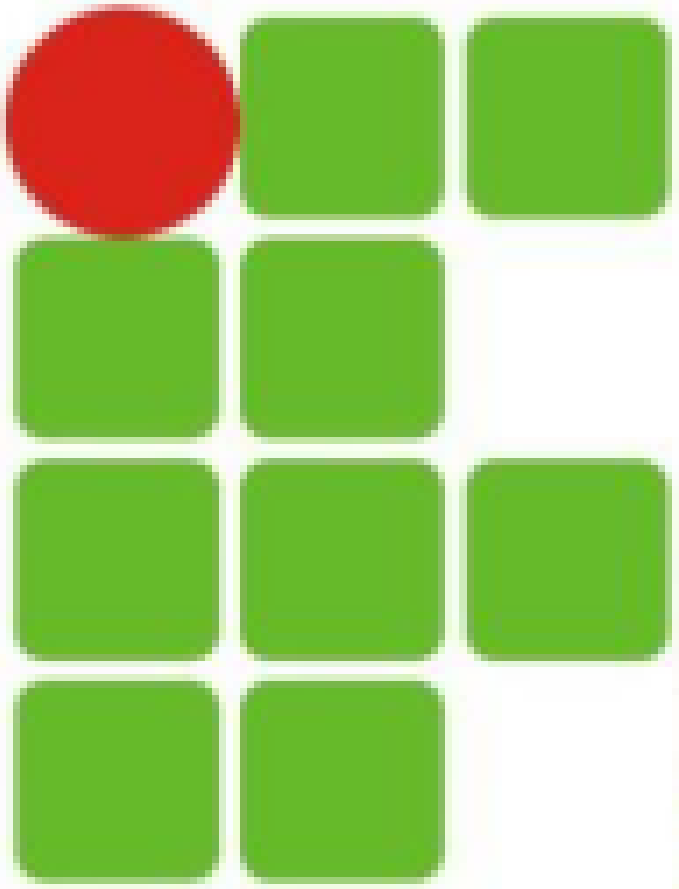
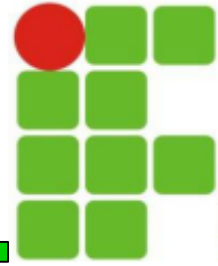


Python: Cadeias de caracteres (Strings)

Galileu Batista de Sousa

Galileu.batista -at +ifrn -edu +br

Fundamentos de strings



- Strings são cadeias de caracteres.
- Podem ser expressos de três maneiras:
  - Strings de uma só linha

```
nome = "Giovanna Camila"
print (nome)
```

–

```
nome = 'Giovanna Camila'
print (nome)
```

  - Strings de múltiplas linhas

```
nome = '''Giovanna
          Camila'''
print (nome)
```

Operações básicas

- Concatenação (+)
  - Não se pode concatenar *strings* com inteiros/float

```
filho1 = 'Tiago' + 'Batista'
print (filho1)
```

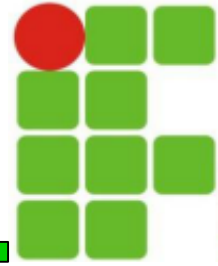
–

- Replicação (*)
  - Não se pode replicar *strings* com *strings*

```
som = "Toc" * 3
print (som)
```

```
filha3 = "Giova" + "n"*2 + "a"
print (filha3)
```

Operações básicas

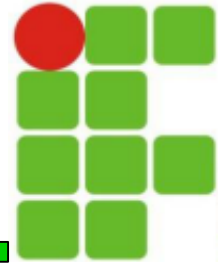


- Comparação de strings
  - Comparação lexicográfica
  - Usa os operadores já conhecidos

```
== !=
<  <=
>  >=
```

```
nome = input("nome?")
if nome == "Gio":
      print ("Seu nome é lindo.")
if nome < "Gio":
      print ("Seu nome vem antes de 'Gio'")
if nome > "Gio":
      print ("Seu nome vem depois de 'Gio'")
```

Conversões de tipos



- Para string – str(...):

```
valor_f   = 25.2
valor_str = str (valor_f)
print ("O valor é "+valor_str)
```

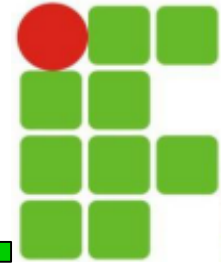
- Para inteiro – int(...)

```
valor_i = int ("25")
valor_f  = float ("25.6")
print (valor_i, valor_f)
```

- Para real – float(...)

