INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 03 – DICIONÁRIOS E SETS

O QUE IREMOS APRENDER

01 SETS E SUAS CARACTERÍSTICAS

02 OPERAÇÕES MATEMÁTICAS COM SETS

03 DICIONÁRIOS E SUAS CARACTERÍSTICAS

04 PERCORRENDO DICIONÁRIOS

05 OPERAÇÕES SUPORTADAS NOS DICIONÁRIOS

06 INTRODUÇÃO AULA 03

Sets e suas características

Um "set" (conjunto) é uma coleção de elementos únicos e não ordenados. Isso significa que um conjunto não permite elementos duplicados e não mantém a ordem de inserção dos elementos.

Os conjuntos são uma das estruturas de dados fundamentais da linguagem Python e são úteis para várias operações matemáticas e tarefas que envolvem coleções únicas de elementos.

Sets e suas características

Veja a seguir alguns exemplos com sets em Python:

```
meu_set = {'Infinity', 'School'}
print(type(meu_set))
#<class 'set'>
```

```
frutas = {"maçã", "banana", "cereja", "maçã"}

print(frutas)
#{'maçã', 'cereja', 'banana'}
```

Sets e suas características

Para se criar um set (conjunto) possui algumas formas:

1.Elementos separados por Vírgula dentro de um Conjunto:

```
set1 = {'Infinity','School','202...'}
```

2. Usando Compreensão de Conjuntos:

```
set1 = {letra for letra in 'infinity' if letra not in 'aeiou'}
```

3. Usando o método Construtor:

```
set1 = set(['Infinity','School','202...'])
print(set1)
```

Sets e suas características

Após a criação de um Set, você não pode alterar seus itens. Contudo você pode adicionar novos itens e para isso podemos utilizar o método **add()**.

```
convidados = {'João', 'Maria', 'Eduarda'}

convidados.add('Marcela')
print(convidados)
#{'Marcela', 'Eduarda', 'João', 'Maria'}
```

Sets e suas características

Para adicionar itens de outro conjunto ao set especificado, podemos utilizar o método update(). Você pode utilizar esse método com qualquer tipo de objeto iterável (tuplas, listas, dicionários etc.)

```
ids = {10, 12, 13, 14}
novos_ids = {11, 13, 15}

ids.update(novos_ids)
print(ids)
#{10, 11, 12, 13, 14, 15}
```

Sets e suas características

Sets não possibilitam acessar seus elementos através de índices (assim como Listas) ou chaves (como os Dicionários).
Assim, podemos acessá-los de duas maneiras: percorrendo o conjunto ou verificando se o elemento desejado se encontra no set.

```
set_1 = {1, 2, 3}

print(set_1[0])
#  File "c:\Users\RAAMA\Desktop\Workshop\teste\teste.py", line 3, in <module>
#    print(set_1[0])
#TypeError: 'set' object is not subscriptable
```

Sets e suas características

```python
convidados = {'João', 'Maria', 'Eduarda'}

print('Maria' in convidados)
#True
```

```python
convidados = {'João', 'Maria', 'Eduarda'}
for x in convidados:
    print(x)
    # Eduarda
    # Maria
    # João
```

Sets e suas características

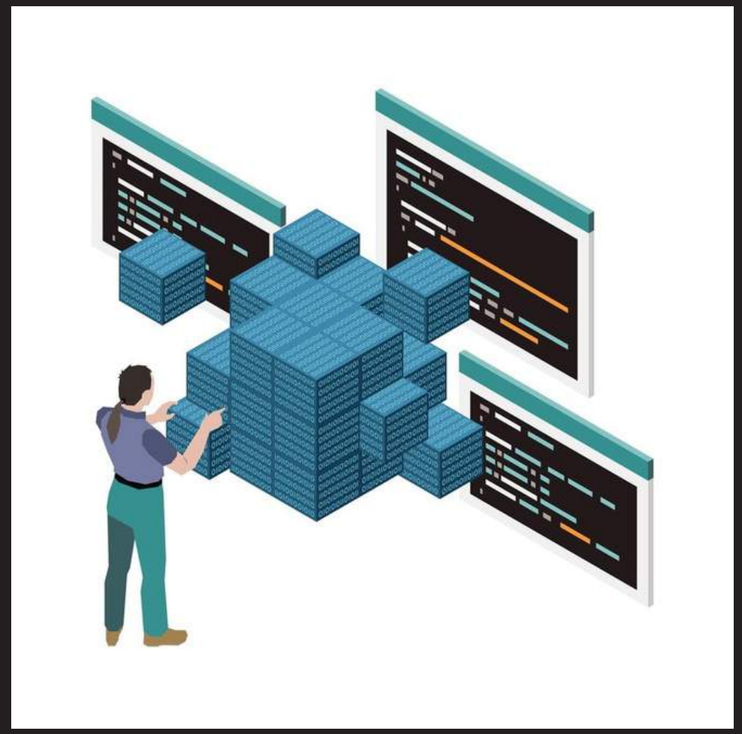
Para remover itens de um set, você pode, inicialmente, utilizar dois métodos: o remove() e discard().

```
convidados = {'João', 'Maria', 'Eduarda'}

print(convidados.remove('Maria'))
print(convidados.discard('Eduarda'))
print(convidados)
#{'João'}
```

