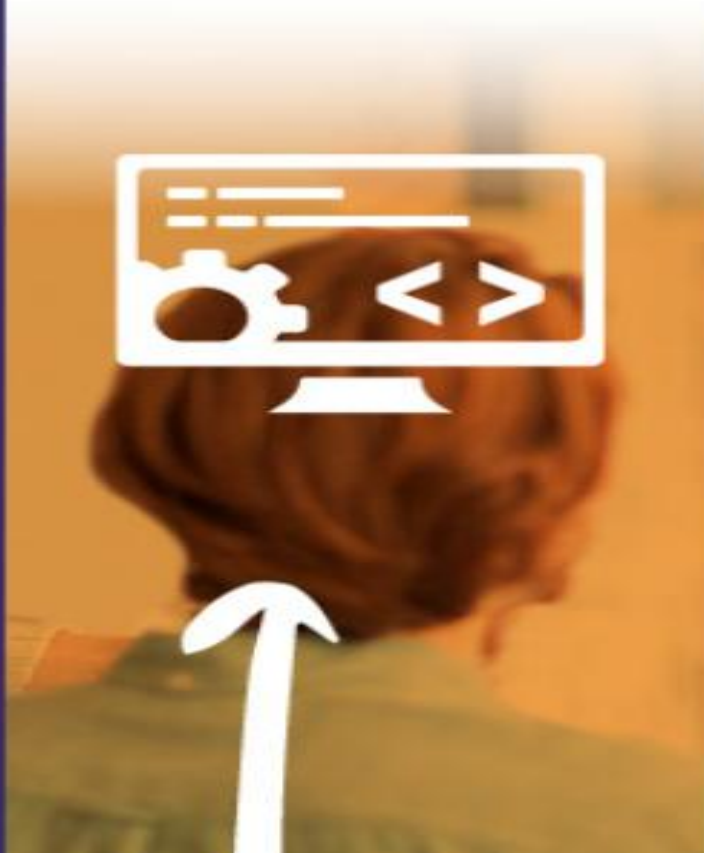


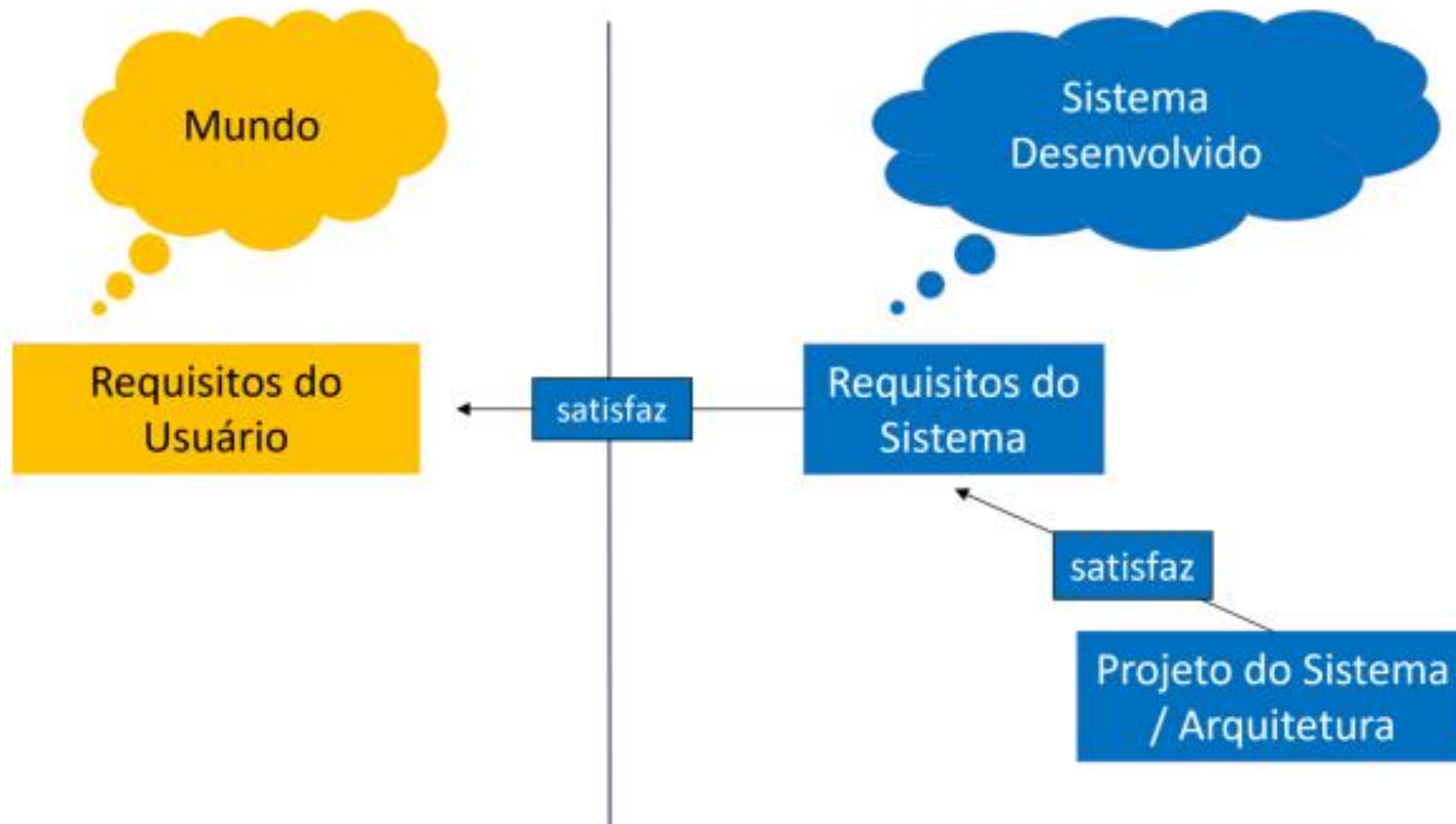
# Engenharia de Software para Ciência de Dados

