Namespace Documentation

desafio_1 Namespace Reference

Functions

- def **tamamhoLista**
- def **validaS**
- def **temSoma**

Variables

- list **vet** = [1, 2, 3, 4, 5, 6, 7, 8, 9]
- int **soma** = 9

Function Documentation

def desafio_1.tamamhoLista (*lista*)

Parameters:

lista	vetor de números inteiros

Returns:

o tamanho da lista complexidade O(n)

```
   18 def tamamhoLista(lista: list) -> int:
   19     """ Função que retorna o tamanho da lista passada como parâmetro """
   20     c = 0
   21     for _ in lista:
   22         c += 1
   23     return c
   24
   25
```

def desafio_1.temSoma (*lista*)

Parameters:

lista	Vetor de números inteiros
s	O valor da soma buscada tomando dois elementos do vetor

```
   47 def temSoma(lista: list, s: int):
   48     """
   49     Função que indica se a combinação de dois elementos distintos do vetor somados
corresponde a s
   50
   51     """
   52     # indice do elemento que esta sendo tomado atualmente em comparação aos demais
   53     atual = 0
   54     # indice dos elementos que serão comparados com o número atual
   55     prox = 1
   56     tam = tamamhoLista(lista)
   57     menor, somatot = validaS(lista)
   58
   59     result = ""
   60     if (menor < s) and (somatot > s) and (tam >= 2):
   61         while True:
   62             if (lista[atual] + lista[prox]) == s:
   63                 result += "{} + {} = {}\n".format(lista[atual], lista[prox], s)
   64
   65             if prox < tam - 1:
   66                 prox += 1
   67             else:
   68                 if atual == tam - 1:
```

```
   69                     break
   70                 else:
   71                     atual += 1
   72                     if atual < tam - 1:
   73                         prox = atual + 1
   74     if result:
   75         print(result)
   76     else:
   77         print("Falso: não existe nenhuma combinação de dois elementos que some",
s)
   78
   79
```

def desafio_1.validaS (*lista*)

Parameters:

lista	lista vetor de números inteiros

Returns:

o menor valor da lista e a soma de seu elementos complexidade O(n)

```
   30 def validaS(lista: list) -> tuple:
   31     """ Função que auxilia na validaçao da soma """
   32     if lista:
   33         menor = lista[0]
   34         somatot = 0
   35         for i in lista:
   36             if i < menor:
   37                 menor = i
   38             somatot += i
   39         return menor, somatot
   40     return 0, 0
   41
   42
```

Variable Documentation

int desafio_1.soma = 9

list desafio_1.vet = [1, 2, 3, 4, 5, 6, 7, 8, 9]

desafio_2 Namespace Reference

Functions

- def **inverte** (frase)
- def **junta** (texto)
- def **tamanho** (texto)
- def **palindromo** (t)

Variables

- string **texto1** = 'Hoje subi no onibus correndo.'
- string **texto2** = 'Na casa do vizinho a grama é amarga'
- string **texto3** = 'A arara azul é linda!'

Function Documentation

def desafio_2.inverte (*frase*)

Parameters:

frase	é o texto(sem espaços em branco) onde o palíndromo deve estar ou não

Returns:

t retorna o texto passado como argumento de trás para frente complexidade O(tam)
função que concatena os caracteres da frase em ordem inversa

```
18 def inverte(frase):
19     """ função que concatena os caracteres da frase em ordem inversa"""
20     tam = tamanho(frase)
21     t = ''
22     for i in range(tam):
23         t += frase[(tam - 1) - i]
24     return t
25
26
```

def desafio_2.junta (*texto*)

Parameters:

texto	texto onde o palíndromo é buscado

Returns:

t complexidade O(len(texto))
função para remover todos os espaços em branco do texto e juntá-lo

```
31 def junta(texto):
32     """ função para remover todos os espaços em branco do texto e juntá-lo """
33     t = ''
34     for letra in texto:
35         if letra != " ":
36             t += letra
37     return t
38
39
```

def desafio_2.palindromo (*t*)

Parameters:

t	texto onde o palíndromo é buscado @ return t retorna o maior palíndromo encontrado no texto passado complexidade O(n^2)

	Função que retorna o maior palíndromo encontrado

```
   56 def palindromo(t):
   57     """ Função que retorna o maior palíndromo encontrado """
   58     texto = junta(t)  # aqui os espaços em branco são removidos e texto é juntado
   59     primeiro = 0  # indice para o primeiro caracter do possível palíndromo
   60     ultimo = tamanho(texto) - 1  # indice para o último caracter do possível
palíndromo
   61     c = 0  # contador para os caracteres do texto
   62     t = ''  # variável para guardar o palíndromo encontrado no texto
   63     controle = 0  # serve para garantir que o maior palíndromo seja guardado em
t
   64     while c < tamanho(texto):  # enquanto c não alcança o tamanho do texto - 1
   65         while ultimo > primeiro:
   66             #  se o primeiro caracter do texto for igual ao último
   67             #  e se o inverso dos caracteres entre o primeiro e o último for
   68             #  igual aos carateres do texto entre o primeiro e o último
   69             if texto[primeiro] == texto[ultimo] and \
   70                     inverte(texto[primeiro: ultimo + 1]) == texto[primeiro:
ultimo + 1]:
   71                 text = texto[primeiro:ultimo + 1]
   72                 if tamanho(text) > controle:
   73                     controle = tamanho(text)
   74                     t = text
   75             ultimo -= 1
   76         ultimo = tamanho(texto) - 1
   77         primeiro += 1
   78         c += 1
   79     return t
   80
   81
```

def desafio_2.tamanho (*texto*)