Operações Matemáticas com Sets

Os Sets em Python nada mais são que Conjuntos Matemáticos. Neles, você também pode aplicar os conceitos de Interseção, União, Diferença e etc.



Operações Matemáticas com Sets

Intersection()

O método intersection() retorna um novo conjunto contendo apenas os itens presentes em ambos:

```
convidados = {'João', 'Maria', 'Eduarda'}
convidados2 = {'Pedro', 'Raama', 'Maria'}

print(convidados.intersection(convidados2))
#{'Maria'}
```

Operações Matemáticas com Sets

Intersection_update()

Se você deseja já atualizar um dos sets com a interseção entre eles, use o método intersection_update():

```
convidados = {'João', 'Maria', 'Eduarda'}
convidados2 = {'Pedro', 'Raama', 'Maria'}

convidados.intersection_update(convidados2)
print(convidados)
#{'Maria'}
```

Operações Matemáticas com Sets

union()

Você pode utilizar o método union()
para retornar um
conjunto de elementos contendo
elemento de ambos sets:

```
set1 = {1, 2, 3}
set2 = {'z', 'x', 'a'}

print(set1.union(set2))
# ou
print(set1 | set2)
#{1, 2, 3, 'x', 'a', 'z'}
#{1, 2, 3, 'x', 'a', 'z'}
```

ATIVIDADE PRÁTICA 1

Crie um conjunto vazio chamado frutas e adicione as seguintes frutas a ele: "maçã", "banana", "uva", "laranja" e "morango". Em seguida, imprima o conjunto.

ATIVIDADE PRÁTICA 2

Verifique se a fruta "banana" está presente no conjunto frutas e imprima o resultado.

ATIVIDADE PRÁTICA 3

Crie um conjunto chamado frutas_vermelhas e adicione as seguintes frutas a ele: "morango", "cereja" e "framboesa". Em seguida, imprima o conjunto.

ATIVIDADE PRÁTICA 4

Remova a fruta "cereja" do conjunto frutas_vermelhas e imprima o conjunto atualizado.

ATIVIDADE PRÁTICA 5

Crie dois conjuntos, A e B, e realize a união dos dois conjuntos.

ATIVIDADE PRÁTICA 6

Crie um programa que recebe dois conjuntos e exibe a interseção deles.

ATIVIDADE PRÁTICA 7

Escreva um programa que receba duas listas e calcule a união dos elementos únicos dessas listas, usando conjuntos.

Dicionários e suas características

Dicionários são uma das estruturas de dados que permitem armazenar valores associados a chaves, formando assim uma espécie de “mapeamento” entre chaves e valores. Em outras palavras, um dicionário é uma coleção de pares chave-valor.

```
dicionario= {
    'Instituição': 'Infinity School',
    'Curso': 'Dev Full Stack',
    'Módulo': 'Python'
}

print(dicionario)
print(type(dicionario))
```

```
{'Instituição': 'Infinity School', 'Curso': 'Dev Full Stack', 'Módulo': 'Python'}
<class 'dict'>
```

Dicionários e suas características

Em Python, os dicionários são representados por chaves {} e os pares chave-valor são separados por dois pontos :. O uso de dicionários no desenvolvimento de algoritmos de inteligência artificial é muito comum, devido a facilidade e praticidade.