Um guia de boas práticas com ênfase na construção de sistemas de Machine Learning em Python



*"A parte mais difícil da construção de um sistema de software é decidir precisamente o que deve ser construído. Nenhuma outra parte do trabalho conceitual é tão difícil quanto estabelecer detalhadamente os requisitos técnicos, incluindo todas as interfaces com pessoas, máquinas e outros sistemas de software."*

— Frederick P. Brooks Jr.



## Engenharia de Requisitos

```
graph TD; A[Engenharia de Requisitos] --> B[Produção de Requisitos]; A --> C[Gerência de Requisitos];
```

### Produção de Requisitos

- Levantamento
- Registro
- Verificação
- Validação

### Gerência de Requisitos

- Gerência de configuração
- Controle de mudanças
- Rastreabilidade
- Gerência da qualidade de requisitos



# Recapitulando String

**Como posso definir “string”  
em python?**



**Quais os parâmetros da função print?**





**Em que situação eu  
uso as aspas  
triplas?**



- `programacao_python = "Programação em Python"`
- `# checa se a string contém "Python"`
- `print("Python" in programacao_python)`
- `# checa se a string contém "python"`
- `print("python" in programacao_python)`
- `# checa se a string contém "abacate"`
- `print("abacate" in programacao_python)`

# Operações Matemáticas com Python

```
# Script de exemplos de operações aritméticas em Python
#Autor: Roberto Fabiano Fernandes
# Esse script demonstra operações básicas como soma, subtração, multiplicação,
# divisão, divisão inteira, módulo, potenciação e radiciação, com comentários
explicativos.

# Soma: a + b
# Exemplo: Somar 2 e 3 resulta em 5.
print("2 + 3 =", 2 + 3)  # Calcula 2 + 3 e imprime "2 + 3 = 5"

# Subtração: a - b
# Exemplo: Subtrair 5 de 4 resulta em -1.
print("4 - 5 =", 4 - 5)  # Calcula 4 - 5 e imprime "4 - 5 = -1"

# Multiplicação: a * b
# Exemplo: Cálculo do fatorial de 5 (5 * 4 * 3 * 2 * 1) resulta em 120.
print("5 * 4 * 3 * 2 * 1 =", 5 * 4 * 3 * 2 * 1)  # Multiplica 5, 4, 3, 2 e 1 para
obter 120

# Divisão: a / b (lembre que b deve ser diferente de zero)
# Exemplo: Dividir 10 por 6.5 resulta em um número decimal.
print("10 / 6.5 =", 10 / 6.5)  # Divide 10 por 6.5 e imprime o resultado decimal
```

```
# Divisão inteira: a // b (descarta a parte decimal)
# Exemplo: Dividir 10 por 6.5, considerando somente a parte inteira do quociente.
print("10 // 6.5 =", 10 // 6.5)  # O resultado mostra apenas a parte inteira da
divisão

# Módulo: a % b (resto da divisão de a por b)
# Exemplo: Dividir 10 por 6 deixa resto 4.
print("10 % 6 =", 10 % 6)  # Calcula o resto da divisão de 10 por 6 e imprime 4

# Potenciação: a ** b (a elevado a b)
# Exemplo: 10 elevado a 6 resulta em 1.000.000.
print("10 ** 6 =", 10 ** 6)  # Calcula 10**6 e imprime 1000000

# Radiciação: a ** (1/b) equivale à b-ésima raiz de a
# Exemplo: A raiz cúbica de 8 é calculada por 8**(1/3).
print("8**(1/3) =", 8**(1/3))  # Calcula a raiz cúbica de 8, resultado
aproximadamente 2
print("8**(-1/3) =", 8**(-1/3))  # Calcula o inverso da raiz cúbica de 8,
resultado aproximadamente 0.5
print("(1/8)**(1/3) =", (1/8)**(1/3))  # Calcula a raiz cúbica de 1/8, também
resultando em aproximadamente 0.5
```