Parameters:

texto	texto onde o palíndromo é buscado

Returns:

c contador de caracteres do texto complexidade O(len(texto))

```
   45 def tamanho(texto):
   46     c = 0
   47     for _ in texto:
   48         c += 1
   49     return c
   50
   51
```

Variable Documentation

string desafio_2.texto1 = 'Hoje subi no onibus correndo.'

string desafio_2.texto2 = 'Na casa do vizinho a grama é amarga'

string desafio_2.texto3 = 'A arara azul é linda!'

desafio_3 Namespace Reference

Variables

- **A1** = int(input("A1: "))
- **A2** = int(input("A2: "))
- **A3** = int(input("A3: "))
- int **a1** = 0 else 0
- **a2** = **A2**
- int **a3** = 1000 else 0
- int **tA1** = **a2** * 2 + **a3** * 4
- int **tA2** = **a1** * 2 + **a3** * 2
- int **tA3** = **a1** * 4 + **a2** * 2
- int **menor_tempo** = **tA1**

Variable Documentation

desafio_3.A1 = int(input("A1: "))

int desafio_3.a1 = 0 else 0

desafio_3.A2 = int(input("A2: "))

desafio_3.a2 = A2

desafio_3.A3 = int(input("A3: "))

int desafio_3.a3 = 1000 else 0

desafio_3.menor_tempo = tA1

int desafio_3.tA1 = a2 * 2 + a3 * 4

int desafio_3.tA2 = a1 * 2 + a3 * 2

int desafio_3.tA3 = a1 * 4 + a2 * 2

desafio_4 Namespace Reference

Classes

- class **No**
  classe para implementação dos nós de uma árvore binária

Variables

- **raiz** = **No**(8)
- **n1** = **No**(3)
- **n2** = **No**(10)
- **n3** = **No**(2)
- **n4** = **No**(11)
- **n5** = **No**(15)
- **n6** = **No**(16)
- **n7** = **No**(9)

Variable Documentation

desafio_4.n1 = No(3)

desafio_4.n2 = No(10)

desafio_4.n3 = No(2)

desafio_4.n4 = No(11)

desafio_4.n5 = No(15)

desafio_4.n6 = No(16)

desafio_4.n7 = No(9)

desafio_4.raiz = No(8)

desafio_big_Data Namespace Reference

Python 3.7 system os windows10 Desafio 1.

Detailed Description

Python 3.7 system os windows10 Desafio 1.

Desafio 4 Python 3.7 system os windows10.

Desafio 3 Python 3.7 system os windows10.

Desafio 2 Python 3.7 system os windows10.

Author:
Wellington Oliveira

Since:
15/07/2018

Author:
Wellington Oliveira

Since:
18/07/2018

Author:
Wellington Oliveira

Since:
15/07/2018 complexidade O(2)

Class Documentation

desafio_4.No Class Reference

classe para implementação dos nós de uma árvore binária

Public Member Functions

- def **__init__** (self, **chave**, **valor**=None, **no_direito**=None, **no_esquerdo**=None)
- def **inseriNo** (self, no)
- def **imprimeBFS** (self)
  imprimindo os nós da árvore em ordem de busca em amplitude
- def **__repr__** (self)
  O objeto é representado por sua chave.

Public Attributes

- **chave**
- **valor**
- **no_direito**
- **no_esquerdo**
- **filhos_no**
- **fila**

Detailed Description

classe para implementação dos nós de uma árvore binária

Parameters:

chave	é um número inteiro único que identifica cada nó é a informação contida no nó
no_direito	filho direito de um dado nó
no_esquerdo	filho esquerdo de um dado nó
filhos_no	é uma lista com os filhos esquerdo e direito do no
fila	é uma lista com os nós na ordem em que foram inseridos na fila

---

Constructor & Destructor Documentation

def desafio_4.No.__init__ (*self*, *chave*, *valor* = `None`, *no_direito* = `None`, *no_esquerdo* = `None`)

```
24     def __init__(self, chave, valor=None, no_direito=None, no_esquerdo=None):
25         self.chave = chave
26         self.valor = valor
27         self.no_direito = no_direito
28         self.no_esquerdo = no_esquerdo
29         self.filhos_no = []  # se  vazio o nó é folha
30         self.fila = [self]  # o primeiro nó na fila é o no raiz da subarvore
31
```

Member Function Documentation

def desafio_4.No.__repr__ (*self*)

