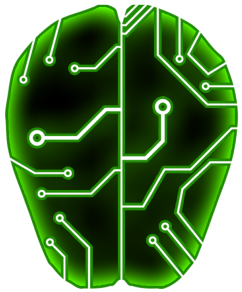
Curso de Python: Parte 1

Conceitos Básicos



COMPUTER SOCIETY

IFB

Guilherme Paiva

Instituto Federal de Brasília, Campus Taguatinga

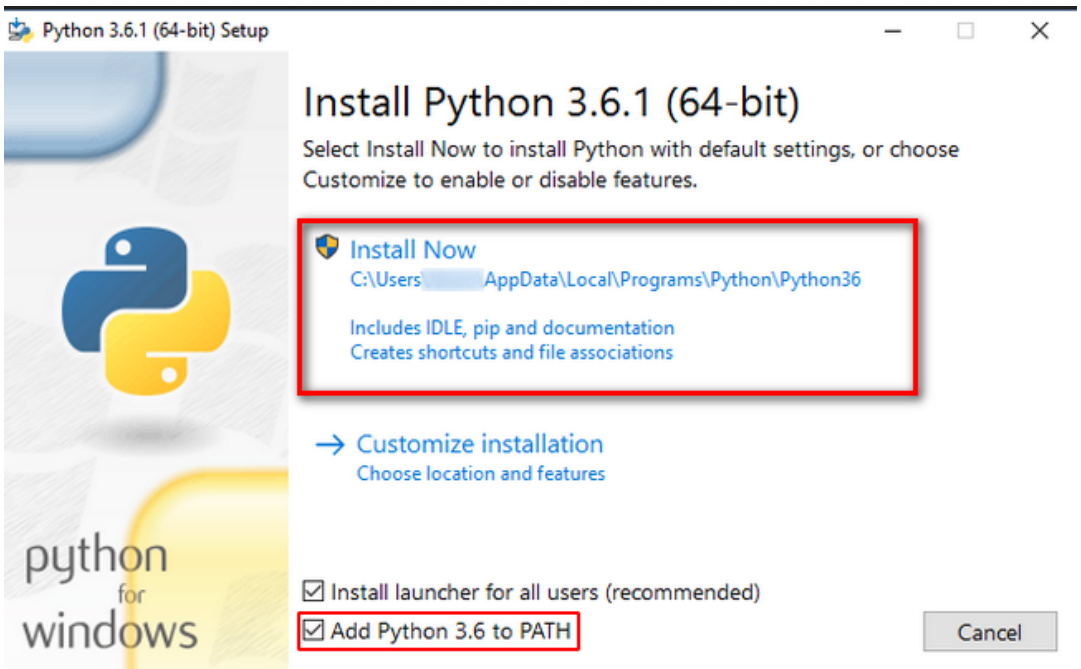
Sumário

História e Curiosidades

- Guido van Rossum, 1991
- Tendência

Preparação

- Linux
  - Já está instalado!
- Windows
  - Download em https://www.python.org/downloads/
  - Ao instalar selecione “add to PATH”



IDEs e Editores de Texto

- Você também pode usar IDEs para escrever e executar o seu código.
- Compilador Online

Interpretador

- Modo linha de comando
- Modo script

Primeiro Programa

- Hello World!

Variáveis e atribuições

- Não é necessário declarar variáveis
- **Exceção:** Quando a variável for incrementada
- Exemplos:
  - i = 5
  - b = 'Palavra Qualquer'
  - terra_plana = False

Nomes de Variáveis

- Eles podem conter tanto letras quanto números, mas **devem** começar com uma letra
- Case sensitive
  - Isso significa que *variavel* é diferente de *Variavel*
- pode-se utilizar símbolos como caractere sublinhado para separar palavras
  - Ex: nome_da_variavel

Palavras Reservadas

São palavras do Python que são reservadas para uso específico, não podem ser usadas como nome de variáveis.

```
>>>import keyword
>>>print(keyword.kwlist)
```

Funções Built-in

No Python temos um conjunto funções nativas da linguagem, dentre elas temos:

- type()
- print()
- isinstance()
- **help()***

Teste algumas das funções do Python, caso tenha dúvidas de como utilizá-las, use a função help().

Funções Built-in

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Conversão de Tipos

Podemos converter uma variável de um tipo para outros, exemplos:

- int(5.3)
- float("5.3")
- str(5)
- **int('a')**
- ord('a')
- chr(97)
- int(True)

Operadores

Podemos utilizar o interpretador do Python para fazer cálculos iterativamente

```
>>>2+3
>>>2/3
```

