Curso Python Avançado

Funções e Controle

olist

Estruturas de Controle

Estruturas de Controle

If

If funciona de maneira semelhante a outras linguagens...

```
>>> x = int(input("Digite um número inteiro: "))
>>> if x < 0:
...     print('Negativo')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Um')
... else:
...     print('Mais')
```

Estruturas de Controle

While

… assim como while.

```
>>> a = 0
>>> while a < 10:
...     print(a)
...     a = a + 1
```

Estruturas de Controle

For

For também é similar, com algumas funcionalidades extras (ex: in)

```
>>> palavras = ['teste', 'de', 'repeticao']
>>> for p in palavras:
...     print(p, len(p))
```

Estruturas de Controle

For

```
>>> palavras = ['teste', 'de', 'repeticao']
>>> for p in palavras[:]:
...     if len(p) > 6:
...         palavras.insert(0, p)
...
>>> palavras
```

Estruturas de Controle

For

```
>>> palavras = ['teste', 'de', 'repeticao']
>>> for i in range(len(palavras)):
...     print(i, palavras[i])
```

Estruturas de Controle

For

É possível usar `else` com `for` ou `while`

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...             print(n, 'igual a', x, '*', n // x)
...             break
...     else:
...         print(n, 'é número primo')
```

Estruturas de Controle

For

E, similar a outras linguagens, continue

```
>>> for num in range(2, 10):
...     if num % 2 == 0:
...         print(num, "É número par")
...         continue
...     print(num, "É um número...")
```

Funções

Funções

Definindo Funções

Funções são definidas usando a palavra-chave `def` seguido do nome da função e a lista de parâmetros separados por vírgula dentro de `()` e finalizada com `:`

```
>>> for num in range(2, 10):
...     if num % 2 == 0:
...         print(num, "É número par")
...         continue
...     print(num, "É um número...")
```

Funções

Retorno de Funções

Como em outras linguagens, é possível retornar objetos.

```
>>> def fib2(n):
...     result = []
...     a, b = 0, 1
...     while a < n:
...         result.append(a)
...         a, b = b, a+b
...     return result
...
>>> f100 = fib2(100)
>>> f100
```

Funções

Parâmetros Default

Parâmetros default são definidos usando `=`.

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)
```

Funções

Parâmetros Default

Parâmetros *default* são "processados" no local onde foram definidos.

```
i = 5
def f(arg=i):
    print(arg)

i = 6
f()
```

Funções

Parâmetros Default

Tome cuidado com objetos mutaveis sendo usados como parâmetros *default*.

```
def f(a, L=[]):
    L.append(a)
    return L

print(f(1))
print(f(2))
print(f(3))
```

Funções

Parâmetros Default

Uma maneira de evitar este comportamento.

```
def f(a, L=None):
    if L is None:
        L = []
    L.append(a)
    return L
```

Funções

Passando Argumentos por Nome

Como comentado em outras aulas, é possível definir qual parâmetro recebe qual argumento usando seu nome.

```
def imprime_nome_completo(nome, sobrenome):
    print(nome, sobrenome)

imprime_nome_completo("Joao", "Silva")
imprime_nome_completo(sobrenome="Silva", nome="Joao")
```

Exercícios de Lista

Exercícios de Lista

Exercício 1

Crie uma função que receba uma lista e retorne uma lista apenas com elementos únicos (remova os duplicados).

Exercícios de Lista

Exercício 1

Crie uma função que receba uma lista e retorne uma lista apenas com elementos únicos (remova os duplicados).

```
def unique_list(l):
  x = []
  for a in l:
    if a not in x:
      x.append(a)
  return x
```

Exercícios de Lista

Exercício 1

Crie uma função que recebe uma string e a retorna ao contrário (Ex: "ABCD123" -> "321DCBA").

Exercícios de Lista

Exercício 1

Crie uma função que recebe uma string e a retorna ao contrário (Ex: "ABCD123" -> "321DCBA").

```
def string_reverse(str1):
    rstr1 = ''
    index = len(str1)
    while index > 0:
        rstr1 += str1[ index - 1 ]
        index = index - 1
    return rstr1
```