O objeto é representado por sua chave.

```
   87     def __repr__(self):
   88         return f"{self.chave}"
   89 
   90 
```

def desafio_4.No.imprimeBFS (*self*)

imprimindo os nós da árvore em ordem de busca em amplitude

Parameters:

s	é uma lista cotendo os nós na ordem em que foram visitados função que imprime os nós inseridos em ordem de largura

```
   70     def imprimeBFS(self):
   71         """
   72         função que imprime os nós inseridos em ordem de largura
   73 
   74         """
   75         s = [self]  # o primeiro nó na lista é o nó raiz da árvore
   76         while self.fila:
   77             pai = self.fila.pop(0)  # retira da fila o primeiro nó e assim a cada
laço a filha vai esvaziando
   78             for filho in pai.filhos no:
   79                 if filho:
   80                     s.append(filho)
   81 
   82         s = "".join(str(s))
   83         print(s)
   84 
```

def desafio_4.No.inseriNo (*self*, *no*)

```
Função que avalia o nó passado para a função
e o insere na arvore binária segundo a sua chave
```

```
   41     def inseriNo(self, no):
   42         """
   43         Função que avalia o nó passado para a função
   44         e o insere na arvore binária segundo a sua chave
   45         """
   46         if no.chave == self.chave:  # avalia se o nó passado possui uma chave
única
   47             print('chave inválida')
   48         else:
   49             # a partir daqui o no será inserido ou na subarvore esquerda ou direita
do nó raiz da subarvore
   50             if no.chave > self.chave:
   51                 if self.no direito is None:
   52                     self.no_direito = no
   53                 else:
   54                     self.no_direito.inseriNo(no)
   55 
   56             else:
   57                 if self.no esquerdo is None:
   58                     self.no_esquerdo = no
   59                 else:
   60                     self.no_esquerdo.inseriNo(no)
   61             # chamada para inserção dos nós na lista de filhos
   62             self.__setFilhos(self.no_esquerdo, self.no_direito)
   63             # o nó inserido na árvore vai para a fila de nós
   64             self.fila.append(no)
   65 
```

Member Data Documentation

desafio_4.No.chave

desafio_4.No.fila

desafio_4.No.filhos_no

desafio_4.No.no_direito

desafio_4.No.no_esquerdo

desafio_4.No.valor

The documentation for this class was generated from the following file:

- C:/Users/requi/Desktop/desafio_bigData/**desafio_4.py**

File Documentation

C:/Users/requi/Desktop/desafio_bigData/desafio_1.py File Reference

Namespaces

- **desafio_1**
- **desafio_big_Data**

Python 3.7 system os windows10 Desafio 1. Functions

- def **desafio_1.tamamhoLista**
- def **desafio_1.validaS**
- def **desafio_1.temSoma**

Variables

- list **desafio_1.vet** = [1, 2, 3, 4, 5, 6, 7, 8, 9]
- int **desafio_1.soma** = 9

C:/Users/requi/Desktop/desafio_bigData/desafio_2.py File Reference

Namespaces

- **desafio_2**
- **desafio_big_Data**

Python 3.7 system os windows10 Desafio 1. Functions

- def **desafio_2.inverte** (frase)
- def **desafio_2.junta** (texto)
- def **desafio_2.tamanho** (texto)
- def **desafio_2.palindromo** (t)

Variables

- string **desafio_2.texto1** = 'Hoje subi no onibus correndo.'
- string **desafio_2.texto2** = 'Na casa do vizinho a grama é amarga'
- string **desafio_2.texto3** = 'A arara azul é linda!'

C:/Users/requi/Desktop/desafio_bigData/desafio_3.py File Reference

Namespaces

- **desafio_3**
- **desafio_big_Data**

Python 3.7 system os windows10 Desafio 1. Variables

- **desafio_3.A1** = int(input("A1: "))
- **desafio_3.A2** = int(input("A2: "))
- **desafio_3.A3** = int(input("A3: "))
- int **desafio_3.a1** = 0 else 0
- **desafio_3.a2** = A2
- int **desafio_3.a3** = 1000 else 0
- int **desafio_3.tA1** = a2 * 2 + a3 * 4
- int **desafio_3.tA2** = a1 * 2 + a3 * 2
- int **desafio_3.tA3** = a1 * 4 + a2 * 2
- int **desafio_3.menor_tempo** = tA1

C:/Users/requi/Desktop/desafio_bigData/desafio_4.py File Reference

Classes

- class **desafio_4.No**
  classe para implementação dos nós de uma árvore binária

Namespaces

- **desafio_4**
- **desafio_big_Data**

Python 3.7 system os windows10 Desafio 1. Variables

- **desafio_4.raiz** = No(8)
- **desafio_4.n1** = No(3)
- **desafio_4.n2** = No(10)
- **desafio_4.n3** = No(2)
- **desafio_4.n4** = No(11)
- **desafio_4.n5** = No(15)
- **desafio_4.n6** = No(16)
- **desafio_4.n7** = No(9)