Aula 10

Professores:

Dante Corbucci Filho
Leandro A. F. Fernandes

Conteúdo:

- Estrutura de Dados
  - Conjunto (set)

Conjunto

- Python, assim como várias linguagens, inclui um tipo de dado chamado conjunto (set), que é uma estrutura de dados mutável, desordenada e sem elementos repetidos.

Conjunto

- Python, assim como várias linguagens, inclui um tipo de dado chamado conjunto (set), que é uma estrutura de dados mutável, desordenada e sem elementos repetidos.

- O uso básico deste tipo de dado se dá quando se necessita de teste de pertinência de um elemento em um conjunto ou eliminação de dados duplicados.

Conjunto

- Python, assim como várias linguagens, inclui um tipo de dado chamado conjunto (set), que é uma estrutura de dados mutável, desordenada e sem elementos repetidos.

- O uso básico deste tipo de dado se dá quando se necessita de teste de pertinência de um elemento em um conjunto ou eliminação de dados duplicados.

- Estruturas do tipo conjunto suportam as operações matemáticas:
  - A pertinência de um elemento a um conjunto,
  - A união de dois conjuntos,
  - A interseção de dois conjuntos,
  - A diferença de dois conjuntos,
  - etc.

Conjunto (*set*)

- Em Python, uma variável pode ser um conjunto contendo elementos de tipo(s) imutável(áveis), tais como números inteiros e de ponto flutuante, Strings e Tuplas.

Conjunto (*set*)

- Em Python, uma variável pode ser um conjunto contendo elementos de tipo(s) imutável(áveis), tais como números inteiros e de ponto flutuante, Strings e Tuplas.

- Diferentemente de vetores, conjuntos não têm seus elementos acessados por índice.

Conjunto (*set*)

- Em Python, uma variável pode ser um conjunto contendo elementos de tipo(s) imutável(áveis), tais como números inteiros e de ponto flutuante, Strings e Tuplas.

- Diferentemente de vetores, conjuntos não têm seus elementos acessados por índice.

- No entanto, conjuntos são iteráveis, podendo seus elementos serem acessados por uma estrutura **for**.

Conjunto (*set*)

- Em Python, uma variável pode ser um conjunto contendo elementos de tipo(s) imutável(áveis), tais como números inteiros e de ponto flutuante, Strings e Tuplas.

- Diferentemente de vetores, conjuntos não têm seus elementos acessados por índice.

- No entanto, conjuntos são iteráveis, podendo seus elementos serem acessados por uma estrutura **for**.

- Além disso, um conjunto pode ser escrito diretamente no vídeo via comando **print**.

As funções *set()*, *add()*, *discard()* e *len()*

A função **set()** associa um conjunto vazio a uma variável.

```
escolhidos = set()
print(escolhidos)
```

As funções *set()*, *add()*, *discard()* e *len()*

A função **set()** associa um conjunto vazio a uma variável.

```
escolhidos = set()
print(escolhidos)
```

A função **add()** adiciona um elemento ao conjunto, caso o elemento ainda não ocorra no conjunto.

```
escolhidos = set()
escolhidos.add(13)
print(escolhidos)
```

As funções *set()*, *add()*, *discard()* e *len()*

A função **set()** associa um conjunto vazio a uma variável.

```
escolhidos = set()
print(escolhidos)
```

A função **add()** adiciona um elemento ao conjunto, caso o elemento ainda não ocorra no conjunto.

```
escolhidos = set()
escolhidos.add(13)
print(escolhidos)
```

A função **discard()** retira um elemento do conjunto, caso o elemento esteja no conjunto.

```
escolhidos = {20, 11, 68, 93}
escolhidos.discard(68)
print(escolhidos)
```

As funções *set()*, *add()*, *discard()* e *len()*

A função **set()** associa um conjunto vazio a uma variável.

```
escolhidos = set()
print(escolhidos)
```

A função **add()** adiciona um elemento ao conjunto, caso o elemento ainda não ocorra no conjunto.

```
escolhidos = set()
escolhidos.add(13)
print(escolhidos)
```