Um pouco sobre os operadores do Python

- $+, -, *, /$
- **, //, %
- $<, <=, >, >=, ==, !=$
- and, or, not

Ordem de Precedência

Operator	Description
`lambda`	Lambda expression
`if – else`	Conditional expression
`or`	Boolean OR
`and`	Boolean AND
`not x`	Boolean NOT
`in`, `not in`, `is`, `is not`, `<`, `<=`, `>`, `>=`, `!=`, `==`	Comparisons, including membership tests and identity tests
`|`	Bitwise OR
`^`	Bitwise XOR
`&`	Bitwise AND
`<<`, `>>`	Shifts
`+`, `-`	Addition and subtraction
`*`, `@`, `/`, `//`, `%`	Multiplication, matrix multiplication, division, floor division, remainder [5]
`+x`, `-x`, `~x`	Positive, negative, bitwise NOT
`**`	Exponentiation [6]
`await x`	Await expression
`x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`	Subscription, slicing, call, attribute reference
`(expressions...)`, `[expressions...]`, `{key: value...}`, `{expressions...}`	Binding or parenthesized expression, list display, dictionary display, set display

Expressões

Qual é o resultado das seguintes expressões?

- a $=10 + 5\%2 - 3 * *3 \ // \ 10$
- b $= (10 > 2)$ and $(9 == 3 * *2)$ or $(1$ and $0)$

Entrada e Saída (Básico)

Já aprendemos como mostrar algo na tela, mas e para inserir algo no programa?

input():

- Deve ser atribuído à alguma variável
- Retorna uma *String*
- Também podemos mostrar uma mensagem com essa função input('Mensagem:')

Strings

- São cadeias de caracteres
- Operações com strings:

```
#Concatenação
a = 'Python'
b = 'É bom'
print(a+b)
#Repetição
t = 'top'
print(top*10)
```

- Uma string pode ser definida ‘assim’, “assim” ou
  “““Esta string é tão grande
  que não cabe em uma só linha """

Formatando a Saída

- Método format():

```
nome = input()
print('Olá {}, seja bem vindo!'.format(nome))
pi = 3.14159265359
print('PI = {:.2f}'.format(pi))
n = 2.5 * 10**9
print('{:.1e}'.format(n))
```

- *end*, *sep*

Estruturas de Condição

São utilizadas caso uma condição seja atendida... Ou não

```
if condicao:
    acao
elif outra_condicao:
        outra_acao
else:
    outra_acao_diferente
```

Sempre se atente a identação do seu código!

Estruturas de Repetição

São utilizadas para fazer repetições até que certa condição seja atendida...

```
while condicao:
    acao_repetitiva
for i in range(n):
    acao
```

Laço for

O laço for é uma estrutura de repetição que itera sobre uma *sequência*, um a um. Veremos mais sobre sequências depois. Temos, no Python, comandos específicos para lidar com laços de repetições

Comando Break

Função: Encerrar um laço, dada certa condição

```
while True:
    x = input('Digite alguma coisa'):
    if x == 'sair':
        print('Encerrando')
        break
    print('Você digitou',len(x),'caracteres')
```

Comando Continue

Função: Pular a iteração atual, dada certa condição

```
while True:
    x = input('Digite pelo menos 3 caracteres'):
    if x == 'sair':
        print('Encerrando')
        break
    if len(x) < 3:
        continue
    print('Você digitou',len(x),'caracteres')
```

Listas

- São estruturas de dados que armazenam uma coleção de items
- Não tem um tipo de dado específico
- É uma classe, logo possui métodos
- range() retorna uma lista

Funções de Listas

- append(elem)
- extend(list)
- remove(elem)
- pop(index)
- insert(index,elem)
- index(elem)
- count(elem)
- sort(list)
- reverse(list)
- in, not in

Operações com listas

```
a = [1,2,3]
b = [4,5,6]
x = a+b
print(x,'\n',x[2:4])
c = [0]*4
print(c)
g,h,i = a
print(g,h,i)
```

Tuplas

- São estruturas de dados que armazenam uma coleção de items, porém, são imutáveis
- Não tem um tipo de dado específico
- Suporta slicing, mas não as funções de lista

Sets (Conjuntos)

- São coleções de itens que não permite repetições

```
conj = set([1,2,1,3])
print(conj)
```

Funções

- São blocos de comandos que podem ser executados (chamados) ao longo do código;
- Já vimos alguns exemplos ao longo do curso:
  - len(), range()...

Funções

```
def nome_da_func (parametro_1, parametro_2):
    # comandos
    print("Qual será minha função?")
    # cuidado com a identação
#fim da função
```

*Parâmetros são opcionais

Módulos

- O equivalente em Python de bibliotecas, permitem extrair funções de outros códigos já prontos e utilizá-los;
- São disponibilizados pelo próprio Python (Math), pela comunidade ou reutilização dos seus próprios códigos!
- Você também pode criar seus próprios modulos

Módulos

- A importação é feita com a palavra reservada *import*.

```
import math
print("A raiz quadrada de 25 é ", math.sqrt(25))
```

- Ou com *from...import*

```
from math import sqrt
print("A raiz quadrada de 25 é ", sqrt(25))
```

Referências

- SWAROOP, C.H. **A Byte of Python**. Diponível em: `https://python.swaroopch.com/`
- **Stack Overflow Developer Survey 2019** `https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted`