Informações Básicas Sobre Python

- Criada em 1990

- Linguagem Multiparadigma

- Híbrida: compilada – interpretada

- Dinâmica: Verificação de tipos durante

a execução

- Python EXIGE endentação

\_\_\_DEFININDO FUNÇÕES EM Python\_\_\_\_

**def<nome(parâmetro)>:**

- Primeiro o comando def

- Após o nome da função

- Entre parênteses os parâmetros da

função separados por “,”

- Após isso, é necessário os “ : ”

COMANDOS UTEIS DE LISTA EM PythoN

**Definindo lista:**

- lis = [‘ana’,’joao’,’maria’]

- lisn = [1,2,3,4,5]

**Acessando elementos:**

- lis[0] -> 1º elemento

- lis[1] -> 2º elemento

- lis[-1]-> último elemento

**Fatiando a lista**

- lisn[x:y] -> Fatia de x a (y-1)

- lisn[0:3] ou lisn[:3] -> 1ºs 3 elementos

- lisn[1:4] -> Do 1 ao 3 = [2,3,4]

- lisn[1:] -> Do 1 em diante = [2,3,4,5]

- lisn[-3:0] ou lisn[-3:] -> Últimos 3 elem.

Mais alguns Comandos Uteis de lista

- lis.append(‘teste’) -> Add no fim

- lis.insert(‘teste’) -> Add no inicio

- lis.remove(‘teste’) -> Só o 1º

- del lis[0] -> Deletar pela posição

- len(lis) -> Tamanho da lista

mais algumas operações em Python

- [1,2] + [3,4] = [1,2,3,4]

- [1]\*4 = [1,1,1,1]

- 3 in [1,2,3] -> retorna true

- Min/Max([1,2,3]) retorna 1/3

- lisn.sort() # Após declarada a lista

- lisn.reverse() -> Inverte a lisn

\_FUNÇÕES DE ALTA ORDEM EM PYTHON\_\_

**MAP:** Aplica função a cada elemento de uma lista.

def soma10(x): return x + 10

list(map(**soma10**,[0,1,2])) = [10,11,12]

**FILTER:** Aplica função booleana a cada item. Seleciona somente os que retornam true.

def maiorq5(x): return x>5

list(filter(maiorq5,[0,7,1,2])) = [7]

\_FUNÇÕES DE ALTA ORDEM EM PYTHON\_\_

**REDUCE:** Aplica função de 2 argumentos sucessivamente aos itens da lista, retornando um só valor.

>>>import functools

def f(x,y): return x+y

functools.reduce(f,[1,2,3]) = 6

functools.reduce(f, [47,11,42,13]) = 113

\_\_\_\_FUNÇÕES Anônimas em python\_\_\_\_

**LAMBDA:**Funções sem nome, são definidas para servir de argumento para funções de alta ordem ou como resultado das mesmas.

g = lambda x : x\*\*2

g(4) = 16

h = lambda x,y: x+y

h(4,5) = 9

r = lambda x,y,z : 2\*x+y+z

r(1,2,3) = 7

soma10 = lambda x : x+10

list(map(soma10,[0,1,2])) = [10,11,12]

\_\_MAIS FUNções de lista EM Python\_\_

**RANGE:** P.A em python.

range(n) -> Sequência de 0 a n-1

List(range(10)) = [0,1,2,3,4,5...9]

range(start,stop) -> Sequência de start a stop-1

list(range(3,9)) = [3,4,5,6,7,8]

range(start,stop,step) -> Produz sequência com passo step

list(range(1,10,2)) = [1,3,5,7,9]

**MAIS EXEMPLOS:**

List(range(8,-1,-1)) = [8,7,6,5,4,3,2,1,0]

List(range(10,0,-2)) = [10,8,6,4,2]

List(map(lambda n:n\*2, range(10,0,-2))) = [20,16,12,8,4]

\_\_\_LIST comprehension EM Python\_\_\_\_

[2\*x for x in [1,2,3,4]] = [2,4,6,8]

Nomes = [“Joao”,”Maria”,”Ana”,”Jose”]

[len(x) for x in nomes] = [4, 5, 3, 4]

[x[0] for x in nomes] = [‘J’, ‘M’, ‘A’, ‘J’]

[x[-1] for x in nomes] = [‘o’, ‘a’, ‘a’, ‘e’]

\_\_\_LIST comprehension EM Python\_\_\_

**Forma Aninhada:** [Expressão com variáveis for var1 in sequencia1 for var2 in sequencia2]

[(x,y) for x in range(2) for y in range(2)]

[ (0,0), (0,1), (1,0), (1,1) ]

[x+y for x in range(2) for y in range(2) ]

[ 0, 1, 1, 2 ]