A função **discard()** retira um elemento do conjunto, caso o elemento esteja no conjunto.

```
escolhidos = {20, 11, 68, 93}
escolhidos.discard(68)
print(escolhidos)
```

A função **len()** retorna a cardinalidade do conjunto, isto é, seu tamanho.

```
escolhidos = {20, 11, 68, 93}
print(len(escolhidos))
```

Criando um Conjunto de Nomes via Teclado

O programa abaixo faz a leitura de cinco nomes e cria um conjunto com até cinco nomes distintos digitados pelo usuário. A impressão do conteúdo do conjunto ocorre a cada tentativa de inclusão de nome.

```
escolhidos = set()
for i in range(5):
    nome = input("Digite nome: ")
    escolhidos.add(nome)
    print(escolhidos)
```

Criando um Conjunto de Nomes via Teclado

O programa abaixo faz a leitura de cinco nomes e cria um conjunto com até cinco nomes distintos digitados pelo usuário. A impressão do conteúdo do conjunto ocorre a cada tentativa de inclusão de nome.

```
escolhidos = set()
for i in range(5):
    nome = input("Digite nome: ")
    escolhidos.add(nome)
    print(escolhidos)
```

Criando um Conjunto de Nomes Diretamente

```
escolhidos = {"Maria", "Ana", "Giovanna", "Leandro", "Dante"}
print(escolhidos)
```

Operadores para Conjuntos

UNIÃO: **s.union(t)** ou **s | t**

Retorna um novo conjunto resultante da união de dois conjuntos **s** e **t** é formado por todo elemento que pertence a **s** ou que pertence a **t** (ou a ambos).

Operadores para Conjuntos

UNIÃO: **s.union(t)** ou **s | t**

Retorna um novo conjunto resultante da união de dois conjuntos **s** e **t** é formado por todo elemento que pertence a **s** ou que pertence a **t** (ou a ambos).

$$x \in (\textbf{s.union(t)}) \Leftrightarrow x \in \textbf{s} \text{ ou } x \in \textbf{t}$$

Operadores para Conjuntos

UNIÃO: **s.union(t)** ou **s | t**

Retorna um novo conjunto resultante da união de dois conjuntos **s** e **t** é formado por todo elemento que pertence a **s** ou que pertence a **t** (ou a ambos).

$x \in$ (**s.union(t)**) $\Leftrightarrow$ $x \in$ **s** ou $x \in$ **t**

{1, 3, 4}.union({1, 2, 4}) = {1, 2, 3, 4}
{1, 3}.union({2, 4}) = {1, 2, 3, 4}
{‘A’, ‘C’, ‘E’}.union({‘B’, ‘C’, ‘D’} = {‘A’, ‘B’, ‘C’, ‘D’, ‘E’}
{‘C’}.union({‘B’, ‘C’, ‘D’} = {‘B’, ‘C’, ‘D’}

Operadores para Conjuntos

INTERSEÇÃO: **s.intersection(t)** ou **s & t**

Retorna um novo conjunto resultante da interseção de dois conjuntos **s** e **t** é formado por todo elemento que pertence a **s** e que pertence a **t**.

Operadores para Conjuntos

INTERSEÇÃO: **s.intersection(t)** ou **s & t**

Retorna um novo conjunto resultante da interseção de dois conjuntos **s** e **t** é formado por todo elemento que pertence a **s** e que pertence a **t**.

$$x \in s.intersection(t) \Leftrightarrow x \in \mathbf{s} \text{ e } x \in \mathbf{t}$$

Operadores para Conjuntos

INTERSEÇÃO: **s.intersection(t)** ou **s & t**

Retorna um novo conjunto resultante da interseção de dois conjuntos **s** e **t** é formado por todo elemento que pertence a **s** e que pertence a **t**.

