CURSO DE SISTEMAS PARA INTERNET

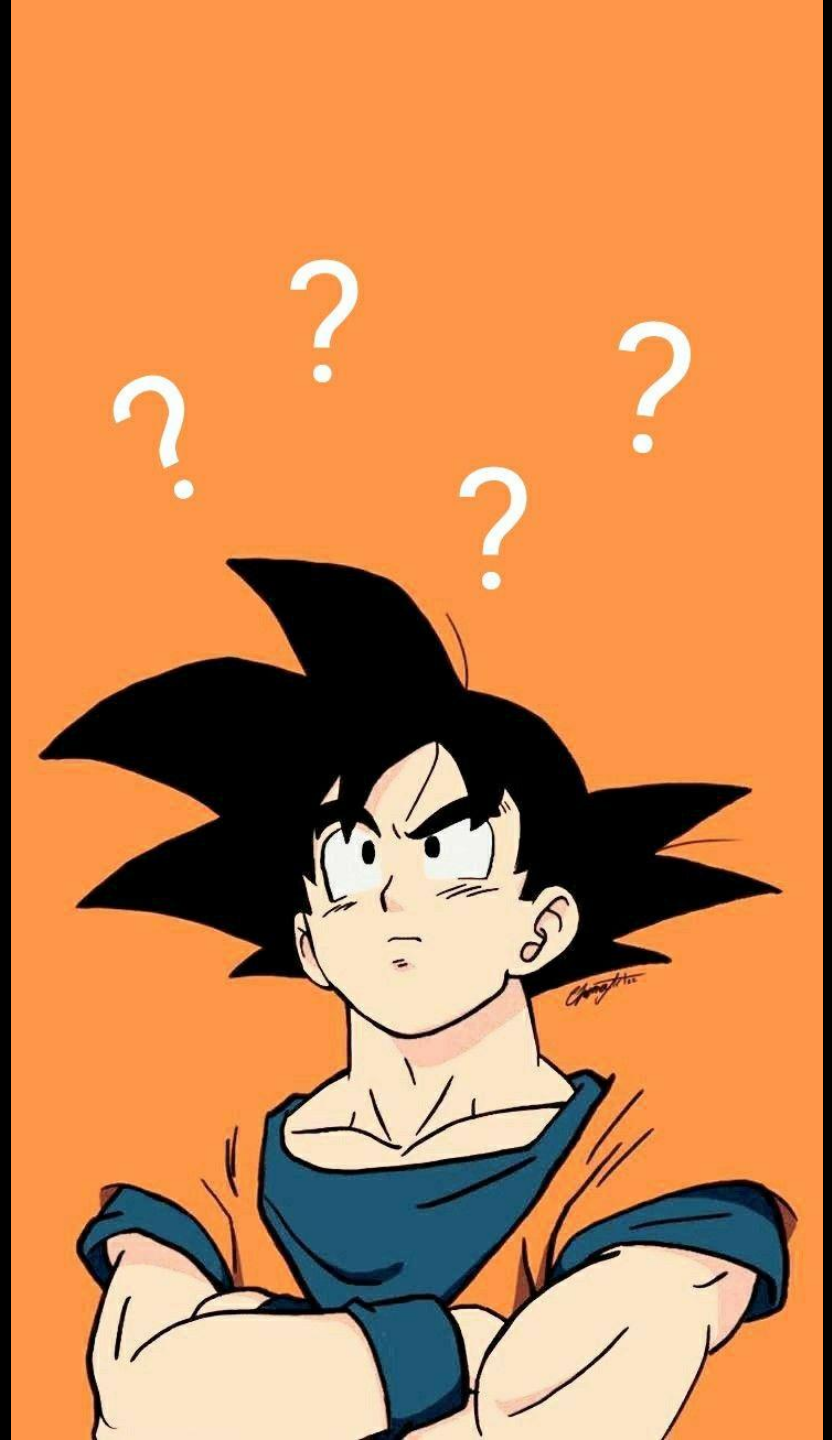
PROGRAMAÇÃO DE COMPUTADORES

TURMA: P1/P4

AULA 11

PROFESSOR: JEOFTON COSTA

Dúvidas aula anterior?



