Laboratório de Programação Competitiva

Profa. Silvia Brandão

2024.2

Aula de hoje

Resolução de exercícios usando:

- vetores em Python.
- estruturas de Controle, incluindo Condicionais e Loops
- estruturas de dados em Python: listas, tuplas, dicionários e conjuntos.

Discussão sobre estratégias de resolução de código.

Elaboração de códigos

Elabore funções em Python para:

- ler um conjunto de números reais, diferentes de zero, armazenando-os no vetor **vet**, até que o usuário digite zero. Zero não poderá fazer parte do vetor;
- calcular a média aritmética dos valores dos vetor;
- determinar o maior valor do vetor;
- determinar a quantidade de números ímpares no vetor

Observação: Um número real só pode ser classificado como par ou ímpar se for um número inteiro. Então converta o número real em inteiro.

Para testar as funções (métodos) , implemente o programa principal com a chamada das funções e imprima os resultados obtidos em saída formatada.

Elaboração de códigos - Beecrowd

Última aula: Faça os exercícios 1015 e

Use quando necessário: *x1, y1 = map(float, input().split())*

Faça os exercícios destacados

1°, 2°, 3°, 7°, 10°

Exercícios em Python – Beecrowd 2782

beecrowd | 2782

Escadinha

Por OBI Brasil

Timelimit: 1

Dizemos que uma sequência de números é uma escadinha, se a diferença entre números consecutivos é sempre a mesma. Por exemplo, "2, 3, 4, 5" e "10, 7, 4" são escadinhas. Note que qualquer sequência com apenas um ou dois números também é uma escadinha! Neste problema estamos procurando escadinhas em uma sequência maior de números. Dada uma sequência de números, queremos determinar quantas escadinhas existem. Mas só estamos interessados em escadinhas tão longas quanto possível. Por isso, se uma escadinha é um pedaço de outra, consideramos somente a maior. Por exemplo, na sequência "1, 1, 1, 3, 5, 4, 8, 12" temos 4 escadinhas diferentes: "1, 1, 1", "1, 3, 5", "5, 4" e "4, 8, 12".

Entrada

A primeira linha da entra contém um inteiro **N** ($1 \leq N \leq 1000$) indicando o tamanho da sequência de números. A segunda linha contém **N** inteiros definindo a sequência. O valor dos números da sequência está entre -10^6 e 10^6 inclusive.

Saída

Imprima uma linha contendo um inteiro representando quantas escadinhas existem na sequência.

Exemplos de Entrada	Exemplos de Saída
8 1 1 1 3 5 4 8 12	4
1 112	1
5 11 -106 -223 -340 -457	1

				10	7	4	
			5	10			
2	3	4	5				

					4	8	12
				5	4		
		1	3	5			
1	1	1					

```
def contar_escadinhas(seq):
    # Se houver apenas um número, há exatamente uma escadinha
    if len(seq) == 1:
        return 1

    # Inicialização
    num_escadinhas = 0
    j = 0
    n = len(seq) – 1

    # Percorrendo a sequência para encontrar escadinhas
    while j < n:
        # Identifica a diferença inicial
        i = j
        diff = seq[i+1] - seq[i]

        # Avança até que a diferença mude
        while i < n and seq[i+1] - seq[i] == diff:
            print(f"seq[{i}] = {seq[i]} e seq[{i+1}] = {seq[i+1]} e diff = {diff}")
            i += 1

        # Quando a escadinha termina, contamos
        num_escadinhas += 1
        j = i

    return num_escadinhas
```

```
# Leitura da entrada
N = int(input())
seq = list(map(int, input().split()))

# Processamento e saída do resultado
print(contar_escadinhas(seq))
```

