Server-Side - Python

Católica de Santa Catarina – Centro Universitário
Jaraguá do Sul - SC, Brasil

Professor: MSc. Andrei Carniel (Prof. Andrei)
Contato: andrei.carniel@gmail.com

1. Conceitos

Python

1. Não tem “;” no final da linha.
2. Identação é necessária para definer a hierarquia do código.
3. Usado para:
    1. Desenvolvimento Web;
    2. Desenvolvimento de software;
    3. Matematica;
    4. Inteligência artificial;
    5. Script de sistema;

Criação de variáveis

- Python não tem comando para declarar uma variável.
- Uma variável é criada no momento em que você atribui um valor a ela.

- Exemplo:

```
x = 5
y = "John Snow"
print(x)
print(y)
```

```
x = str(3)    # x será '3'
y = int(3)    # y será 3
z = float(3)  # z será 3.0
```

```
a = 4
A = "Daenerys"
#A não irá sobrescrever a
```

```
x = 5
y = "John Snow"
print(type(x))
print(type(y))
#Exibe o tipo de arquivo
```

Declaração

- Nomes de válidos

```
myvar = "John Snow"
my_var = "John Snow"
_my_var = "John Snow"
myVar = "John Snow"
MYVAR = "John Snow"
myvar2 = "John Snow"
```

- Nomes de inválidos

```
2myvar = "John Snow"
my-var = "John Snow"
my var = "John Snow"
```

- Múltiplas variáveis

```
x, y, z
=  "Laranja", "Banana", "Cereja"
print(x)
print(y)
print(z)
```

```
x = y = z =  "Laranja"
print(x)
print(y)
print(z)
```

```
fruits = ["maçã", "banana", "cereja"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

Tipos de dados

x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["maçã", "banana", "cereja"]	list
x = ("maçã", "banana", "cereja")	tuple
x = range(6)	range
x = {"name" : "John Snow", "age" : 36}	dict
x = {"maçã", "banana", "cereja"}	set
x = frozenset({"maçã", "banana", "cereja"})	frozenset
x = True	bool

Tipos de dados

x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

Para descobrir o tipo de dado…

```
x = 5
print(type(x))
```

Tipos de números

- Tipos

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

- Conversão

```
#converte de int para float:
a = float(x)
#converte de float para int:
b = int(y)
#converte de int para complex:
c = complex(x)
```

```
x = int(1)   # x será 1
y = int(2.8) # y será 2
z = int("3") # z será 3
```

Strings

- São iguais…

```
print("Hello")
print('Hello')
```

- Lembre-se, strings são vetores.

```
a = "Hello, World!"
print(a[1])
```

- Particionando strings. Pega as 5 primeiras, exceto a posição 5.

```
a = "Hello, World!"
print(b[:5])
```

- Particionando strings. Pega os caracteres a partir da posição 2.

```
a = "Hello, World!"
print(b[2:])
```

- Uppercase e lowercase.

```
a = "Hello, World!"
print(a.upper())
print(a.lower())
```

Strings

- Trocar strings

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

- Particionando strings.

```
a = "Hello, World!"
print(a.split(","))
# retura ['Hello', ' World!']
```

- Concatenar:

```
c = a + b
c = a + " " + b
```

- Escape Characters

```
txt = "Python é a melhor linguagem para
\"Websites\"."
```

Escape caracteres

Code	Result
\'	Apas simples
\\	Backslash
\n	Nova linha
\r	Retorno
\t	Tab

Listas

- Declarando listas:

```
lista1 = ["maçã", "banana", "uva"]
lista2 = [1, 5, 7, 9, 3]
lista3 = [True, False, False]
lista4 = ["abc", 34, True, 40, "male"]
```

- Tamanho de uma lista:

```
print(len(lista1))
```

- Acessar um item ou vários da lista:

```
print(lista1[1])
print(lista1[2:5])
```

- Alterando itens:

```
lista1 = ["maça", "banana", "uva"]
lista1[1] = "cereja"
lista1[1:2] = ["cereja", "melão"]
```

- Adicionar itens para uma lista:

```
lista1.append("laranja")
lista1.insert(1, "laranja")
```

- Extendendo uma lista:

```
frutasA = ["maçã", "banana", "cereja"]
frutasB = ["manga", "abacaxi", "mamão"]
frutasA.extend(frutasB)
```

Listas

- Adicionar itens para uma lista:

```
lista1.append("laranja")
lista1.insert(1, "laranja")
```

- Extendendo uma lista:

```
frutasA = ["maçã", "banana", "cereja"]
frutasB = ["manga", "abacaxi", "mamão"]
frutasA.extend(frutasB)
```

- Removendo um item específico:

```
lista1 = ["maçã", "banana", "cereja"]
lista1.remove("banana")
```

- Remove o item da posição…

```
lista1 = ["maçã", "banana", "cereja"]
list1.pop(1)
del list1[0]
```

- Remove o último item:

```
lista1 = ["maçã", "banana", "cereja"]
lista1.pop()
```

- Deleta a lista:

```
lista1.clear()
```

Repetição para listas

- For:

```
lista1 = ["maçã", "banana", "cereja"]
for x in lista1:
  print(x)
```

```
lista1 = ["maçã", "banana", "cereja"]
for i in range(len(lista1)):
  print(lista1[i])
```

- While:

```
lista1 = ["maçã", "banana", "cereja"]
i = 0
while i < len(lista1):
  print(lista1[i])
  i = i + 1
```

Tupla

- Itens de tupla são ordenados, imutáveis e permitem valores duplicados.