$x \in s.intersection(t) \Leftrightarrow x \in \mathbf{s}$ e $x \in \mathbf{t}$

{1, 3, 4}.intersection({1, 2, 4}) = {1, 4}
{1, 3}.intersection({2, 4}) = { }
{‘A’, ‘C’, ‘E’ }.intersection({‘B’, ‘C’, ‘D’}) = {‘C’}
{‘C’ }.intersection({‘B’, ‘C’, ‘D’}) = {‘C’}

Operadores para Conjuntos

DIFERENÇA: **s.difference(t)** ou **s - t**

Retorna um novo conjunto resultante da diferença entre dois conjuntos **s** e **t**. O resultado é formado por todo elemento que pertence a **s** e que não pertence a **t**.

Operadores para Conjuntos

DIFERENÇA: **s.difference(t)** ou **s - t**

Retorna um novo conjunto resultante da diferença entre dois conjuntos **s** e **t**. O resultado é formado por todo elemento que pertence a **s** e que não pertence a **t**.

$$x \in (\mathbf{s}.\text{difference}(\mathbf{t})) \Leftrightarrow x \in \mathbf{s} \text{ e } x \notin \mathbf{t}$$

Operadores para Conjuntos

DIFERENÇA: **s.difference(t)** ou **s - t**

Retorna um novo conjunto resultante da diferença entre dois conjuntos **s** e **t**. O resultado é formado por todo elemento que pertence a **s** e que não pertence a **t**.

$x \in (\mathbf{s}.\text{difference}(\mathbf{t})) \Leftrightarrow x \in \mathbf{s}$ e $x \notin \mathbf{t}$

{1, 3, 4}.difference({1, 2, 4}) = {3}
{1, 3, 4} - {1, 2, 4} = {3}
{1, 3}.difference({2, 4}) = {1, 3}
{‘A’, ‘C’, ‘E’}.difference({‘B’, ‘C’, ‘D’}) = {‘A’, ‘E’}
{‘C’}.difference({‘B’, ‘C’, ‘D’}) = { }

Operadores para Conjuntos

Exemplo: Utilizando os operadores que sobre conjuntos, declare variáveis do tipo conjunto para representarem: um ano, as férias de fim de ano, as férias de meio de ano, todas as férias e o período letivo de um ano.

Operadores para Conjuntos

Exemplo: Utilizando os operadores que sobre conjuntos, declare variáveis do tipo conjunto para representarem: um ano, as férias de fim de ano, as férias de meio de ano, todas as férias e o período letivo de um ano.

```
ano = {“jan”,“fev”,“mar”,“abr”,“mai”,“jun”,“jul”,“ago”,“set”,“out”,“nov”,“dez”}

feriasFimAno = {“jan”, “fev”, “dez”}

feriasMeioAno = {“jul”}

ferias = feriasFimAno.union(feriasMeioAno)

periodoLetivo = ano.difference(ferias)
```

Operadores Relacionais para Conjuntos

IGUAL (**==**) e **DIFERENTE** (**!=**):

Sejam **s** e **t** dois conjuntos:

s == **t** é verdadeiro ⇔ **s** e **t** contêm os mesmos elementos;

s != **t** é verdadeiro, em caso contrário.

Operadores Relacionais para Conjuntos

IGUAL (**==**) e **DIFERENTE** (**!=**):

Sejam **s** e **t** dois conjuntos:

s == **t** é verdadeiro ⇔ **s** e **t** contêm os mesmos elementos;

s != **t** é verdadeiro, em caso contrário.

{1, 3} == {1, 3} é verdadeiro	{1, 3} != {1, 3} é falso
{1, 3} == {3, 1} é verdadeiro	{1, 3} != {3, 1} é falso
{1, 3} == {1, 2} é falso	{1, 3} != {1, 2} é verdadeiro
{1, 3} == { } é falso	{1, 3} != { } é verdadeiro

Operadores Relacionais para Conjuntos

CONTÉM (**>=** ou **issubset**) e
ESTÁ CONTIDO (**<=** ou **issuperset**):

Sejam **s** e **t** dois conjuntos:

s <= **t** é verdadeiro ⇔ todo elemento de **s** está em **t**.

s >= **t** é verdadeiro ⇔ todo elemento de **t** está em **s**.