				5	4		
		1	3	5			
1	1	1					

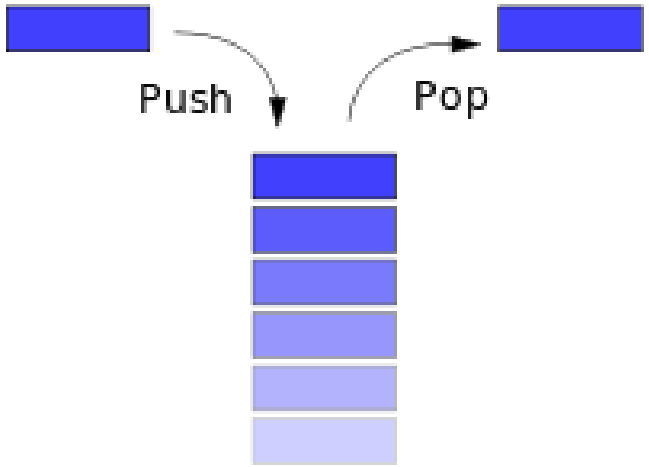
Tarefa MANUSCRITA – valor 5pts

Pesquise e responda as perguntas a seguir, exemplificando em Python:

a) como se dá a manipulação de uma lista (operações de acesso ao elemento)? Exemplifique.
b) qual a forma de acesso aos dados de uma fila? E, como acessar um deque? Exemplifique.
c) qual a forma de acesso aos dados de uma pilha? Exemplifique.
d) o que é Compreensões de lista? Exemplifique.
e) vamos explorar os conceitos de dicionários e conjuntos? Explique e exemplifique.

Grupos de 2 alunos.
Entrega do material escrito e apresentação dos tópicos pelos grupos de alunos.

Estruturas de dados:
Tipos Abstratos de Dados (TAD) - Pilhas e Filas



Pilha

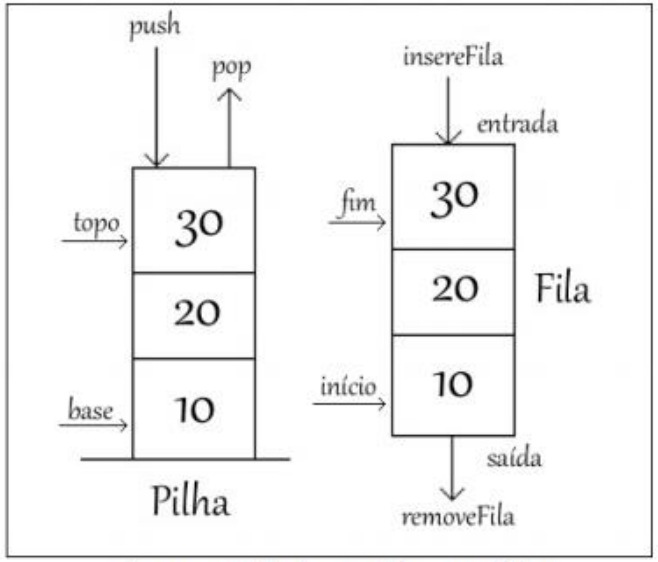
Estratégia LIFO *(Last in, first out)*

Os elementos da pilha são retirados na ordem inversa à ordem em que foram introduzidos: o último a entrar é o primeiro a sair.

Fila

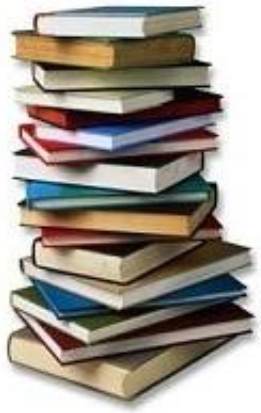
Estratégia FIFO *(Fist in, first out)*

A estrutura de fila é análoga ao conceito que temos de filas em geral. O primeiro a chegar é sempre o primeiro a sair, e a entrada de novos elementos sempre se dá no fim da fila.



Representação de uma Fila e uma Pilha

Estruturas de dados em Python:



Usando listas como pilhas

Os métodos de lista tornam muito fácil utilizar listas como pilhas, onde o item adicionado por último é o primeiro a ser recuperado (política de acesso: "último a entrar, primeiro a sair").

- Para adicionar um item ao topo da pilha, use append().
- Para recuperar um item do topo da pilha use pop() sem nenhum índice explícito.

Usando listas como filas

Você também pode usar uma lista como uma fila, onde o primeiro item adicionado é o primeiro a ser recuperado (política de acesso: "primeiro a entrar, primeiro a sair"); porém, listas não são eficientes para esta finalidade. Embora appends e pops no final da lista sejam rápidos, fazer inserts ou pops no início da lista é lento (porque todos os demais elementos têm que ser deslocados).

