INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

TTI103
Lógica de Programação

Aula T11

Recursão

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Primeiro: entender funções

- Funções
  - instrumentos úteis que agrupam um conjunto de instruções para que possam ser executadas mais que uma vez
  - ou seja, permitem não ter que escrever repetidamente o mesmo código
  - são um dos níveis mais básicos de reutilização de código em Python ou qualquer outra linguagem

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Lembra da sintaxe?

```
def nome_função (arg1, arg2, ...):
    ...
    return resultado
```

- Tente manter nomes signifativos, por exemplo, len() é um bom nome para uma função length().
  - Tenha cuidado com os nomes, não chamar uma função com o mesmo nome de uma função interna em Python (como len).
- Dentro do par de parênteses estão vários argumentos separados por uma vírgulas.
  - São as entradas para a função.
- Não se esqueça dos dois pontos.
- Para iniciar o código dentro da função corretamente, não se esqueça de endentar o código.
  - Depois de tudo isso você começa a escrever o código que deseja executar.
  - Algumas funções têm a cláusula return, outras não.

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Exemplos

```
# uma função simples
def say_hello():
    print('hello')

say_hello()


# uma função com parâmetro
def saudacoes (nome):
    print(f'Hello {nome}')

saudacoes('Jose')
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Exemplos

```
# uma função com parâmetros e return
def soma(num1, num2):
    return num1+num2

print(soma(4, 5))

# adicionando lógica às funções
def e_par (num):
    return num % 2 == 0

print(e_par (23))
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Além de funções, vimos listas: verificar se existe algum para na lista

```
def existe_par_na_lista(lista_num):
    # verifica cada número até encontrar
    cont = 0
    for num in lista_num:
        #conta o número de iterações realizadas
        cont = cont + 1
        # assim que encontrar o primeiro, devolve True
        if num % 2 == 0:
            return True, cont
    # se no final não tiver encontrado, retorna False
    return False, cont


# perceba a eficiência da lógica ;-)
res1, res2 = existe_par_na_lista([1, 2, 3, 4, 5, 6])
print(f'iteracoes: {res2}')
print(f'existe: {res1}')
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

E se quisermos obter todos os pares da lista?

```
def pares_na_lista(lista_num):
    pares = [] # variável local
    for num in lista_num:
        if num % 2 == 0:
            pares.append(num) # note que aqui temos um método
    return pares

print (pares_na_lista([1, 2, 3, 4, 5, 6]))

# ouuuu

lista_num = pares_na_lista([1,3,5,7]) # note o escopo da variável
print(lista_num)
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Composição de funções

- É possível chamar uma função de dentro de outra
  - Composição de funções
- Exemplo
  - Escrever uma função que recebe dois pontos, o centro de uma circunferência e um ponto nela, computa a área do círculo e exibe o resultado
- Lógica
  - O centro é armazenado nas variáveis xc e yc.
  - O ponto no perímetro em xp e yp.
  - O primeiro passo é encontrar o raio r do círculo: a distância entre os dois pontos.
  - Calcular a área dada por $A = \pi r^2$

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Composição de funções

```
import math

def distancia(x1, y1, x2, y2):
        dx = x2 - x1
        dy = y2 - y1
        soma_quadrados = dx**2 + dy**2
        dist = math.sqrt(soma_quadrados)
        return dist

def area(raio):
    a = 3.14159 * math.pow(raio, 2)
    return a

def calculos(xc, yc, xp, yp):
    raio = distancia(xc, yc, xp, yp)
    result = area(raio)
    return result

print(calculos(0,0,3,4))
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Recursão

- Recursão significa definir um problema em termos de si mesmo
- Quebra-se um problema complexo em subproblemas menores, até alcançar um problema simples o bastante para ser resolvido trivialmente.
- O uso de algoritmos recursivos permite escrever soluções elegantes para problemas que, de outra forma, seriam muito difíceis de programar.

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Alguns exemplos clássicos

- Fatorial de um número:

- $F(n) = \begin{cases} 1, se\ n = 0\ ou\ n = 1 \\ n * (n-1)!, se\ n > 1 \end{cases}$

- Note que a definição matemática é recursiva, com um caso base
  - perfeito para implementar

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Implementação do fatorial

```
def fatorial_iterativo(n):
    result = 1
    while n>0:
        result = result * n
        n = n - 1
    return result


def fat_rec(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * fat_rec(n-1)
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Os três fundamentos da recursão

- Todo algoritmo recursivo deve obedecer a três leis:
  - O algoritmo recursivo deve possuir um **caso base**.
  - O algoritmo recursivo deve alterar o seu estado de maneira a se aproximar do caso base (**redução do problema**).
  - O algoritmo recursivo deve ter uma **chamada a si mesmo** (direta ou indiretamente).

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Soma dos elementos de uma lista

```
def soma_lista_it (lista):
    soma = 0
    for i in lista:
        soma = soma + i
    return soma


def soma_lista_rec(lista):
    if len(lista) == 1:
        return lista[0]
    else:
        return lista[0] + soma_lista_rec(lista[1:])
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Bora praticar!

1. Calcular o máximo de uma sequência
2. A sequencia de Fibonacci é dada por

- $$Fib(n) = \begin{cases} 1, se\ n = 0\ ou\ n = 1 \\ Fib(n-1) + Fib(n-2), se\ n > 1 \end{cases}$$
  - desenvolver uma função recursiva em Python para calcular a sequência de Fibonacci

3. A função a seguir calcula o maior divisor comum dos inteiros estritamente positivos m e n. Escreva uma função recursiva equivalente.

```
def euclides (m, n):
    r = m % n
    while r != 0:
        m = n
        n = r
        r = m % n
    return n
```

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

TTI103
Lógica de Programação

Aula T11

Recursão