas de dados, funções e módulos. Após praticar, as questões objetivas devem ser respondidas, considerando os elementos apresentados nas atividades abaixo.

Não será necessária a entrega de nenhum código.

Atividades

Os alunos deverão desempenhar as seguintes atividades:

Atividade 01. Execute e analise a saída do seguinte código no Google Colab[1].

```
# relação dos nomes
nomes = ['Maria', 'Julieta', 'Fernando', 'Cristiano', 'Julieta', 'Maria', 'Fernando', 'Cláudio']

# estrutura que irá armazenar o número de letras de cada nome
qtd_letras = {}

# calcula o tamanho de cada nome (em número de letras) e armazena o valor na estrutura
for nome in nomes:
  qtd_letras[nome] = len(nome)
```

Atividade 02. Reescreva o código da Atividade 01 utilizando o conceito de compreensão de dicionários (*dict comprehension*).

Atividade 03. Crie uma função area(r, pi), que calcule a área de um círculo.

A área de um círculo é dada por $a = \pi r^2$, onde:

- π é o número pi, cujo valor é $\sim 3{,}14159265359$
- r é o raio do círculo

Esta função deve receber dois argumentos, r e pi, e retornar a área de um círculo com raio r. Entretanto, você deve fornecer duas opções de utilização da função:

a) A utilização de um valor padrão para pi ($\pi = 3{,}14$), sem a necessidade de declarar explicitamente o seu valor;

b) A utilização com a declaração explicita do valor exato de pi desejado.

Por exemplo, para um raio $r = 8$, o usuário poderá utilizar a sua função das seguintes maneiras:

[1] https://colab.research.google.com/

```
# Utilização da função sem declarar pi
# Neste caso o valor de pi deverá ser 3.14 e o resultado: 3.14 * (8 ** 2)
area(8)

# Utilização da função declarando explicitamente o valor de pi
# Neste caso o valor de pi deverá ser 3,141592 e o resultado: 3,141592 * (8 ** 2)
area(8, 3,141592)
```

DICA: Para resolver esse problema, utilize o conceito de argumentos padrões (*default*).

Atividade 04. Reescreva a função area da atividade 3, utilizando o conceito de funções anônimas (*funções lambdas*).

Atividade 05. Crie um módulo em Python, chamado processalista (arquivo processalista.py), com as seguintes funções, que deverão receber como argumento uma lista (tipo list) de números inteiros.

```
# Encontra e retorna o maior número impar presente na lista
def maior_impar(lista):
    # <seu codigo aqui>

# Encontra e retorna o menor número impar presente na lista
def menor_impar(lista):
    # <seu codigo aqui>

# Encontra e retorna o maior e o menor número ímpar presentes na lista
def menor_impar(lista):
    # <seu codigo aqui>
```

- Enquanto as duas primeiras funções devem retornar um único valor, a última função deverá retornar dois valores.

DICA: Não se esqueçam de reaproveitar as funções!

Atividade 06. Considere a seguinte agenda de disponibilidade de atendimento de uma clínica com vários médicos, cada um de uma especialidade distinta. Essa clínica possuí apenas três consultórios, e, por isso, no máximo três médicos podem atender por dia.

Especialidade	Segunda	Terça	Quarta	Quinta	Sexta
Cardiologista		✓	✓		
Ortopedista		✓		✓	
Dermatologista	✓		✓		✓
Neurologista		✓		✓	✓
Psiquiatra	✓		✓		✓

A clínica percebeu que os pacientes que desejam se consultar com mais de um médico sempre solicitam que o atendimento seja realizado no mesmo dia. Por exemplo, um paciente gostaria de se consultar, no mesmo dia, com o *ortopedista* e com o *neurologista*, então os dias disponíveis serão: *terça* e *quinta*. Outro exemplo é um paciente que gostaria de se consultar com o *dermatologista*, o *neurologista* e o *psiquiatra*. O único dia disponível será a *sexta*.

Entretanto, essas solicitações estão causando dor de cabeça, pois é necessário verificar manualmente a disponibilidade. Você, como um(a) aspirante a desenvolvedor Python, aceitou o desafio de ajudar a automatizar essa tarefa. Para tanto, irá desenvolver duas funções que receberão a relação de disponibilidade de cada médico e que retornarão os dias disponíveis:

```
# relação de dias da semana que cada médico atende
cardiologista = {'terca', 'quarta'}
ortopedista = {'terca', 'quinta'}
dermatologista = {'segunda', 'quarta', 'sexta'}
neurologista = {'terca', 'quinta', 'sexta'}
psiquiatra = {'segunda', 'quarta', 'sexta'}

# Calcula quais os dias possíveis para dois médicos
def disp_dois_especialistas(medico01, medico02):
    # <seu codigo aqui>

# Calcula quais os dias possíveis para três médicos
def disp_tres_especialistas(medico01, medico02, medico03):
    # <seu codigo aqui>
```

Assim, para os exemplos anteriores, as funções deverão ter os seguintes retornos:

```
# Médicos solicitados: ortopedista e neurologista
disp_dois_especialistas(ortopedista, neurologista)
# RETORNA: terça e quinta

# Médicos solicitados: dermatologista, neurologista, psquiatra
disp_tres_especialistas(dermatologista, neurologista, psiquiatra)
# RETORNA: sexta
```

DICA: Não se esqueçam das operações com conjuntos!