Operadores Relacionais para Conjuntos

CONTÉM (**>=** ou **issubset**) e
ESTÁ CONTIDO (**<=** ou **issuperset**):

Sejam **s** e **t** dois conjuntos:

s <= **t** é verdadeiro ⇔ todo elemento de **s** está em **t**.

s >= **t** é verdadeiro ⇔ todo elemento de **t** está em **s**.

{1, 3} <= {1, 2, 3, 4} é verdadeiro	{1, 3} >= {1, 2, 3, 4} é falso
{1, 3} <= {1, 3} é verdadeiro	{1, 3} >= {1, 3} é verdadeiro
{ } <= {1, 3} é verdadeiro	{ } >= {1, 3} é falso
{1, 2, 3, 4} <= {1, 3} é falso	{1, 2, 3, 4} >= {1, 3} é verdadeiro
{1, 3} <= { } é falso	{1, 3} >= { } é verdadeiro

Operadores Relacionais para Conjuntos

PERTINÊNCIA (**in**):

Seja **s** um conjunto.

Seja **x** um elemento imutável.

Operadores Relacionais para Conjuntos

PERTINÊNCIA (**in**):

Seja **s** um conjunto.

Seja **x** um elemento imutável.

x in s é verdadeiro ⇔ **x** é um elemento de **s**.

3 in {1, 2, 3, 4} é verdadeiro
5 in {1, 2, 3, 4} é falso
1 in { } é falso

Exemplo: Este programa imprime o número de vogais, e dígitos existentes em uma frase.

```
# Subprograma
def contaVogaisDigitos (frase):
    vogais = {“A”, “E”, “I”, “O”, “U”, “a”, “e”, “i”, “o”, “u”}
    digitos = {“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”}
    nVogais = 0
    nDigitos = 0
    for letra in frase:
        if letra in vogais:
            nVogais += 1
        elif letra in digitos:
            nDigitos+= 1
    print(“Quantidade de Vogais:”, nVogais)
    print(“Quantidade de Dígitos:”, nDigitos)
    return None
# Programa Principal
lida = input(“Diga a frase: ”)
contaVogaisDigitos(lida)
```

Exemplo: Este programa imprime o número de vogais, e dígitos existentes em uma frase.

```
# Subprograma
def contaVogaisDigitos (frase):
    vogais = {“A”, “E”, “I”, “O”, “U”, “a”, “e”, “i”, “o”, “u”}
    digitos = {“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”}
    nVogais = 0
    nDigitos = 0
    for letra in frase:
        if letra in vogais:
            nVogais += 1
        elif letra in digitos:
            nDigitos+= 1
    print(“Quantidade de Vogais:”, nVogais)
    print(“Quantidade de Dígitos:”, nDigitos)
    return None
# Programa Principal
lida = input(“Diga a frase: ”)
contaVogaisDigitos(lida)
```

Exemplo: Este programa imprime o número de vogais, e dígitos existentes em uma frase.

```
# Subprograma
def contaVogaisDigitos (frase):
    vogais = {“A”, “E”, “I”, “O”, “U”, “a”, “e”, “i”, “o”, “u”}
    digitos = {“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”}
    nVogais = 0
    nDigitos = 0
    for letra in frase:
        if letra in vogais:
            nVogais += 1
        elif letra in digitos:
            nDigitos+= 1
    print(“Quantidade de Vogais:”, nVogais)
    print(“Quantidade de Dígitos:”, nDigitos)
    return None
# Programa Principal
lida = input(“Diga a frase: ”)
contaVogaisDigitos(lida)
```

Exemplo: Este programa imprime o número de vogais, e dígitos existentes em uma frase.

```
# Subprograma
def contaVogaisDigitos (frase):
    vogais = {"A", "E", "I", "O", "U", "a", "e", "i", "o", "u"}
    digitos = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9"}
    nVogais = 0
    nDigitos = 0
    for letra in frase:
        if letra in vogais:
            nVogais += 1
        elif letra in digitos:
            nDigitos+= 1
    print("Quantidade de Vogais:", nVogais)
    print("Quantidade de Dígitos:", nDigitos)
    return None
# Programa Principal
lida = input("Diga a frase: ")
contaVogaisDigitos(lida)
```