# Expressões com resultados diferentes

```
# Expressões com resultados diferentes
```

```
print (3+2*4)
```

```
print ((3+2)*4)
```

# Operadores

<i>Operador</i>	<i>Equivalente a</i>
<code>=</code>	<code>n = 1</code>
<code>+=</code>	<code>n = n + 1</code>
<code>-=</code>	<code>n = n - 1</code>
<code>*=</code>	<code>n = n * 1</code>
<code>/=</code>	<code>n = n / 1</code>
<code>%=</code>	<code>n = n % 1</code>

```

n =13
print (n)
n -= 15
print (n)
n **=4
print (n)
n /=4
print (n)
n %=3
print (n)
n +=12
print (n)
n *=0.5
print (n)

```

```
minutos = 700
horas = minutos /60
print ( minutos , " minutos corresponde a", horas , " horas ")
minutos = 500
horas = minutos /60
print ( minutos , " minutos corresponde a", horas , " horas ")
```



```
# Atribui o valor 700 à variável 'minutos'
minutos = 700
```

```
# Calcula o total de horas (resultado decimal) dividindo os minutos por 60
horas = minutos / 60
```

```
# Calcula a parte inteira das horas usando divisão inteira (//)
horas_inteiras = minutos // 60
```

```
# Calcula os minutos que sobram usando o operador módulo (%)
minutos_restantes = minutos % 60
```

```
# Imprime o resultado formatado: quantos minutos correspondem a quantas horas inteiras e os minutos restantes
print(minutos, "minutos corresponde a", horas_inteiras, "horas e", minutos_restantes, "minutos")
```

```
# Atribui o valor 500 à variável 'minutos'
minutos = 500
```

```
# Calcula o total de horas (resultado decimal) dividindo os 500 minutos por 60
horas = minutos / 60
```

```
# Calcula a parte inteira das horas para 500 minutos usando divisão inteira (//)
horas_inteiras = minutos // 60
```

```
# Calcula os minutos restantes para 500 minutos usando o operador módulo (%)
minutos_restantes = minutos % 60
```

```
# Imprime o resultado formatado para 500 minutos: quantos minutos correspondem a quantas horas inteiras e os minutos restantes
print(minutos, "minutos corresponde a", horas_inteiras, "horas e", minutos_restantes, "minutos")
```

# Variáveis

● ● ● Roberto Fabiano  
Fernandes

---

“Variáveis são utilizadas para **armazenar valores** e para **dar nome a uma área de memória** do computador onde armazenamos dados.”  
(Menezes, 2010)

# Variável

---



O nomes de variáveis são, em linguagens de programação, um tipo de **identificador**.

Podem ser usadas para muitas coisas em programação, a mais elementar seja **conter valores numéricos** que podem ser usados em cálculos.

# Manipulação de Dados

## Identificação

- Para que os **dados** sejam manipulados no computador, é necessário que estes estejam associados a um nome, um **IDENTIFICADOR**.
- **Identificadores funcionam como etiquetas.**
  - Um identificador está para uma região de memória assim como uma etiqueta está para uma gaveta.

# Manipulação de Dados



## Identificação

Python é uma linguagem Case Sensitive, letras minúsculas e maiúsculas são tratadas de maneira diferentes.

- Regras:
  - Deve começar com uma letra ou \_ (sublinhado)
  - Não pode começar com números.
  - Não pode conter caracteres especiais (exceto o sublinhado).
  - Não deve utilizar palavras reservadas (palavras da linguagem Python)
  - Usar o padrão **Snake case**, onde cada palavra é unida à palavra seguinte por um sublinhado.
- Utilize identificadores mnemônicos, ou seja, palavras que nos façam lembrar o caráter do conteúdo armazenado.