OBS: Os elementos de um dicionário são armazenados de forma não ordenada.

```
dados_usuario= {
    'Nome': 'Raama',
    'Idade': 18,
    'Cidade': 'Salvador'
}
```

Exemplo de Dicionário
Nome, Idade e Cidade são as chaves
Raama, 18 e Salvador são os valores

Dicionários e suas características

Em Python, os dicionários você pode criar dicionários utilizando chaves vazias. `meu_dicionario={}`

Usando o construtor dict():

```
dicionario = dict() #Dicionário vaziio
dicionario = dict([('Modulo', 'Python'), ('Instituição', 'Infinity School')])
dicionario = dict(Modulo = 'Python' , Instituicao = 'Infinity School')
```

Dicionários e suas características

Embora seja menos comum, você pode criar um dicionário vazio usando uma compreensão de dicionário:

```
1  dicionario = {elemento: f'Valor {elemento}' for elemento in range(6)}
2  print(dicionario)
```

```
{0: 'Valor 0', 1: 'Valor 1', 2: 'Valor 2', 3: 'Valor 3', 4: 'Valor 4', 5: 'Valor 5'}
```

Dicionários e suas características

O método .items() é utilizado em dicionários em Python para obter uma visualização dos pares chave-valor presentes no dicionário. Especificamente, o método retorna uma view (uma visão) que mostra uma lista de tuplas, onde cada tupla contém um par chave-valor do dicionário.

```python
linguagens = {"Programação01":"Python",
              "Programação02":"JavaScript",
              "Programação03":"PHP"}

for i in linguagens.items():
    print(i)
```

```
('Programação01', 'Python')
('Programação02', 'JavaScript')
('Programação03', 'PHP')
```

```python
linguagens = {"Programação01":"Python",
              "Programação02":"JavaScript",
              "Programação03":"PHP"}

for chave,valor in linguagens.items():
    print(f"Ola {chave}, {valor}")
```

```
Ola Programação01, Python
Ola Programação02, JavaScript
Ola Programação03, PHP
```

Dicionários e suas características

```
dados_usuario= {
    'Nome': 'Raama',
    'Idade': 18,
    'Cidade': 'Salvador'
}

print(dados_usuario['Idade'])
```

```
dados_usuario= {
    'Nome': 'Raama',
    'Idade': 18,
    'Cidade': 'Salvador'
}

print(dados_usuario.get('Nome'))
```

Para acessar um valor de um dicionário basta printarmos o dicionário e colocar entre colchetes e aspas a chave que queremos. Também podemos utilizar a função get() para acessar um valor.

ATIVIDADE PRÁTICA 8

Escreva um programa que EXIBA um dicionário contendo informações de pessoas (nome, idade) e exiba essas informações.

ATIVIDADE PRÁTICA 9

Escreva um programa que percorra as chaves e valores de um dicionário separadamente e os exiba.

Operações Suportadas nos Dicionários

01

list(dicionario)

Retorna uma lista com todas as chaves usadas no dicionário

02

len(dicionario)

Retorna o número de itens de um dicionário

03

dicionario[chave]

Retorna o valor da chave especificada entre colchetes. Caso a chave não exista, uma exceção do tipo **KeyError** será lançada

04

dicionario[chave] = valor

Se a chave já existir no dicionário, terá seu valor sobrescrito. Se a chave não existir, ela será criada e o valor será atribuída a ela

05

del dicionario[chave]

Remove a chave e seu respectivo valor do dicionario

06

chave in dicionario

Retorna **True** se a chave for encontrada no dicionário, senão, retornará **False**

IN

Operações Suportadas nos Dicionários

list(dicionario)

```
1  computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}
2
3  print(list(computador))
4  #['CPU', 'RAM', 'SSD']
```

dicionario[chave] = valor

```
1  computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}
2
3  print(computador['CPU'])#Intel
4  computador['CPU']= 'Xiaomi'
5  print(computador['CPU'])#Xiaomi
```

len(dicionario)

```
1  computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}
2
3  print(len(computador))
4  #3
```

dicionario[chave]

```
1  computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}
2
3  print(computador['RAM'])
4  #8gb
```

Operações Suportadas nos Dicionários

del dicionario[chave]

```python
computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}

del(computador['CPU'])
print(computador)
#{'RAM': '8gb', 'SSD': '250bg'}
```

chave in dicionario

```python
computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}
print('CPU' in computador)
#True
```

Operações Suportadas nos Dicionários

01

chave not in dicionario

Retorna True se a chave não for encontrada no dicionário, senão, retornará False

02

dicionario.clear()

Remove todos os itens do dicionário

03

iter()

Retorna um iterador para as chaves do dicionário. Retorna o mesmo que dicionario.keys()

04

dicionario.copy()

Retorna uma cópia do dicionário

05

dicionario.get(chave, valor padrão)

Retorna o valor para a chave especificada se esta existir no dicionário, senão, será retornado um valor padrão definido. Caso este valor não seja definido, a função retornará **None**