- Para implementar uma fila, use a classe collections.deque que foi projetada para permitir appends e pops eficientes nas duas extremidades.

Deque

É uma estrutura de dados (fila) na qual os elementos podem ser inseridos ou excluídos de qualquer uma de suas extremidades (do início ou do fim).

```
[11] from collections import deque

     queue = deque(["Eric", "John", "Michael"])
     queue.append("Terry")           # Terry arrives
     queue.append("Graham")          # Graham arrives
     print(queue)
```

```
deque(['Eric', 'John', 'Michael', 'Terry', 'Graham'])
```

```
[12] queue.popleft()                 # The first to arrive now leaves
```

```
'Eric'
```

```
[13] queue.popleft()                 # The second to arrive now leaves
```

```
'John'
```

```
     queue                           # Remaining queue in order of arrival
```

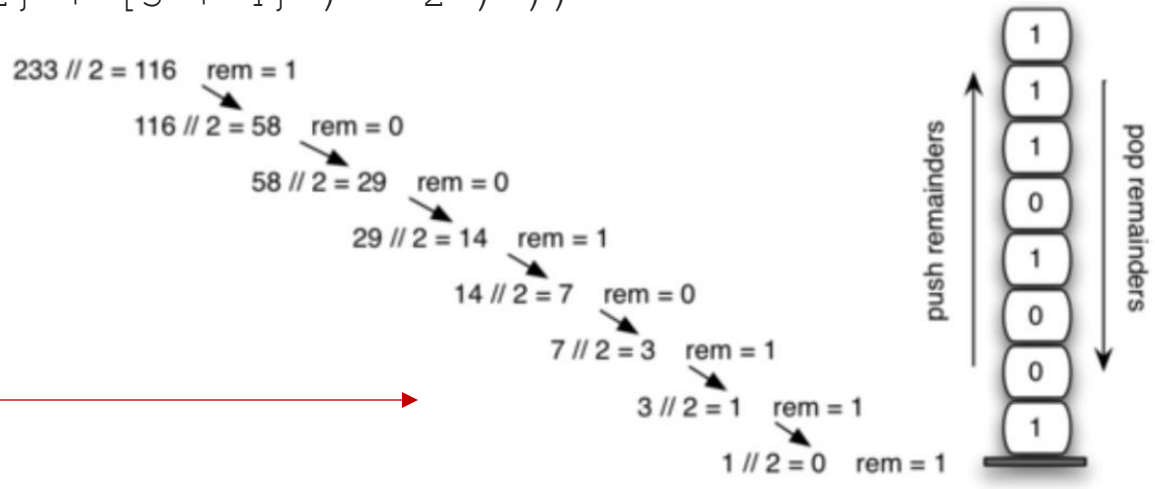
```
deque(['Michael', 'Terry', 'Graham'])
```

Tarefa: usando o TAD Pilha – entregar próxima aula

1. Faça um código em Python que use a estrutura de dados pilha para verificar o balanceamento dos parênteses, colchetes e chaves em expressões aritméticas:

```
print('Expressoes simples')
print(verifica_expressao_simples('((1 + 2) * (3 + 4)) - (5 + 6)'))
print()
print(verifica_expressao_simples('((1 + 2) * (3 + 4))))'))
print()
print(verifica_expressao_simples('(((((1 + 2) * (3 + 4))'))
print()
print('Expressoes compostas')
print(verifica_expressao_composta('{ [ (1 + 2) + (3 + 4) ] * 2 }'))
print()
print(verifica_expressao_composta('{ [ (1 + 2} + [3 + 4} ) * 2 )'))
```

2. Implemente uma função em Python que converta um número inteiro arbitrário na base 10 (decimal) para a representação binária.



Próxima Aula

Teoria dos Números x Teoria dos Números em Python:

- Algoritmo de Euclides. MDC. Implementação.
- Cálculo do Fatorial tendo como resultado números grandes.
- Teorema fundamental da aritmética: MDC, MMC, Primos, Fatorial. Implementações.
- Discussão em grupo sobre estratégias de resolução.

Não se esqueçam do Uniube+