**Conteúdo + Prática  
  
 Tipo numérico**   


**Seção 3**   
PEP8: Escrever códigos limpos (Propostas de aprimoramento do python), no caso são propostas de melhoria pra linguagem Python

Dir e help: Funçções da linguagem Python que auxiliam na programação, um lhe oferece todos os métodos disponíveis para aquela determinada variávael e o outro em sequência dá a documentação de cada método disponível.

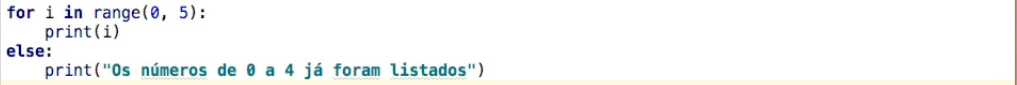
**Seção 4**  
type(algo que queira saber o tipo) : vai retornar o tipo daquela variável

**Condicionais**  
obs: Sendo o código de condicionais escritos em menos linhas e com o detalhe de no bloco de execução pular 4 linhas, que no caso um tab, como também escrito de forma diferente o else if, sendo escrito como elif no python.  
  
**Estrutura Lógicas:** and (e), or (ou), not (não), is(é)  
  
No caso do not e do is funcionam da seguinte forma.  
  
SE isso não(not) não for aquilo  
 execute isso

SE isso for(is) aquilo

execute isso  
  
Prática: condicionais e inicío de estrutura de repetição   
  
 **Estruturas de Repetição**  
Iteração == Repetição

**Algumas maneiras peculiares que podem ser usadas o for em Python:**in == dentro1) A primeira é usando o for interando sobre uma lista, que fica da seguinte maneira:  
****  
obs: Sendo o else usado em python para mostrar um trecho de código quando o for é finalizado.

2) A segunda é fazendo o for repetir por uma determinada quantidade de vezes:   
****

obs: O range é responsável por retornar uma lista de número inteiros identificados no seu parâmetro. É uma função interna do python que retorna sequências numéricas

ex1: range(valordeinicio, valordeparada)

ex2: range(valordeinicio, valordeparada, passo)  
obs: Sendo o passo aquel que define como vai sair do valor inicial até o valor de parada, por padrão é de +1 em+1

while enquanto essa condição for verdadeira:

execute isso;

**Coleções**

**Listas:** Listas em Python é representada como vetores/matrizes/array, com a grande diferença de que em python serem mais **dinâmicos** e também podermos colocar **qualquer tipo de dado**.

O que quer dizer com ser dinâmico ou qualquer tipo de dado?

R: O dinâmico se refere a não ter um tamanho fixo, apenas criar a lista e ir adicionando elementos. Infinito? Talvez, vai ir adicionando enquanto tiver memória disponível  
  
R: Que diz respeito a qualquer tipo de dado, ou seja, pode ser tanto string quanto valores inteiros e decimais.

Listas representadas em python por []

**Funções**

Apesar de ter tudo anotado lá no Vscode, é vom deixar anotado aqui também.

**Funções com retorno:** Como o próprio nome diz, ela se caracteriza por retorna algo no final, sempre lembrando que a função precisa ser chamada pra isso acontecer.

1 - Ela finzaliza a função, ou seja, ela sai da execução da função

2 - Podemos ter em uma função, diferentes retornos

3 - Podemos, em uma função, retornar qualquer tipo de dado e até mesmo múltilos valores

def quadrado\_de\_10():

    return 10 \* 10

print(quadrado\_de\_10())

**Funções com parâmetro:** As funções com parâmetros se caracterizam por ter algo passado no parêntes dela, um valor que vai ser usado dentro da função.

def repeticaomsg(*num1*, *num2*, *msg*):

    return (*num1* + *num2*) \* *msg*

print(repeticaomsg(2, 4, ' Já acabou Jéssica?'))

obs: Tendo uma diferença entre parâmetro e argumento,

**Funções e docstring:** É simplesmente a documentação...

Também é possível ver a documentação de um função usando o \_\_doc\_\_

**Args:** aO parâmetro \*args utilizado em uma função, coloca os valores extras informados como entrada em uma tupla. Então desde já lembre-se que tuplas são imutáveis.

**Kwargs:** Diferentemente do arg o kwargs transforma esses parâmetros extras em um dicionário.

**Comprehensions**

Comprehensions basicamente interage com os elementos dentro de um dicionário, tupla, lista, matriz... dane-se, ele interage com aquela sequência de valores e faz algo com aquilo.

**List Comprehensions:**

# Exemplo 1

numeros = [1, 2, 3, 4, 5]

resposta= [i \* 10 for i in numeros]

print(resposta)

**obs:**

"""

 Para entender sobre o que se trata a comprehensions, é necessário ter uma leitura de trás pra frente.

 Sendo para cada nummero(contador) dentro da lista de número, pegue o contador percorrendo e multiplique

 por cada posição que ele passar aquele elemento por 10

"""

**Listas aninhadas:** São como matrizes em C

**Expressões Lambdas e Funções Integradas**

**Lambdas:** São conhecidas como funções sem nome, ou seja, funções anônimas. Semelhantes a arrowfunction em JS

Funções integradas: São basicamente funções lambdas e coleções que fazem um determinado tipo de tarefa

**Map:** Ele pega uma determinada coleção de dados e através de uma função já criada ele vai adicionando os elementos dessa coleção a retornando o resultado

def area(*r*):

    return 3.14 \* (*r* \*\* 2)

raios = [1, 20, 13, 83, 42, 64, 42]

# Utilizando o map

areas = map(area, raios)

#pegando cada elemento da lista raios e usando lá na função e depois adicioando os resultados em um Map Object

print(list(areas)) # É nessário passar de Map Object pra lista, pode também passar pra dicionário, tuplas etc.

**obs:** Pode passar de MapObject pra um diferente tipo de coleção.

**Filter:** A principal diferença do filter pro map é que no filter ele retorna valores boleanos, no caso... True. Sendo que no map ele retorna apenas os valores passados que serão dados como return.

# Imporatando biblioteca que trabalha com estatísticas

import statistics

dados = [2, 3, 4, 5, 6, 7, 8]

media = statistics.mean(dados) #função mean é responsável por calcular a média dos dados

print(media)

# Utilizando o filter, sendo o filter recebendo dois parâmetros (uma função e um iterável)

resultado = filter(lambda *x*: *x* > media, dados)

"""

O filtro recebe a função lambda com o retorno só dos número que estão acima da media

e logo em seguida passa o iterável, que seria a determinada coleção, ele pega cada dado do

iterável e passa como parâmetro da função lambda

"""

print(list(resultado)) #Sempre transformando em uma lista, sendo que os resultados de mapobj lá

**obs:** Sendo que também é possível passar lambda como uma função já em um dos parâmetros lá no Map, só foi por peculiaridade minha não ter colocado um exemplo semelhante. E parecido com o map no final também tem que converter o resultado pra um tipo de coleção