- **Funções.**

```python
print('Cadastro Comics\n')
print("--- * ---\nEscolha o tipo de personagem que deseja cadastrar:\n --- * ---")
heros = []
villians = []
type_character = 1
while type_character != 0:
    type_character = int(input('Digite:\n1 para cadastrar "Héroi";\n2 para cadastar "Vilão";\n0 para "Sair"\n'))
    if type_character == 1:
        type_name = "Héroi"
        name = input(f"Digite o nome do {type_name}:\n")
        alias = input(f"Digite o nome do {type_name}:\n")
        hero = f"Nome do {type_name}: {name}. Identidade Secreta do(a){name}: {alias}"
        heros.append(hero)
    elif type_character == 2:
        type_name = "Vilão"
        name = input(f"Digite o nome do {type_name}:\n")
        alias = input(f"Digite o nome do {type_name}:\n")
        villian = f"Nome do {type_name}: {name}. Identidade Secreta do(a){name}: {alias}"
        villians.append(villian)
    elif type_character == 0:
        print('cadastramento encerrado.\n')
    else:
        print('O número digitado não tem correspondência.\n')
        type_character = int(
            input('Digite uma destas opções:\n1 para cadastrar "Héroi";\n2 para cadastar "Vilão";\n0 para "Sair"'))
print(heros)
print(villians)

```

Observem este código, para cadastro de personagens.

```python
print('Cadastro Comics\n')
print("--- * ---\nEscolha o tipo de personagem que deseja cadastrar:\n --- * ---")
heros = []
villians = []
type_character = 1
while type_character != 0:
    type_character = int(input('Digite:\n1 para cadastrar "Héroi";\n2 para cadastar "Vilão";\n0 para "Sair"\n'))
    if type_character == 1:
        type_name = "Héroi"
        name = input(f"Digite o nome do {type_name}:\n")
        alias = input(f"Digite o nome do {type_name}:\n")
        hero = f"Nome do {type_name}: {name}. Identidade Secreta do(a){name}: {alias}"
        heros.append(hero)
    elif type_character == 2:
        type_name = "Vilão"
        name = input(f"Digite o nome do {type_name}:\n")
        alias = input(f"Digite o nome do {type_name}:\n")
        villian = f"Nome do {type_name}: {name}. Identidade Secreta do(a){name}: {alias}"
        villians.append(villian)
    elif type_character == 0:
        print('cadastramento encerrado.\n')
    else:
        print('O número digitado não tem correspondência.\n')
        type_character = int(
            input('Digite uma destas opções:\n1 para cadastrar "Héroi";\n2 para cadastar "Vilão";\n0 para "Sair"'))
print(heros)
print(villians)

```

Perceberam que existem trechos repetidos?

Não seria bacana que existisse algum modo de reduzirmos nossos códigos, eliminando repetições, ou que pudéssemos reutilizar o mesmo trecho de código sem precisar reescrevê-lo.

Isto existe, fedelho.

São blocos de códigos nomeados, concebidos para realizar uma tarefa específica, conhecidos como funções

```
1  def exibir_mensagem():
2      print("Esta mensagem está sendo gerada por minha função.")
3
4
5  exibir_mensagem()
```

Para implementar uma função usamos a palavra resevada **def**, depois estabelecemos o **nome da função**, aí devemos abrir e fechar parênteses **()**, depois colocamos os **:** e, por fim, indentamos o trecho de código que descreverá o **corpo da função**.

Veja a sintaxe python para funções.

```
1  def exibir_mensagem():
2      print("Esta mensagem está sendo gerada por minha função.")
3
4
5  exibir_mensagem()
6
7
```

Neste exemplo, temos a função na sua forma mais simples, ou seja, sem passagem de parâmetros/argumentos.

Aqui temos a chamada da função

```python
def exibir_mensagem(nome_usuario):
    print(f"{nome_usuario} veja, esta mensagem está sendo gerada por minha função.")



exibir_mensagem("Louis")
```

Este valor chamamos de parâmetros da função, que será um valor para uma variável exigida pela função

Neste outro caso, dentro dos parênteses estamos passando os parâmetros da função.

```python
def exibir_mensagem(nome_usuario):
    print(f"{nome_usuario}, veja! Esta mensagem está sendo gerada por minha função.")

nome = input("Quem está usando este programa? Digite o nome e tecle enter.\n")
exibir_mensagem(nome)
```

Na chamada da função nós passamos um ou mais argumentos

Aqui, o valor do argumento está sendo atribuído dinamicamente.

```
1 usage
def exibir_mensagem(nome_usuario):
    return f"{nome_usuario}, veja! Esta mensagem está sendo gerada por minha função."

nome = input("Quem está usando este programa? Digite o nome e tecle enter.\n")

mensagem = exibir_mensagem(nome)

print(mensagem)
```

Algumas funções podem ter valores de retorno. O retorno será devolvido ao código, na mesma linha na qual a função foi chamada.