# Manipulação de Dados

## Palavras Reservadas Python

and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	
class	exec <sup>(3)</sup>	in	raise	
continue	finally	is	return	
def	for	lambda	try	

# Manipulação de Dados

## Exemplos de Identificadores Válidos

- A
- Nota
- Peso
- Media
- Matricula
- Nota\_1
- \_nota
- nota1



# Manipulação de Dados

## Exemplos de Identificadores Inválidos

- 1nota (começa por numeral)
- A 2 (contém espaço)
- X-y (contém caractere especial)
- Nota do aluno (contém espaço)
- Case (palavra reservada)
- Nota (1) (contém espaço e caractere especial)
- 2a (começa por numeral)

# Manipulação de Dados

## Declaração

- Em Python **não é** preciso definir o tipo da variável, pois a linguagem usa tipagem dinâmica.
  - O tipo é identificado a partir do valor armazenado por inferência.
- É preciso identificar (nomear) a variável antes de manipulá-las.
- O tipo de uma variável muda (dinâmico) conforme o valor atribuído.

# Tipos de Dados



- Numéricos (Inteiros e Reais)
- Lógico
- String

# Operações aritméticas

- Python permite que você realize as seguintes operações aritméticas sobre números:
- **Soma**, indicada pelo símbolo “+”
- **Subtração**, indicada pelo símbolo “-”
- **Multiplicação**, indicada pelo símbolo “\*”
- **Divisão**, indicada pelo símbolo “/”
- **Divisão inteira**, indicada pelo símbolo “//”
- **Resto da divisão inteira**, indicada pelo símbolo “%” – usando espaços antes e depois
- **Potenciação**, indicada pelo símbolo “\*\*”

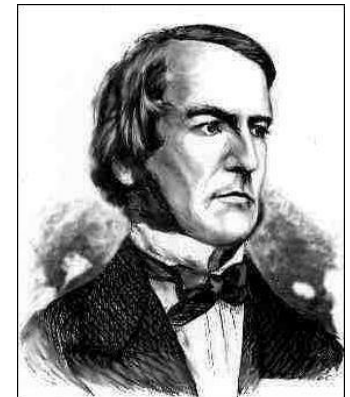
- Existem quatro tipos de valores numéricos em Python:
- Inteiro (**int**): como 5, 78, -467 ou 234567890656545.
- Números de ponto flutuante (**float**): como 1.0, 45.2222, - 46.78
- Boolean (bool): como “True” e “False”
- Números complexos: como (3+4j), (-2+4.5j), (56.2-7.67j)

- Int
  - Números Inteiros
  - São **Positivos** ou **Negativos**
  - Não possuem parte fracionária
  - I = 50
- Float
  - Real de ponto flutuante
  - São **Positivos** ou **Negativos**
  - Possuem parte fracionária
  - F = 3.14

# Tipos de Dados

## Tipo Booleano (Lógico)

- Podem assumir apenas um dentre dois valores:
  - Verdadeiro (sim / 1 / true)
  - Falso (nao / 0 / false)
- São chamados booleanos por causa da álgebra de Boole.
- Em Python é chamado de bool.
- `b = True`
- `o = False`