Exemplo: Este programa imprime o número de vogais, e dígitos existentes em uma frase.

```
# Subprograma
def contaVogaisDigitos (frase):
    vogais = {“A”, “E”, “I”, “O”, “U”, “a”, “e”, “i”, “o”, “u”}
    digitos = {“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”}
    nVogais = 0
    nDigitos = 0
    for letra in frase:
        if letra in vogais:
            nVogais += 1
        elif letra in digitos:
            nDigitos+= 1
    print(“Quantidade de Vogais:”, nVogais)
    print(“Quantidade de Dígitos:”, nDigitos)
    return None
# Programa Principal
lida = input(“Diga a frase: ”)
contaVogaisDigitos(lida)
```

Exemplo: Este programa gera e imprime os números primos entre 2 e N, escolhido pelo usuário, usando o algoritmo chamado de "**Crivo de Eratóstenes**".

```
# Subprogramas
def imprime(num, osPrimos):
    print("Primos entre 2 e", num, ":")
    for candidato in range(2, num+1):
        if candidato in osPrimos:
            print(candidato, end="  ")
    print()
    return None
def eratostenes(num):
    resposta = set()
    ...
    return resposta
# Programa Principal - Crivo de Eratostenes
n = int(input("Diga o valor: "))
primos = eratostenes(n)
imprime(n, primos)
```

Exemplo: Este programa gera e imprime os números primos entre 2 e N, escolhido pelo usuário, usando o algoritmo chamado de "**Crivo de Eratóstenes**".

```
# Subprogramas
def imprime(num, osPrimos):
    print("Primos entre 2 e", num, ":")
    for candidato in range(2, num+1):
        if candidato in osPrimos:
            print(candidato, end="  ")
    print()
    return None
def eratostenes(num):
    resposta = set()
    ...
    return resposta
# Programa Principal - Crivo de Eratostenes
n = int(input("Diga o valor: "))
primos = eratostenes(n)
imprime(n, primos)
```

Exemplo: Este programa gera e imprime os números primos entre 2 e N, escolhido pelo usuário, usando o algoritmo chamado de "**Crivo de Eratóstenes**".

```
# Subprogramas
def imprime(num, osPrimos):
    print("Primos entre 2 e", num, ":")
    for candidato in range(2, num+1):
        if candidato in osPrimos:
            print(candidato, end="  ")
    print()
    return None
def eratostenes(num):
    resposta = set()
    ...
    return resposta
# Programa Principal - Crivo de Eratostenes
n = int(input("Diga o valor: "))
primos = eratostenes(n)
imprime(n, primos)
```

