1 INTRODUÇÃO

1.1 Características da linguagem Python

A linguagem de programação Python foi criada em 1991 por Guido Van Rossumem, com a finalidade de ser uma linguagem simples e de fácil compreensão. Apesar de simples, Python é uma linguagem muito poderosa, que pode ser usada para desenvolver e administrar grandes sistemas.

Uma das principais características que diferencia a linguagem Python das outras é a legibilidade dos programas escritos. Isto ocorre porque, em outras linguagens, é muito comum o uso excessivo de marcações (ponto ou ponto e vírgula), de marcadores (chaves, colchetes ou parênteses) e de palavras especiais (begin/end), o que torna mais difícil a leitura e compreensão dos programas. Já em Python, o uso desses recursos é reduzido, deixando a linguagem visualmente mais limpa, de fácil compreensão e leitura.

Entre outras características existentes na linguagem Python, destaca-se a simplicidade da linguagem, que facilita o aprendizado da programação. Python também possui uma portabilidade muito grande para diversas plataformas diferentes, além de ser possível utilizar trechos de códigos em outras linguagens.

Python é um software livre, ou seja, permite que usuários e colaboradores possam modificar seu código fonte e compartilhar essas novas atualizações, contribuindo para o constante aperfeiçoamento da linguagem. A especificação da linguagem é mantida pela empresa *Python Software Foundation* (PSF).

2 VARIÁVEIS

2.1 Tipos de dados básicos

Variáveis são pequenos espaços de memória, utilizados para armazenar e manipular dados. Em Python, os tipos de dados básicos são: **tipo inteiro** (armazena números inteiros), **tipo float** (armazena números em formato decimal), e **tipo string** (armazena um conjunto de caracteres). Cada variável pode armazenar apenas um tipo de dado a cada instante.

Em Python, diferentemente de outras linguagens de programação, não é preciso declarar de que tipo será cada variável no início do programa. Quando se faz uma atribuição de valor, automaticamente a variável se torna do tipo do valor armazenado, como apresentado nos exemplos a seguir:

Exemplos:

A variável **a** se torna uma variável do tipo inteiro.

```
>>> a = 10
>>> a
10
```

A variável **b** se torna uma variável do tipo float.

```
>>>
>>> b = 2.3
>>> b
2.3
```

A variável **c** se torna uma variável do tipo string.

```
>>> c = "Olá mundo"
>>> c
'Olá mundo'
```

ou

```
>>> c = 'Olá mundo'
>>> c
'Olá mundo'
```

A variável d se torna uma variável do tipo lógico.

```
>>> d = True
>>> d
True
```

ou

```
>>> d = False
>>> d
False
```

2.2 Comando de entrada de dados

A atribuição de valor para uma variável pode ser feita utilizando o comando **input()**, que solicita ao usuário o valor a ser atribuído à variável.

Exemplo:

```
>>> nome = input('Entre com o seu nome: ')
...
Entre com o seu nome: Diogenes
>>>
>>> nome
...
'Diogenes'
```

O comando **input()**, sempre vai retornar uma string. Nesse caso, para retornar dados do tipo inteiro ou float, é preciso converter o tipo do valor lido. Para isso, utiliza-se o **int**(string**)** para converter para o tipo inteiro, ou **float(**string**)** para converter para o tipo float.

Exemplos:

```
>>> num = int(input('Entre com um número: '))
...
Entre com um número: 45
>>> num
...
45
```

```
>>> altura = float(input('Entre com sua altura: '))
...
Entre com sua altura: 1.75
>>> altura
...
1.75
```

2.3 Comando de Saída de dados

Para mostrar o valor de uma variável ou de um objeto em Python podemos fazer uso do comando ***print()***.

```
>>> s = 20
>>> print(s)
20
```

O comando **print()** é a única forma de visualizarmos alguma coisa no terminal quando estamos executando um programa via arquivo fonte.

2.4 Funções de conversão de base

Python oferece várias funções embutidas e métodos para trabalhar com números em diferentes bases numéricas, como hexadecimal, binária e octal.

- **hex(x)**: Converte um número inteiro em sua representação hexadecimal como uma string.
- **bin(x)**: Converte um número inteiro em sua representação binária como uma string.
- **oct(x)**: Converte um número inteiro em sua representação octal como uma string.

2.5 Identificadores

Em Python, os nomes das variáveis devem ser iniciados com uma letra, mas podem possuir outros tipos de caracteres, como números e símbolos. O símbolo sublinha (_) também é aceito no início de nomes de variáveis.

Tabela 1 – Exemplos de nomes válidos e inválidos.

Nome	Válido	Comentários
ac1	Sim	Embora contenha um número, o nome ac1 inicia com letra.
código	Sim	Nome formado com letras.
codigo90	Sim	Nome formado por letras e números, mas inicia com letras.
Salario_medio	Sim	O símbolo (_) é permitido e facilita a leitura de nomes grandes.
Salario médio	Não	Nomes de variáveis não podem conter espaços em branco.
_salario	Sim	O sublinha (_) é aceito em nomes de variáveis, mesmo no início.
8a	Não	Nomes de variáveis não podem começar com números.

2.6 Exercícios: Variáveis

1) Escreva mais 3 identificadores que não possam ser usados como nome de variáveis.

2) Experimente as funções de conversão de base.

3) Experimente também a forma de formatação de string para converter números entre diferentes bases.

3 STRINGS

Uma **string** é uma sequência de caracteres simples. Na linguagem Python, as strings são utilizadas com aspas simples ('... ') ou aspas duplas (''... '').

Para se exibir uma string utiliza se o comando **print()**.

Exemplo:

```
>>> print('Olá mundo')
Olá mundo
```

Ou

```
>>> print("Olá mundo")
Olá mundo
```

3.1 Concatenação de strings

Para concatenar strings, utiliza-se o operador +.

Exemplo:

```
>>> print("Apostila"+"Python")
ApostilaPython
```

Ou

```
>>> a = 'Programação'
>>> b = 'Python'
>>> c = a+b
>>> print(c)
ProgramaçãoPython
```

Para se concatenar a mesma string múltiplas vezes, utiliza-se o operador *.

Exemplo:

```
>>> b*5
'PythonPythonPythonPythonPython'
```

3.2 Manipulação de strings

Em Python, existem várias funções (métodos) para manipular strings.
Na tabela a seguir são apresentados os principais métodos para a manipulação as strings.

Tabela 2 - Manipulação de strings

Método	Descrição	Exemplo

len()	Retorna o tamanho da string.	`teste = "Apostila de Python"`
		`len(teste)`
		`18`

capitalize()	Retorna a string com a primeira letra maiúscula	`a = "python"`
		`a.capitalize()`
		`'Python'`

count()	Informa quantas vezes um caractere (ou uma sequência de caracteres) aparece na string.	`b = "Linguagem Python"`
		`b.count("n")`
		`2`

startswith()	Verifica se uma string inicia com uma determinada sequência.	`c = "Python"`
		`c.startswith("Py")`
		`True`

endswith()	Verifica se uma string termina com uma determinada sequência.	`d = "Python"`
		`d.endswith("Py")`
		`False`

isalnum()	Verifica se a string possui algum conteúdo alfanumérico (letra ou número).	`e = "!@#$%"`
		`e.isalnum()`
		`False`

isalpha()	Verifica se a string possui apenas conteúdo alfabético.	`f = "Python"`
		`f.isalpha()`
		`True`

islower()	Verifica se todas as letras de uma string são minúsculas.	`g = "pytHon"`
		`g.islower()`
		`False`

isupper()	Verifica se todas as letras de uma string são maiúsculas.	`h = "# PYTHON 12"`
		`h.isupper()`
		`True`

lower()	Retorna uma cópia da string trocando todas as letras para minúsculo.	`i = "#PYTHON 3"`
		`i.lower()`
		`'#python 3'`

upper()	Retorna uma cópia da string trocando todas as letras para maiúsculo.	`j = "Python"`
		`j.upper()`
		`'PYTHON'`

swapcase()	Inverte o conteúdo da string (Minúsculo / Maiúsculo).	`k = "Python"`
		`k.swapcase()`
		`'pYTHON'`

title()	Converte para maiúsculo todas as primeiras letras de cada palavra da string.	`l = "apostila de python"`
		`l.title()`
		`'Apostila De Python'`

replace(S1,S2)	Substitui na string o trecho S1 pelo trecho S2.	`n = "Apostila teste"`
		`n.replace("teste", "Python")`
		`'Apostila Python'`

find()	Retorna o índice da primeira ocorrência de um determinado caractere na string. Se o caractere não estiver na string retorna -1.	`o = "Python"`
		`o.find("h")`
		`3`

ljust()	Ajusta a string para um tamanho mínimo, acrescentando espaços à direita se necessário.	`p = "Python"`
		`p.ljust(15)`
		`'Python         '`

rjust()	Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda se necessário.	`p = "Python"`
		`p.rjust(15)`
		`'      Python '`

center()	Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda e à direita, se necessário.	`p = "Python"`
		`p.center(15)`
		`'    Python    '`

lstrip()	Remove todos os espaços em branco do lado esquerdo da string.	`s = " Python "` `s.lstrip()`

		'Python '
rstrip()	Remove todos os espaços em branco do lado direito da string.	s = " Python " s.rstrip() ' Python'
strip()	Remove todos os espaços em branco da string.	s = " Python " s.strip() 'Python'

3.3 Fatiamento de strings

O fatiamento é uma ferramenta usada para extrair apenas uma parte dos elementos de uma string;

Nome_String [Limite_Inferior : Limite_Superior]

Retorna uma string com os elementos das posições do limite inferior até o limite superior – 1.

Exemplo:

```
>>> s = 'Programação Orientada a Objetos'
```

Seleciona os elementos das posições 0 até 10:

```
>>> s[0:11]
'Programação'
```

Seleciona os elementos a partir da posição 12:

```
>>> s[12:]
'Orientada a Objetos'
```

Seleciona os elementos até a posição 5:

```
>>> s[:6]
'Progra'
```

3.4 Exercícios: Strings

1) Considere a string A = "Um elefante incomoda muita gente". Que fatia corresponde a "elefante incomoda"?

2) Escreva um programa que solicite uma frase ao usuário e reescreva a frase toda em maiúscula e sem espaços em branco.

3) Escreva um programa que solicite uma frase ao usuário e reescreva a frase usando "leet speak". Por exemplo, hello -> h3110, leet -> 1337, gaming -> g4m1ng, password -> p455w0rd. Utilize a função replace neste caso.

4 Números

Os quatro tipos numéricos simples, utilizados em Python, são números inteiros (**int**), números longos (**long**), números decimais (**float**) e números complexos (**complex**).

A linguagem Python também possui operadores aritméticos, lógicos, de comparação e de bit.

4.1 Operadores numéricos

Tabela 3 - Operadores Aritméticos

Operador	Descrição	Exemplo
+	Soma	5 + 6 = 11
-	Subtração	7 – 2 = 5
*	Multiplicação	2 * 2 = 4
//	Divisão inteira	7 // 2 = 3
%	Resto da divisão inteira	10 % 3 = 1
/	Divisão real	7 / 2 = 3.5
**	Potencia	4 ** 2 = 16

Tabela 4 - Operadores Relacionais

Operador	Descrição	Exemplo
==	Igualdade	A == 5
!=	Diferença	B != 12
<	Menor que	C < 10
<=	Menor ou igual a	D <=7
>	Maior que	E > 2
>=	Maior ou igual a	F >= 8

Tabela 5 - Operadores Lógicos

Operador	Descrição	Exemplo
not	NÃO	not a
and	E	(a<=10) and (b==5)
or	OU	(a<=10) or (b==5)

4.2 Exercícios: números

1) Escreva um programa que receba 2 valores do tipo inteiro x e y, e calcule o valor de z:

$$z = \frac{(x^2 + y^2)}{(x - y)^2}$$

2) Escreva um programa que receba o salário de um funcionário (float), e retorne o resultado do novo salário com reajuste de 35%.

3) Escreva um programa que ache as raízes da equação $2x^2 – 18x + 12$.