Disciplina:

PYTHON ENGENHARIA DE DADOS

Professor: Nelson Júnior

PUC Minas

PUC Minas

EMENTA

- ✓ Introdução O que é Python?
- ✓ Declaração de Variáveis
- ✓ Entrada e Saída de Dados
- ✓ Tipo de dados
- ✓ Operadores Aritméticos
- ✓ Controle de Fluxo IF/ELSE/ELIF
- ✓ Estrutura de Repetição
- ✓ Dicionários
- ✓ Funções

Python

Python é uma linguagem de programação relativamente **simples** que foi criada por **Guido van Rossum** em 1991, ela é de **alto nível**, **interpretada** e de **alta produtividade**

- **Simples**
  - Elegante - *Menos linhas de código comparando como Java, C, C++*
  - Documentação Gratuita e de fácil acesso
- **Alto nível**
  - Abstração elevada
  - Longe do código de máquina
  - Próximo à linguagem humana – *É como escrever uma carta*

Java:

```
public class PythonandJava {
  public static void main(String[] args)
  {
    System.out.println("Python and Java!");
  }
}
```

Python:

```
print("Python and Java !")
```

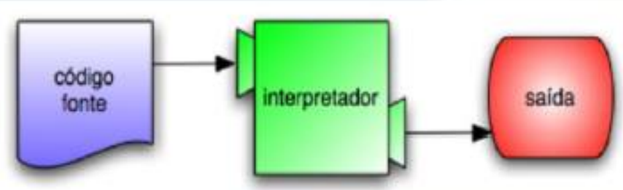
Python

Python é uma linguagem de programação relativamente **simples** que foi criada por **Guido van Rossum** em 1991, ela é de **alto nível**, **interpretada** e de **alta produtividade**

- **Interpretada**

O código fonte é executado por um programa de computador, evita "codifica-compila-roda"

Compilada



Interpretada

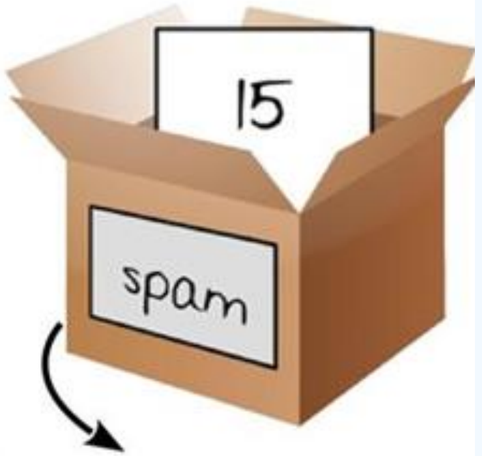
- **Alta Produtividade**
  - Imperativa
  - Orientada a objetos
  - Funcional

PUC Minas


MADE IN PYTHON
python
Google
YAHOO!
Dropbox
reddit
IBM
edX
Instagram
NASA
YouTube
NOKIA
UBER
Spotify

Variáveis

- **Variáveis** são, na verdade, **nomes** dados a **áreas de memória** e servem para:
  - Guardar **valores intermediários**.
  - Construir **estruturas de dados**.
- Uma variável é modificada por meio de um **comando de atribuição**:
  - Var = expressão.
- É possível **atribuir** a várias variáveis **simultaneamente**:
  - Var1, Var2, ..., VarN = expre1, expr2, ..., exprN.




```
>>> spam = 15
```

PUC Minas

Você não pode usar palavras reservadas do python como nome de variável válido

Lista de palavras reservadas do Python: ‘False’, ‘None’, ‘True’, ‘and’, ‘as’, ‘assert’, ‘async’, ‘await’, ‘break’, ‘class’, ‘continue’, ‘def’, ‘del’, ‘elif’, ‘else’, ‘except’, ‘finally’, ‘for’, ‘from’, ‘global’, ‘if’, ‘import’, ‘in’, ‘is’, ‘lambda’, ‘nonlocal’, ‘not’, ‘or’, ‘pass’, ‘raise’, ‘return’, ‘try’, ‘while’, ‘with’, ‘yield’

O restante do nome da variável pode ter letras, números e underline.

variavel_com_0 = ‘variável válida’

Os nomes de variáveis são sensíveis a maiúsculas

```
x = 28
y = X*5
=>NameError: name ‘X’ is not defined
```

PUC Minas

Variáveis

Variáveis String

São variáveis do tipo texto, o texto fica entre aspas “ ”

CÓDIGO

```
a = "Olá mundo"
b = "Hello World"
c = "Olá PET-SI"
d = "Olá UFRRJ"
e = "Curso"
f = "Python"

print(a)
print(b)
print(c)
print(d)
print(e+" de "+f)
```

SAÍDA

```
Olá mundo
Hello World
Olá PET-SI
Olá UFRRJ
Curso de Python
```

```
# Inteiros
a = 28
print(a)
# Saída: 28

# Ponto flutuante
pi = 3.1415
print(pi)
# Saída: 3.14.15

# String
name = 'Alexsandro Felix'
print(name)
# Saída: Alexsandro Felix
```

```
# Boolean
b = True
print(b)
# Saída: True

# Valor vazio ou dado do tipo nulo
x = None
print(x)
# Saída: None

# Variável declarada de forma errada
0 = x
# Saída: SyntaxError: can't assign to literal
```

PUC Minas

```python
a = 2
print(type(a))
# Saída: <type 'int'>

b = 92233720368547758O7
print(type(b))
# Saída: <type 'int'>

pi = 3.14
print(type(pi))
# Saída: <type 'float'>

c = 'A'
print(type(c))
# Saída: <type 'str'>
```

```python
name = 'John Doe'
print(type(name))
# Saída: <type 'str'>

q = True
print(type(q))
# Saída: <type 'bool'>

x = None
print(type(x))
# Saída: <type 'NoneType'>
```

PUC Minas

Comando (função) de entrada do python: **input**

Esse comando permite receber o valor digitado no teclado pelo usuário e atribuir esse valor a variável associadas ao comando.

Forma Geral:

```
variável = input('Mensagem para o usuário')
```

Exemplo:

```
nome = input('Digite o seu Nome')
```

CÓDIGO

```
nome = input("Digite seu primeiro nome: ")
print("A primeira letra do seu nome é: "+nome[0])
```

```
Digite seu primeiro nome: Lucas
A primeira letra do seu nome é: L
```

```
print("Digite seu nome:")
nome = input()
print("Bem-vindo, ", nome)
```

```
Digite seu nome:
Algoritmos em Python
Bem-vindo,  Algoritmos em Pyhon
```

PUC Minas

EXERCÍCIOS

Exercícios

python.org.br

Exercício 1:

Faça um programa que mostre o tradicional “Hello World!” na tela

Exercício 2:

Faça um programa que peça um número e então mostre a mensagem: *O número informado foi [número]*.

Exercício 3: (Sem estruturas de repetição)

Faça um programa que peça 5 itens e suas respectivas quantidades e mostre na tela a lista de itens com a quantidade,

Item 1 – Quantidade: V
Item 2 – Quantidade: W
Item 3 – Quantidade: Y
Item 4 – Quantidade: X
Item 5 – Quantidade: Z

Exercício

python.org.br

Criar uma lista de compra com as seguintes regras:

- É necessário um total de 5 frutas;

- A **primeira** fruta deve **custar 1,00**;
- A **segunda** fruta deve **custar o dobro do valor da primeira**;
- A **terceira** fruta deve **custar metade do valor da primeira**;
- A **quarta** fruta deve **custar 3 vezes o valor da terceira fruta**;
- A **quinta** fruta deve **custar metade do valor da quarta**;

- Cada fruta deve possuir uma variável;
- Usar a menor quantidade possível de variáveis;
- Todas as frutas e seus valores devem ser impressos no seguinte formato:

“A fruta _____ custa _____”

- Operador: Lógico

```
> and    "E" operador lógico
> or     "ou" operador lógico
> not    "Negação" operador lógico
>!=      "Diferente" operador
         lógico
```

- Operador: Lógico

```
> and    "E" operador lógico
> or     "ou" operador lógico
> not    "Negação" operador lógico
>!=      "Diferente" operador
         lógico
```

PUC Minas

O if é uma estrutura de condição que permite avaliar uma expressão e, de acordo com seu resultado, executar uma determinada ação.

```
1 | idade = 18
2 | if idade < 20:
3 |     print('Você é jovem!')
```

No código a seguir temos um exemplo de uso do if no qual verificamos se a variável idade é menor que 20. Em caso positivo, imprimimos uma mensagem na tela e em caso negativo o código seguirá normalmente, desconsiderando a linha 3.