Exemplo: Este programa gera e imprime os números primos entre 2 e N, escolhido pelo usuário, usando o algoritmo chamado de "**Crivo de Eratóstenes**".

```
# Subprogramas
def imprime(num, osPrimos):
    print("Primos entre 2 e", num, ":")
    for candidato in range(2, num+1):
        if candidato in osPrimos:
            print(candidato, end="  ")
    print()
    return None
def eratostenes(num):
    resposta = set()
    ...
    return resposta
# Programa Principal - Crivo de Eratostenes
n = int(input("Diga o valor: "))
primos = eratostenes(n)
imprime(n, primos)
```

```
# Subprogramas
def imprime(num, osPrimos):
def eratostenes(num):
    resposta = set()          # inicializa resposta
    vazio = set()             # inicializa conjunto vazio
    crivo = set(range(2, num+1)) # constrói conjunto de 2 a num
    prox = 2
    while crivo != vazio:
        while not (prox in crivo):
            prox += 1
        resposta.add(prox) # ou resposta = resposta | {prox}
        j = prox
        while j <= num:
            crivo.discard(j)  # ou crivo = crivo – {j}
            j += prox
    return resposta
# Programa Principal - Crivo de Erastóstenes
n = int(input("Diga o valor: "))
primos = eratostenes(n)
imprime(n, primos)
```

```
# Subprogramas
def imprime(num, osPrimos):
def eratostenes(num):
    resposta = set()          # inicializa resposta
    vazio = set()             # inicializa conjunto vazio
    crivo = set(range(2, num+1)) # constrói conjunto de 2 a num
    prox = 2
    while crivo != vazio:
        while not (prox in crivo):
            prox += 1
        resposta.add(prox) # ou resposta = resposta | {prox}
        j = prox
        while j <= num:
            crivo.discard(j)  # ou crivo = crivo – {j}
            j += prox
    return resposta
# Programa Principal - Crivo de Erastóstenes
n = int(input("Diga o valor: "))
primos = eratostenes(n)
imprime(n, primos)
```

```
# Subprogramas
def imprime(num, osPrimos):
def eratostenes(num):
    resposta = set()          # inicializa resposta
    vazio = set()             # inicializa conjunto vazio
    crivo = set(range(2, num+1)) # constrói conjunto de 2 a num
    prox = 2
    while crivo != vazio:
        while not (prox in crivo):
            prox += 1
        resposta.add(prox) # ou resposta = resposta | {prox}
        j = prox
        while j <= num:
            crivo.discard(j)  # ou crivo = crivo – {j}
            j += prox
    return resposta
# Programa Principal - Crivo de Erastóstenes
n = int(input("Diga o valor: "))
primos = eratostenes(n)
imprime(n, primos)
```

Este subprograma imprime o conjunto de características comuns a todos os indivíduos pertencentes a um subconjunto de uma determinada população. Cada indivíduo da população está associado a um identificador (0..MaxPop-1) e possui um conjunto de características.

Este subprograma imprime o conjunto de características comuns a todos os indivíduos pertencentes a um subconjunto de uma determinada população. Cada indivíduo da população está associado a um identificador (0..MaxPop-1) e possui um conjunto de características.

```
caracteristicas = {“esporte”, “tv”, “cinema”, “livro”, “jornal”, “teatro”, “musica”}

def perfilComum(habitantes, caracteristicas, grupo):
    comuns = caracteristicas
    for ident in range(len(habitantes)):
        if ident in grupo:
            comuns = comuns & habitantes[ident]        # ou intersection
    print(“As características em comum são:”)
    for c in caracteristicas:
        if c in comuns:
            print(c, end=“ ”)
    print()
    return None
```

```
# Subprograma

def perfilComum(habitantes, caracteristicas, grupo):
    comuns = caracteristicas
    for ident in range(len(habitantes)):
        if ident in grupo:
            comuns = comuns & habitantes[ident]
    print("As características em comum são:")
    for c in caracteristicas:
        if c in comuns:
            print(c, end=" ")
    print()
    return None

# Programa Principal

caracteristicas = {"esporte", "tv", "cinema", "livro", "jornal", "teatro", "musica"}
alunos = [{"tv", "cinema","livro"}, {"cinema", "musica"}, {"cinema", "tv", "teatro"}]
perfilComum(alunos, caracteristicas, {2, 0})
```

 # Subprograma

```
def perfilComum(habitantes, caracteristicas, grupo):
    comuns = caracteristicas
    for ident in range(len(habitantes)):
        if ident in grupo:
            comuns = comuns & habitantes[ident]
    print("As características em comum são:")
    for c in caracteristicas:
        if c in comuns:
            print(c, end=" ")
    print()
    return None
```

 # Programa Principal

```
caracteristicas = {"esporte", "tv", "cinema", "livro", "jornal", "teatro", "musica"}
alunos = [{"tv", "cinema","livro"}, {"cinema", "musica"}, {"cinema", "tv", "teatro"}]
perfilComum(alunos, caracteristicas, {2, 0})
```

Faça os Exercícios Relacionados a essa Aula

Clique no botão para visualizar os enunciados:

Aula 10

Professores:

Dante Corbucci Filho
Leandro A. F. Fernandes

Conteúdo:

- Estrutura de Dados
  - Conjunto (set)