

Profa. Dra. Raquel C. de Melo-Minardi  
Departamento de Ciência da Computação  
Instituto de Ciências Exatas  
Universidade Federal de Minas Gerais

---



# MÓDULO 2 – PROGRAMAÇÃO

## Sequências – Conjuntos

# TIPOS DE VARIÁVEIS OU ESTRUTURAS DE DADOS EM PYTHON

- ▶ As **sequências** podem ser de dois tipos:
  - ▶ **sequências imutáveis**: são objetos ordenados e finitos
    - ▶ **strings**: cadeias de caracteres
    - ▶ **tuples**: dois ou mais elementos de qualquer tipo dentro de parênteses e separados por vírgula
  - ▶ **sequências mutáveis**
    - ▶ **lists**: conhecidas em outras linguagens como vetores ou arranjos
    - ▶ **sets**: coleções não ordenadas e não indexadas escritas entre chaves

# CONJUNTOS

- ▶ Um conjunto é uma coleção não ordenada e não indexada
- ▶ No Python, os conjuntos são escritos com chaves

```
nucleotideos = {'A', 'C', 'G', 'T'}
```

- ▶ Os conjuntos não são ordenados, portanto, você não pode ter certeza da ordem na qual os itens serão exibidos

# OPERADORES PARA MANIPULAÇÃO DE CONJUNTOS

- ▶ Há alguns operadores que podem ser usados para operar em objetos do tipo conjunto

# VERIFICAÇÃO DE PERTENCIMENTO EM CONJUNTOS

- ▶ O operador `in` pode ser usado para retornar se um determinado elemento existe (pertence a) em um conjunto:

```
'A' in nucleotideos # Retornará True
```

## TROCA DE ELEMENTOS EM CONJUNTOS

- ▶ Depois que um conjunto é criado, você não pode alterar seus itens
  - ▶ Você pode adicionar novos itens

## ADICIONANDO ELEMENTOS EM CONJUNTOS

- ▶ Adicione um item a um conjunto, usando o método `add()`

```
nucleotideos.add('X')
```

- ▶ Também é possível adicionar vários elementos de uma só vez usando o método `update()` e passando uma lista como argumento

```
nucleotideos.update(['A', 'B', 'C'])
```

# SOMA E MULTIPLICAÇÃO (CONCATENAÇÃO) DE CONJUNTOS

- ▶ Existem várias maneiras de associar dois ou mais conjuntos no Python
- ▶ Você pode usar o método `union()` que retorna um novo conjunto contendo todos os itens de ambos os conjuntos

```
nucleotideosM = {'A', 'C', 'G', 'T'}  
nucleotideosm = {'a', 'c', 'g', 't'}  
nucleotideos = nucleotideosM.union(nucleotideosm)
```

- ▶ ou o método `update()` que insere todos os itens de um conjunto em outro:

```
nucleotideosM = {'A', 'C', 'G', 'T'}  
nucleotideosm = {'a', 'c', 'g', 't'}  
nucleotideosM.update(nucleotideosm)
```

- ▶ Itens duplicados são sempre unificados



# OPERAÇÕES TÍPICAS DE CONJUNTOS

- ▶ É possível:
  - ▶ identificar a **diferença** entre conjuntos: `difference()`
  - ▶ verificar se dois conjuntos possuem **interseção**: `intersection()`
  - ▶ verificar se dois conjuntos são **disjuntos**: `isdisjoint()`
  - ▶ verificar se um conjunto é **subconjunto** de outro: `issubset()`
  - ▶ verificar se um conjunto é **superconjunto** de outro: `issuperset()`
- ▶ Entre outras funções

## REMOÇÃO DE ITENS DE UM CONJUNTO

- ▶ Os métodos `remove()`, `discard()` e `pop()` removem um elemento particular de um conjunto

```
nucleotideos = {'A', 'C', 'G', 'T'}  
nucleotideos.remove('A')
```

- ▶ A diferença é que o método `remove()` gera erro se o elemento não existe no conjunto e o `discard()` não levanta esse erro
- ▶ O método `pop()` remove e retorna o último elemento do conjunto
  - ▶ Como um conjunto não é ordenado, não se tem como prever que elemento será retornado

## LIMPEZA OU ESVAZIAMENTO DE UM CONJUNTO

- ▶ Os métodos `clear()` esvazia um conjunto

```
nucleotideos.clear()
```

## TAMANHO DE UM CONJUNTO

- ▶ O método `len()` nativo de Python retorna o tamanho de um conjunto:

```
len(conjunto)
```

# SÍNTESE

	String	Tupla	Lista	Conjunto
Criar	<code>s = ''</code>	<code>t = ()</code>	<code>l = []</code>	<code>c = {'t'}</code>
Obter parte	<code>s[6:9]</code>	<code>t[6:9]</code>	<code>l[6:9]</code>	-
Substituir	<code>s.replace('AAA', 'CCC')</code>	-	-	-
Contar	<code>s.count('AAA')</code>	<code>t.count('AAA')</code>	<code>l.count('AAA')</code>	-
Encontrar	<code>s.find('AAA')</code>	<code>t.index('AAA')</code>	<code>l.index('AAA')</code>	-
Verificar pertencimento	<code>'AAA' in s</code>	<code>'AAA' in t</code>	<code>'AAA' in l</code>	<code>'AAA' in c</code>
Quebrar	<code>s.split('A')</code>	-	-	-