A palavra **return**, quando encontrada dentro de uma função, possui comportamento de encerrar o processamento da função, retornando ao ponto do código onde houve o chamamento.

```python
def exibir_mensagem(nome_usuario):
    print(f"{nome_usuario}, veja! Esta mensagem está sendo gerada por minha função.\n")

repete = 's'

while repete == 's':
    nome = input("Quem está usando este programa? Digite o nome e tecle enter.\n")
    exibir_mensagem(nome)
    repete = input("Deseja repetir a pergunta? Se sim, tecle s. Caso contrário tecle qualquer letra\n").lower()
```

As funções podem ser chamadas em qualquer lugar do código.

```
1 usage
def soma(a, b):
    print(f"{a + b} foi o valor somado dentro da função.\n")

soma( a: 2, b: 5)
```

Quando precisamos passar mais de um argumento para uma função, os separamos por vírgula.

Do mesmo modo ocorrerá a recepção destes na definição das funções.

Eles são separados por vírgulas.

```python
1 usage
def recebe_sabores(*sabores):
    frases = []
    for sabor in sabores:
        frase = (f"{sabor}, Delícia de função!")
        frases.append(frase)
    return frases


for f in recebe_sabores( *sabores: 'Pepe', 'Mussa', 'Frango', 'Franpyri', 'Sertaneja'):
    print(f)

```

A passagem de parâmetros pode ser realizada de forma dinâmica. Veja esta sintaxe.

Existem diversas formas de passarmos parâmetros, e eles podem ser de todos os tipos de valores


Agradecido, Mestre. Preciso destes exercício para encarar o Vegeta.
Hora de praticar, moçada.

1. Escreva uma função chamada display_message() que mostre uma frase informando algo. Chame a função e certifique-se de que a mensagem seja exibida corretamente.
2. Escreva uma função chamada favorite_book() que aceite um parâmetro. A função deve exibir uma mensagem como: “Um dos meus livros favoritos é Alice no país das maravilhas”. Chame a função e não se esqueça de incluir o título do livro como argumento na chamada da função.
3. Escreva uma função chamada city_country() que aceite o nome de uma cidade e seu país. A função deve devolver uma string formatada assim: "Santiago, Chile" Chame sua função com pelo menos três pares cidade-país e apresente o valor devolvido.
4. Escreva uma função chamada make_shirt() que aceite um tamanho e o texto de uma mensagem que deverá ser estampada na camiseta. A função deve exibir uma frase que mostre o tamanho da camiseta e a mensagem estampada. Chame a função e verifique o resultado.
5. Modifique a função make_shirt() de modo que as camisetas sejam grandes, média ou pequena com uma mensagem qualquer, por exemplo: “Eu amo Python”. Crie várias camisetas de tamanhos diferentes, informando a frase o tamanho.
6. Escreva uma função chamada describe_city() que aceite o nome de uma cidade e seu país. A função deve exibir uma frase simples, como: “ Reykjavik está localizada na Islândia”. Chame sua função para três cidades diferentes em que pelo menos uma delas não esteja no país default.

1. Crie uma lista de nomes de mágicos. Passe a lista para uma função chamada show_magicians() que exiba o nome de cada mágico da lista.
2. Comece com uma cópia de seu programa do anterior. Escreva uma função chamada make_great() que modifique a lista de mágicos acrescentando a expressão o Grande ao nome de cada mágico. Chame show_magicians() para ver se a lista foi realmente modificada.
3. Comece com o trabalho feito no Exercício 2. Chame a função make_great() com uma cópia da lista de nomes de mágicos. Como a lista original não será alterada, devolva a nova lista e armazene-a em uma lista separada. Chame show_magicians() com cada lista para mostrar que você tem uma lista de nomes originais e uma lista com a expressão o Grande adicionada ao nome de cada mágico.

USO DE IMAGENS DE PERSONAGENS DE QUADRINHOS PARA FINS ESTRITAMENTE ACADÊMICOS

"As imagens de personagens de quadrinhos utilizadas neste material têm o propósito exclusivo de enriquecer o conteúdo acadêmico da apresentação. O uso está em conformidade com as leis de direitos autorais, pois se enquadra no contexto educacional. Respeitamos os direitos dos detentores dessas propriedades intelectuais.

As questões apresentadas neste material foram retiradas e adaptadas do livro: Curso intensivo de python uma introdução prática e baseada em projetos à programação, da editora novatec".