Vimos anteriormente como utilizar o if para executar uma ação caso uma condição seja atendida. No entanto, nenhum comportamento específico foi definido para o caso de a condição não ser satisfeita. Quando isso é necessário, precisamos utilizar a reservada **else**.

```
1 | idade = 18
2 | if idade >= 18:
3 |     print('maior de idade')
4 | else:
5 |     print('menor de idade')
```

Dessa vez, caso a condição avaliada na linha 3 não seja atendida, definimos o fluxo alternativo que o código deve seguir

Adicionalmente, se existir mais de uma condição alternativa que precisa ser verificada, devemos utilizar o **elif** para avaliar as expressões intermediárias antes de usar o **else**, da seguinte forma:

```
1  idade = 18
2  if idade < 12:
3      print('crianca')
4  elif idade < 18:
5      print('adolescente')
6  elif idade < 60:
7      print('adulto')
8  else:
9      print('idoso')
```

PUC Minas

EXERCÍCIOS

PRATICANDO

- Calculo de notas, DADOS :

- Nota 1 + Nota 2 / 2
- Se média <= 3 imprimir reprovado
- Se >3 e < 5 imprimir optativa
- Se >=5 imprimir aprovado

Loops com FOR e WHILE

Em algumas situações, é comum que uma mesma instrução (ou conjunto delas) precise ser executada várias vezes seguidas.

Nesses casos, normalmente utilizamos um loop (ou laço de repetição), que permite executar um bloco de código repetidas vezes, enquanto uma dada condição é atendida.

Em Python, os loops são codificados por meio dos comandos for e while. O primeiro nos permite percorrer os itens de uma coleção e, para cada um deles, executar um bloco de código. Já o while, executa um conjunto de instruções várias vezes enquanto uma condição é atendida.

PUC Minas

While

O comando **while** faz com que um conjunto de instruções seja executado enquanto uma condição é atendida. Quando o resultado dessa condição passa a ser falso, a execução do loop é interrompida, como mostra o exemplo a seguir:

```
1 contador = 0
2 while (contador < 5):
3         print(contador)
4         contador    = contador + 1
```

Neste código, enquanto a variável contador, inicializada com 0, for menor do que 5, as instruções das **linhas 3** e **4** serão executadas.

No loop while, a expressão é testada enquanto for verdadeira. A partir do momento que ela se torna falsa, o código da cláusula else será executado, se estiver presente.

```
1 x = 0
2 while x < 10:
3     print(x)
4     x += 1
5 else:
6     print("fim while")
```

Se dentro da repetição for executado um break, o loop será encerrado sem executar o conjunto da cláusula else.

```
x = 0
while x < 10:
    print(x)
    x += 1
    if x == 6:
        print("x é igual a 6")
        break
else:
    print("fim while")
```

PUC Minas

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

A variável definida na linha 1 **X** recebe os valores entre 2 a 6, cuja saída será :

2

3

4

5

A variável definida na linha 1 é uma lista inicializada com uma sequência de valores do tipo string. A instrução for percorre todos esses elementos, um por vez e, em cada caso, atribui o valor do item à variável n, que é impressa em seguida. O resultado, então, é a impressão de todos os nomes contidos na lista.

```
1 | nomes = ['Pedro', 'João', 'Leticia']
2 | for n in nomes:
3 |     print(n)
```

```
Pedro
João
Leticia


...Program finished with exit code 0
Press ENTER to exit console.
```

PUC Minas

FOR

```
lojas = ['Rio de Janeiro', 'São Paulo', 'Belo Horizonte', 'Curitiba']

for loja in lojas:
    print(loja)
print('Acabou o FOR')
```

```
Rio de Janeiro
São Paulo
Belo Horizonte
Curitiba
Acabou o FOR
```

PUC Minas

```
for i in range(4):
    print(i)
    print(lojas[i])
```

```
0
Rio de Janeiro
1
São Paulo
2
Belo Horizonte
3
Curitiba
```

PUC Minas

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

```
D
a
n
i
e
l
```

PUC Minas

EXERCÍCIOS

EXERCÍCIOS

- Faça um programa que permita somar apenas os números pares da sequencia de inteiros contida no intervalos [x, y].

