Aula 12

Comando pass

O “pass” é uma palavra-chave em Python que indica ao interpretador para não fazer nada. Ele é usado como um espaço reservado quando é necessário definir uma estrutura de controle de fluxo, mas ainda não há nenhum código a ser executado. Ele é normalmente usado como um placeholder para estruturas de controle de fluxo que ainda não foram implementadas.
Exemplos de uso :

```
num = int(input('Digite um número : '))

# ainda não sei o quero fazer com o if,
# mas já sei o que fazer com o else
if num == 0:
    pass
else:
    print('o que vier aqui não é o que quero')
```

No exemplo acima, ainda não sei o que fazer caso o número seja igual a zero. Então vou deixar o comando pass ali dentro para pensar mais tarde e ainda deixar meu código executável.

```
frase = input('Digite uma frase : ')

# convertendo a string em uma lista de palavras
lista = frase.split()

# agora vou passar pelas palavras, mas não
# sei o que fazer ainda
for item in lista:
    pass

print('Executando sem erros!')
```

No exemplo acima, recebemos uma frase do usuário e dividimos ela uma lista, mas ainda não sei exatamente o que fazer com ela... então vou deixar só implementado o início e depois penso no resto.

Praticamente, qualquer comando do Python que envolva alguma indentação pode receber um comando pass dentro e deixar a implementação para depois.

```
# vou receber um inteiro do usuário
try:
    numero = int(input('Digite um número : '))
except ValueError as ve:
    # como ainda não sei o que fazer se o usuário
    # digitar algo que não seja um número, vou deixar
    # a exceção vazia e pensar em algo depois
    pass

print('O usuário digitou um número, então segue o código')
```

No exemplo acima, estamos usando a condição try...except (não se preocupe com isso agora, iremos ver essas clausulas com muita calma mais adiante) para pegar um número inteiro do usuário. Se por acaso ele digitar algo que não seja um inteiro, temos que fazer outra coisa, mas depois pensaremos nisso.

Bubble Sort

O bubble sort, ou ordenação por flutuação (literalmente "por bolha"), é um algoritmo de ordenação dos mais simples. A ideia é percorrer o vetor diversas vezes, e a cada passagem fazer flutuar para o topo o maior elemento da sequência. Essa movimentação lembra a forma como as bolhas em um tanque de água procuram seu próprio nível, e disso vem o nome do algoritmo.

No melhor caso, o algoritmo executa ***n*** operações relevantes, onde ***n*** representa o número de elementos do vetor. No pior caso, são feitas n^2 operações. A complexidade desse algoritmo é de ordem quadrática. Por isso, ele não é recomendado para programas que precisem de velocidade e operem com quantidade elevada de dados.

Abaixo está o código que foi desenvolvido em sala de aula:

```python
# cria a lista que será ordenada
numeros = [6,5,3,1,8,7,2,4]
# tamanho da lista a ser ordenada
tam = len(numeros)

# enquanto o tam for maior que 0, continua ordenando
while 0 < tam:
    # passa pelos pares comparando
    for j in range(0, tam - 1):
        # verifica se o número da esquerda é maior
        # que o da direita
        if numeros[j] > numeros[j+1]:
            # se for maior, troca os números de posição
            aux = numeros[j]
            numeros[j] = numeros[j + 1]
            numeros[j + 1] = aux
    # como já jogamos o maior número para a posição mais
    # à direita, diminuimos o tamanho do vetor a ser
    # verificado
    tam -= 1
print(numeros)
```

Links para leitura e aprofundamento:

- https://pt.wikipedia.org/wiki/Bubble_sort
- https://rosettacode.org/wiki/Sorting_algorithms/Bubble_sort
    - Este site oferece o Bubble Sort em MUITAS linguagens de programação diferentes.

Exercícios para Entregar

1. Crie um algoritmo que crie uma lista de 100 números aleatórios entre -10.000 e 10.000. Sem ordenar a lista, separe todos os números dela em listas menores de:
    a. Zeros;
    b. Pares;
    c. Ímpares;
    d. Positivos;
    e. Negativos;
    f. Múltiplos de 3;
    g. Múltiplos de 5;
    h. Múltiplos de 7;
    i. Múltiplos de 3 e 5;
    j. Múltiplos de 3 e 7;
    k. Múltiplos de 5 e 7;
    l. Múltiplos de 3, 5 e 7;
    m. Números que não se encaixaram em nenhum dos critérios acima;

    Ao final do algoritmo, mostre o tamanho das 13 listas e seus respectivos valores. Caso alguma lista tenha ficado vazia, mostre uma mensagem informando ao usuário.

Exercícios para Praticar

1. Crie uma lista vazia e adicione nela os números de 1 a 10 usando um loop for.
2. Crie uma lista com os 10 primeiros números da sequência de Fibonacci.
3. Crie uma lista com o quadrado dos números de 1 a 10.
4. Crie uma lista com o cubo dos números de 1 a 10.
5. Crie uma lista com os números de 1 a 100, mas apenas os que são divisíveis por 3.
6. Crie uma lista com os números de 1 a 100, mas apenas os que são divisíveis por 5.
7. Crie uma lista com os números de 1 a 100, mas apenas os que são divisíveis por 3 e 5 ao mesmo tempo.
8. Crie uma lista com os valores de seno dos ângulos de 0 a 360 graus em incrementos de 30 graus, usando o módulo math.
9. Crie uma lista com os valores de cosseno dos ângulos de 0 a 360 graus em incrementos de 45 graus, usando o módulo math.
10. Crie uma lista com os valores de tangente dos ângulos de 0 a 360 graus em incrementos de 60 graus, usando o módulo math.
11. Crie uma lista com 10 números aleatórios composto por apenas 0 e 1, usando o módulo random.
12. Crie uma lista com 10 números aleatórios inteiros entre 1 e 100, usando o módulo random.
13. Crie uma lista com 5 valores aleatórios escolhidos de uma lista predefinida de 20 números, usando o módulo random (lembre-se da função choice).
14. Crie uma lista com 10 valores aleatórios escolhidos de uma lista predefinida de strings, mas sem repetição, usando o módulo random (lembre-se da função choice).
15. Crie uma lista com os números de 1 a 20, mas apenas os que são divisíveis por 4, e ordene-os em ordem decrescente.
16. Crie uma lista de 10 números que serão os resultados dos fatoriais dos números de 1 a 10. Insira os dados usando um for loop e calcule o fatorial usando um while.
17. Crie uma lista com 10 números pares usando o for. Depois adicione mais 20 números aleatórios usando o while, a partir do último valor inserido pelo for.
18. Crie uma lista com 10 números ímpares usando o for. Depois adicione mais 20 números aleatórios usando o while, a partir do último valor inserido pelo for.