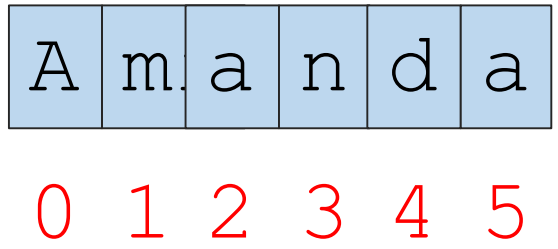
```
valor = int ("25", 8)
print (valor)
```

Acesso a elementos de strings



- É possível obter cada caractere de um string
  - Operação de indexação

[ ]



- Os índices iniciam em zero e vão até é o tamanho do string – 1
  - Índices podem ser expressões
  - Índice fora de limites gera erro

```
nome = "Amanda"
inicial = nome[0]
print (inicial)
```

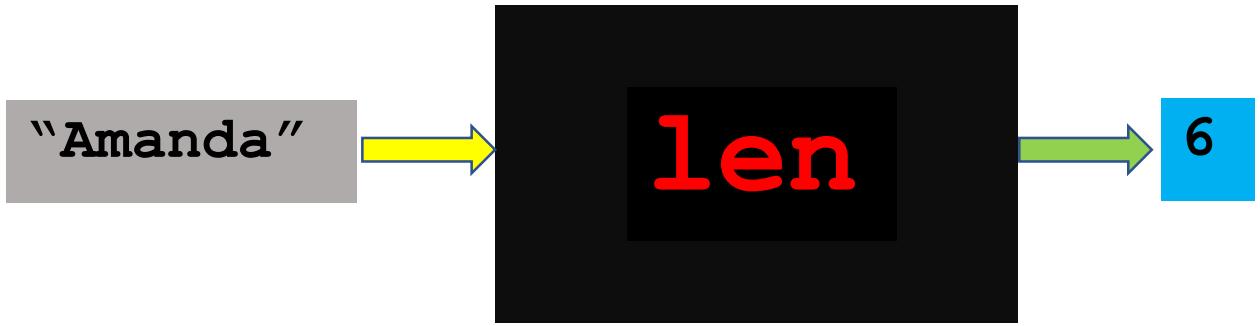
Tamanho de strings

- Há uma função predefinida que retorna o tamanho de uma string

len(...)

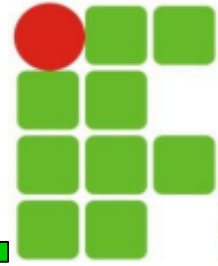
A	m	a	n	d	a
0	1	2	3	4	5

- Uma função é um código que alguém já escreveu. Sorte sua.




```
nome = "Amanda"
print (len(nome))
```

Acesso a todo o string



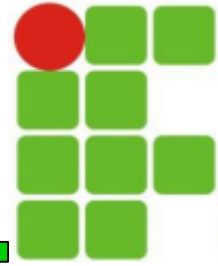
```
nome = input ('Digite seu nome: ')
ind = 0
While ind < len(nome):
    letra = nome[ind]
    print (letra, end='-')
    ind = ind + 1
```

Jane

0	J	J-
1	a	J-a-
2	n	J-a-n-
3	e	J-a-n-e-

Dá pra melhorar, né?

Navegação em strings com for

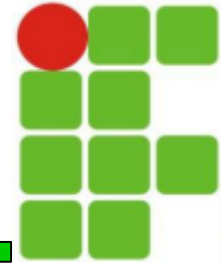


- Strings são conjuntos de dados ...
  - Uso de for é adequado e suportado

```
nome = input ('Digite seu nome: ')
for letra in nome:
    print (letra, end='-')
```

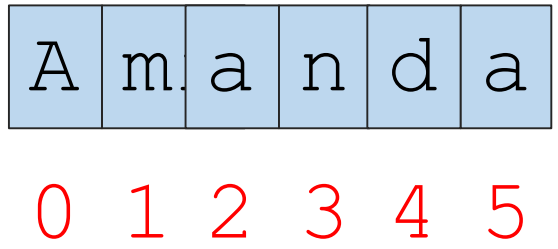
Em cada repetição do for letra recebe um novo caractere de nome

Acesso a elementos de strings



- É possível obter cada caractere de um string
  - Operação de indexação

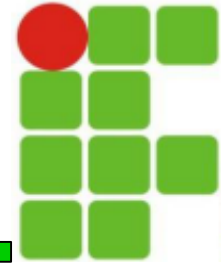
[ ]



- Os índices iniciam em zero e vão até é o tamanho do string – 1
  - Índices podem ser expressões
  - Índice fora de limites gera erro

```
nome = "Amanda"
inicial = nome[0]
print (inicial)
```

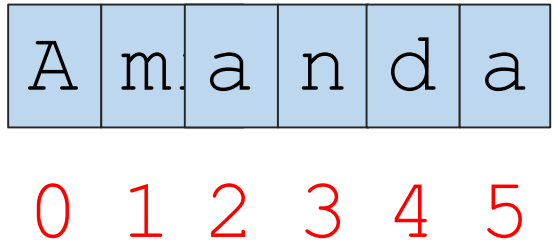
Substrings (*slicing*)



- A indexação retorna um caractere da string
- O slicing retorna um pedaço da string (substring)

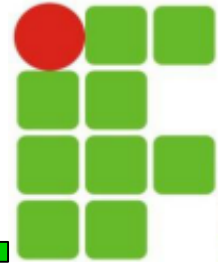
[:]

- Informa-se o índice inicial e o final (não incluído)
  - Podem ser expressões




```
nome = "Amanda"
ela  = nome[0:3]
print (ela)
```

Lembra de *range*?



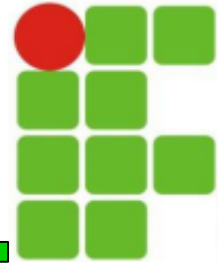
- Tecnicamente trata-se de um ***iterator***

- O *iterator* mais simples é **range**
  - Gera um conjunto de números inteiros
  - Especifica-se inicio, fim, incremento

Ideias similares para slicing

- Exemplos de *range*:
  - range (100) - gera números [0, 100 [ou *[0, 99]*
  - range (60, 100) - gera números [60, 100 [ou *[60, 99]*
  - range (3, 11, 2) - gera números (3, 5, 7, 9)

Substrings (*slicing*)



- É possível omitir o índice inicial – usa **zero**

```
nome = "Amanda"
ela  = nome[:3]
print (ela)
```

→ "Ama"

- É possível omitir o índice final – usa o **len**

```
nome = "Amanda"
ela  = nome[2:]
print (ela)
```

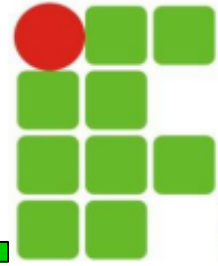
→ "anda"

- É possível informar um passo

```
nome = "Amanda"
ela  = nome[::2]
print (ela)
```

→ ?

Biblioteca para *Strings*



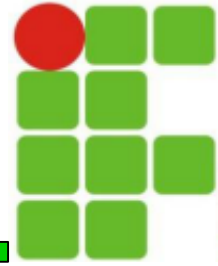
- Há um conjunto de funções (métodos) sobre strings
  - Facilita muito o trabalho

- Antes de mais nada lembre-se:
  - **Strings são read-only**.
  - As funções não modificam strings

```
greet = "Ola"
lgreet = greet.lower()
print (lgreet)
print (greet)
```

Exatamente!!! strings são read-only.

Quais as funções/métodos?



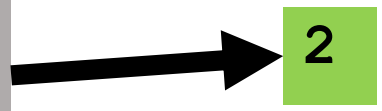
```
>>> dir (str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode',
'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index',
'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier',
'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper',
'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace',
'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
'upper', 'zfill']
```

O que elas fazem: docs.python.org/3/library/stdtypes.html#string-methods

Busca de substrings

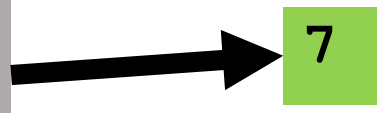
- O método ***find*** **encontra a primeira** ocorrência
  - Retorna o índice (-1, se não encontra)

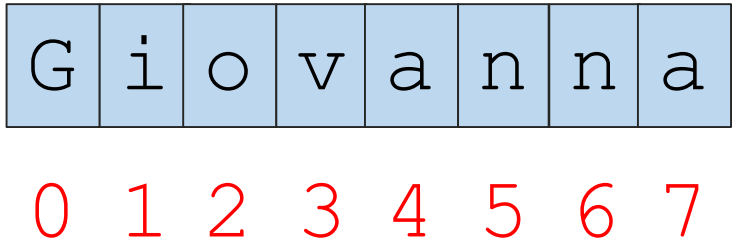
```
nome = "Amanda"
pos_a = nome.find('a')
print (pos_a)
```



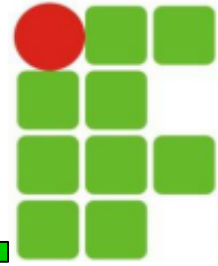
- É possível dizer o índice onde a busca começa

```
nome = "Giovanna"
pos_a = nome.find('a', 5)
print (pos_a)
```





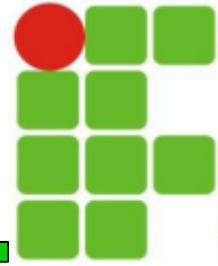
Extração de substrings



- Usa-se ***find*** para encontrar marcadores
- Depois faz-se o *slicing* com a ocorrência final

```
import datetime
now = str(datetime.datetime.now())
# '2020-10-24 22:33:28.652595'
hora_pos = now.find(' ')
hora_fim = now.find(':')
print ("Hora = ", now[hora_pos+1:hora_fim]
```

Troca de substrings

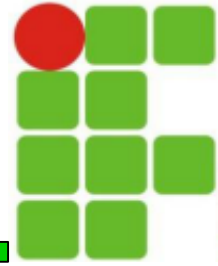


- O método ***replace*** **trocas todas** as ocorrências
  - Retorna o índice (-1, se não encontra)

```
nome  = "Amanda Batista Sousa - Amanda"
irmã  = nome.replace('Amanda', 'Giovanna')
Irmão = nome.replace('Amanda', 'Tiago')
print (nome, irmão, irmã)
```

- Lembre-se: **strings são read-only**

Outros métodos



- ***startswith*** (veja endswith)
  - Verifica se string começa com outro
- ***strip*** (veja lstrip e rstrip)
  - Elimina espaços no início e no final
- ***upper*** (veja lower)
  - Converte para maiúsculos
- ***count***
  - Conta ocorrências de string em outro