EXERCÍCIOS

- Leia um número e calcule e imprima sua tabuada

Dicionários

python.org.br

(Dicionários)

- Dicionários são coleções de elementos onde é possível utilizar um índice de qualquer tipo **imutável**.

- Os dicionários implementam mapeamentos que são coleções de associações entre pares de valores
  - O primeiro elemento é a **chave**
  - O segundo elemento é o **conteúdo/valor**

```
DICIONARIO = {"ALAN":'001',"AMARILDO":'002',"ANA":'003',"ARISTIDES":'004'}
```

- As chaves dos dicionários são armazenadas por tabelas de espalhamento (Hash Tables)
- Diferente de listas, não existe uma ordem específica de armazenamento no dicionário

Dicionários

python.org.br

Criação do Dicionário

dic = {"Nome":'Larissa',"Sobrenome":'Maria'}

dic = {"Alan":'001',"Amarildo":'002',"Ana":'003',"Aristides":'004'}

Operações com Dicionário

print(dic["Nome"]) - Imprime o conteúdo da chave Nome

print(dic["Sobrenome"]) – Imprime o conteúdo da chave Sobrenome

print(dic.**keys**()) – Imprime apenas as chaves

print(dic.**values**()) – Imprime apenas os conteúdos

print(dic.**items**()) – Imprime as chaves e conteúdos

Inserindo um novo item no dicionário

dic["Idade"] = '18‘

Alterando o valor das chaves

dic[“Nome"] = ‘Rose'

Dicionários

Função GET: retorna o valor da chave e NONE caso não exista

```
print(dic.get('Larissa'))
print(dic.get('Rose'))
```

Função DEL: Apaga determinado item do dicionário

```
del dic["Nome"]
```

Função CLEAR: Apaga todo o dicionário

```
dic.clear()
```

Função COPY: Copia o conteúdo de um dicionário para outro

```
dic2 = dic.copy()
```

Dicionários – Exemplos

python.org.br

CÓDIGO

```
listatel = {"ana":210012,"bianca":210045,"camila":210019}

print(listatel["ana"])
print(listatel["bianca"])
print(listatel["camila"])

print(listatel.keys())
print(listatel.values())
```

SAÍDA

```
210012
210045
210019
dict_keys(['camila', 'bianca', 'ana'])
dict_values([210019, 210045, 210012])
```

EXERCÍCIOS

Exercícios - Dicionários

python.org.br

Exercício:

Faça um dicionário que contenha os dados de uma pessoa, são os seguintes dados: (Preencha os dados iniciais como preferir)

- Nome
- Ultimo Nome
- Idade
- Curso
- Endereço

a) Imprima o dicionário completo
b) Imprima cada um dos 5 itens separadamente
c) Exclua a chave **CURSO** do dicionário
d) Altere o **ULTIMO NOME** para Lopes
e) Imprima novamente o dicionário completo
f) Imprima apenas o endereço
g) Crie uma cópia do dicionário e altere **Nome** e **Idade**
h) Imprima o segundo dicionário completo

FUNÇÕES

- Criando e usando uma função

```
> def somaTres(numero):
..      return numero + 3
..
> print somaTres(3)
```

FUNÇÕES

FUNÇÃO RANGE

#Gerando sequencia de números em uma lista

```
> numeros = range(1,101)
> print numeros
```

Acessando um item da lista

> print numeros [10]

Acessando partes da lista

> print numeros [50:60]

FUNÇÕES

Números aleatórios

Importar a bibliotéca “random”

```
from random import*
aux = random()  # gera números aleatórios
print(int(aux*10))
```

FUNÇÕES

Números aleatórios

```
print uniform(10, 20)
print randint(100, 1000)
print randrange(100, 1000, 2)
```

A função **random**() retorna um float x tal que 0 <= x < 1.
A função **uniform(10,20)** retorna um float x tal que 10 <= x < 20.
A função **randint(100,1000)** retorna um inteiro x tal que 100 <= x < 1000.
A função **randrange(100,1000,2)** retorna um inteiro x tal que 100 <= x < 1000 e x é par (ou seja, passo 2).

EXERCÍCIOS

Escreva uma aplicação de dicionário com três funções: adicionar um termo ao dicionário, procurar um termo no dicionário e listar todos os termos existentes.

PUC Minas
Virtual