06

dict.fromkeys(iteravel)

Cria um novo dicionário com chaves proveniente do iteravel (uma lista, um dicionário), os valores por padrão serão None

IN

Operações Suportadas nos Dicionários

chave not in dicionario

```
computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}

print('CPU' not in computador)
#False
```

iter()

```
computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}

print(iter(computador))
#<dict_keyiterator object at 0x00000240520E3DD0>
print(list(iter(computador)))
#['CPU', 'RAM', 'SSD']
```

dicionario.clear()

```
computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}

print(computador)
#{'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}
computador.clear()
print(computador)
#{}
```

dicionario.copy()

```
computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}

sistema_computador= computador.copy()
print(sistema_computador)
#{'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}
```

Operações Suportadas nos Dicionários

dicionario.get(chave, valor padrão)

```
computador = {'CPU': 'Intel', 'RAM': '8gb', 'SSD': '250bg'}

print(computador.get('CPU'))#Intel
print(computador.get('Placa Mãe'))#None
print(computador.get('Placa de Vídeo', 'Chave Inexistente'))
#Chave Inexistente
```

dict.fromkeys(iteravel)

```
computador = ['CPU', 'RAM', 'SSD']

print(computador)
#['CPU', 'RAM', 'SSD']
print(dict.fromkeys(computador))
#{'CPU': None, 'RAM': None, 'SSD': None}
```

ATIVIDADE PRÁTICA 10

Desenvolva um programa que recebe um dicionário, uma chave e um valor como entrada e adiciona a chave e o valor ao dicionário, atualizando o valor se a chave já existir.

ATIVIDADE PRÁTICA 11

Escreva um programa que recebe um dicionário e uma lista de chaves como entrada e verifica se todas as chaves da lista existem no dicionário. A função deve retornar True se todas as chaves existirem e False caso contrário.

ATIVIDADE PRÁTICA 12

Crie um programa que simule um sistema de votação. O programa deve permitir que os eleitores escolham entre opções de eleitores e conte os votos para cada opção. Use um dicionário para armazenar os resultados da votação, onde as chaves são as opções e os valores são o número de votos para cada opção. O programa deve permitir que os eleitores votem, encerre a votação e exiba os resultados finais. Use While True e pare o programa somente se o usuário digitar o número 0 e exiba os resultados finais.

ATIVIDADE PRÁTICA 13

Crie um dicionário que relacione nomes de alunos às suas notas em uma disciplina. Calcule a média das notas e exiba-a.

ATIVIDADE PRÁTICA 14

Crie um programa que receba uma lista de números e remova todas as duplicatas usando um conjunto (set). Em seguida, exiba a lista original e a lista sem duplicatas.

ATIVIDADE PRÁTICA 15

Crie um programa que realize a união de múltiplos conjuntos e exiba o conjunto resultante.

DESAFIO PRÁTICO

Sistema de Cadastro de Alunos - passo 1

Cadastro de Alunos: O programa deve permitir ao usuário cadastrar alunos. Cada aluno terá as seguintes informações: nome, idade e notas em três disciplinas: Matemática, Ciências e História. Os dados de cada aluno devem ser armazenados em um dicionário com as seguintes chaves: 'nome', 'idade', 'notas'. As notas devem ser armazenadas em uma tupla.

DESAFIO PRÁTICO

Sistema de Cadastro de Alunos - passo 2

Visualização de Alunos: O programa deve permitir ao usuário visualizar todos os alunos cadastrados, exibindo suas informações de forma organizada.
Média de Notas: O programa deve calcular a média das notas de cada aluno e exibi-la.
Aluno com Melhor Média: O programa deve identificar e exibir o aluno com a melhor média de notas.

SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 04 DE PYTHON.
FUNÇÕES I.

IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

Introdução

Uma função é como uma mini-fábrica que realiza uma tarefa específica.

Imagine ter uma máquina que faz suco. Você insere frutas (entrada), a máquina processa (corpo da função) e, no final, você obtém suco (saída).

O paradigma funcional é um estilo de programação onde a base são as funções. Ele foca em realizar computações através da avaliação de funções, evitando mudança de estado e dados mutáveis.

Sintaxe, Definição e Escopo

Sintaxe e Definição: Aqui, def é a palavra-chave que define uma função, saudacao é o nome da função, e o que está dentro da função é o corpo.

```
def saudacao():
    print("Olá!")
```

```
saudacao() # Imprime "Olá!"
```

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 03 – DICIONÁRIOS E SETS