```
tupla1 = ("maçã", "banana", "cereja")
tupla2 = (1, 5, 7, 9, 3)
tupla3 = (True, False, False)
tupla4 = ("abc", 34, True, 40, "male")
tupla5 = ("maçã",)
```

- Acessando a tupla da mesma forma que na lista

```
tupla = ("maçã", "banana", "cereja")
print(tupla[1])
```

- Atualizando tupla.

```
x = ("maça", "banana", "cereja")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
```

Diferença entre lista, tupla, se e dicionario

- **Lista** é uma coleção que é ordenada e mutável. Permite membros duplicados.
- **Tupla** é uma coleção ordenada e imutável. Permite membros duplicados.

Variáveis

- Global e local

```
x = "global"

def myfunc():
  x = "local"
  print("Python é " + x)

myfunc()

print("Python é " + x)
```

- Global

```
x = "global"

def myfunc():
  global x
  x = "continua global"

myfunc()

print("Python é " + x)
```

Condição IF

- Igual: `a == b`
- Diferente: `a != b`
- Menor que: `a < b`
- Menor ou igual: `a <= b`
- Maior que: `a > b`
- Maior ou igual: `a >= b`

```
a = 200
b = 33
if b > a:
  print("b é maior que a")
elif a == b:
  print("a e b são iguais")
else:
  print("a é maior que b")
```

- AND
- OR
- NOT

```
if a > b and c > a:
…
if a > b or a > c:
…
if not a > b:
```

Condição IF

- A identação é essencial no Python para o *nested if.*

```
x = 41

if x > 10:
  print("Maior que dez,")
  if x > 20:
    print("e maior que 20!")
  else:
    print("mas não é maior que 20.")
```

- Caso você tenha um IF vazio, é necessário utilizar o Código “pass”. Para evitar erros.

```
if b > a:
  pass
```

While

- Sintaxe é muito similar ao IF.

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

- Temos a possibilidade de usar o ELSE aqui.

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

For

- Exemplos:

```
frutas = ["maçã", "banana", "cereja"]
for x in frutas:
  print(x)
  if x == "banana":
    break
```

- Percorrer uma faixa de números:
  - 2 a 5

```
for x in range(2,6):
  print(x)
```

- Percorrer uma string:

```
for x in "banana":
  print(x)
```

- De 2 a 29, incrementando 3;

```
for x in range(2, 30, 3):
  print(x)
```

- Podemos usar o ELSE também.
  - 0 a 5

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Terminou!")
```

Function

- Uma função é um bloco de código que só é executado quando é chamado. Pode ser executado várias vezes.

- Você pode passar dados, conhecidos como parâmetros, para uma função.
  - Retornar dados como resultado.

```
def minha_funcao():
  print("Teste de chamada de função")

…

my_function()
```

Function

- Função sem retorno:
  - 1 Parâmetro;

```
def imprime_nome(fnome):
  print(fname + " é estudante")

imprime_nome("Andrei")
```

- Função sem retorno:
  - + de 1 Parâmetro;

```
def imprime_nome(fnome, fsobrenome):
  print(fname + " " + " é estudante")

imprime_nome("Andrei", "Carniel")
```

- Função com retorno.

```
def quadrado(numero):
  return numero * numero

…

print("O quadrado de 8 é " + str(quadrado(8)))
```

Atividade

- Crie uma função onde:
  - Você manda dois parâmetros, um número e um divisor.
  - Retorne o resultado da parte inteira.
  - Retorne o resto.

- Como faria?

- 5 min…

Resposta…

- Python pode retornar mais de um valor

```
def divisao(numero, divisor):
    resultado = numero / divisor
    resto = numero % divisor
    return resultado, resto

...

rsl, rst = divisao(5, 2)
print("Resultado: " + str(int(rsl)) + " . Resto: " + str(rst))
```

Atividades

- Pode ser necessário utilizar o código:
  - Esse Código recebe uma entrada no terminal e faz o *cast* (conversão) para um inteiro.

```
x = int(input())
```

- Será enviado na atividade do teams…

- Crie funções para resolver os seguintes problemas:

Atividades

1. Implemente um programa em Python que receba um número e exiba a tabuada do número.
    - Aceitar somente números inteiros positivos;
    - -1 encerra o programa.
2. Desenvolva um programa que armazene três notas em uma lista. A lista será percorrida por uma ÚNICA função que devolverá: a média final, a maior nota, a menor nota e se foi aprovado ou não.
    - Exibir em tela os dados;
    - Considerando 6.0 ou mais como aprovado.
3. Desenvolva um programa que receba números inteiros, apresente a soma dos valores pares e a média dos valores ímpares.
    - O programa deve contar a quantidade de números pares e ímpares;
    - Não há limite para a quantidade de números;
    - Zero finaliza o recebimento de números.