## Entrada de dados pelo teclado – método input()

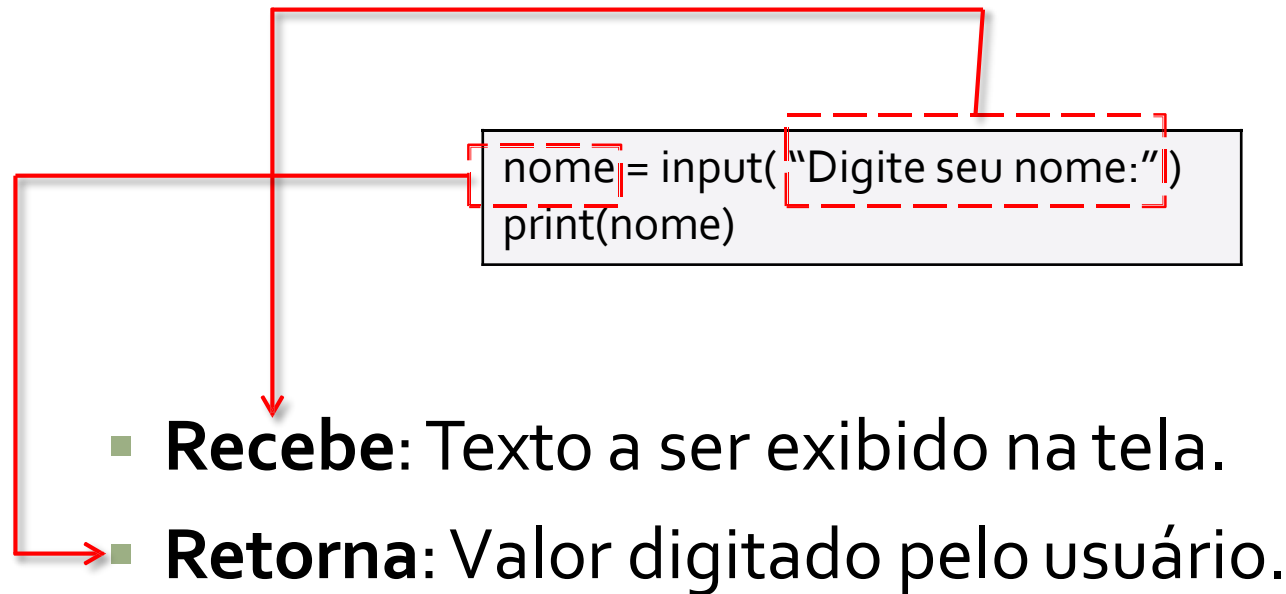


```
num1=int(input("Digite um número: "))  
num2=int(input("Digite outro número: "))  
soma=num1+num2  
print("A soma dos valores foi:", soma)
```



## Entrada de dados pelo teclado – método input()

- O método input recebe um parâmetro string e retorna um dado string.



# Exemplo 1

- Melhorando o exemplo anterior...

```
nome=input("Digite o seu nome: ")  
print("%s é seu nome" % (nome))
```

## Exemplo 2

```
num=input("Digite um número: ")  
print(num)
```

## Exemplo 3

```
num1=input("Digite o primeiro número: ")  
print(num1)  
num2=input("Digite o segundo número")  
soma=num1+num2  
print(soma)
```

# Exemplo 3

```
num1=input("Digite o primeiro número: ")  
print(num1)  
num2=input("Digite o segundo número")  
soma=num1+num2  
print(soma)
```

- Algo de errado na soma?
- Lembre-se que o método input retorna uma String e não int, ou float...

# Conversão da entrada de dados



- `int(input("..."))`
- `float(input("..."))`

```
num1=int(input("Digite o primeiro número: "))
print(num1)
num2=int(input("Digite o segundo número"))
soma=num1+num2
print(soma)
```

```
num1=float(input("Digite o primeiro número: "))
print(num1)
num2=float(input("Digite o segundo número"))
soma=num1+num2
print(soma)
```

# Exemplo 4

- Melhorando o exemplo anterior

```
num1=float(input("Digite o primeiro número: "))  
print(num1)  
num2=float(input("Digite o segundo número"))  
soma=num1+num2  
print("A soma entre %.2f e %.2f é %.2f."%(num1,num2,soma))
```

# Formas de usar o print

- Usando format()
- O método format() permite inserir valores em uma string de maneira mais controlada. Você pode especificar as posições dos valores e formatá-los.

```
[ ] nome = "Joana"
    idade = 22
    mensagem = "Meu nome é {} e tenho {} anos.".format(nome, idade)
    print(mensagem)
```



# Formas de usar o print

```
mensagem = "Meu nome é {nome} e eu estudo {curso}.".format(nome="Ana", curso="Engenharia")  
print(mensagem)
```

# Formas de usar o print

## Usando f-strings

As f-strings foram introduzidas no Python 3.6 e facilitam a interpolação de variáveis diretamente dentro das strings, tornando o código mais legível.

```
nome = "Carlos"  
curso = "Ciência da Computação"  
mensagem = f"Meu nome é {nome} e eu estudo {curso}."  
print(mensagem)
```

# Exemplo 5

- Um funcionário de uma empresa recebe R\$ **x.xxx,xx** de salário por mês. Ao atingir sua meta de produtividade esse funcionário receberá uma bonificação de **xx%** do seu salário. Ajude o dedicado funcionário a descobrir quantos reais ele receberá no fim do mês caso consiga atingir sua meta.
- Escreva um programa em Python que possua uma variável "salario", do tipo float, inicializada com valor **xxxx** e uma variável "novo\_salario" que receberá o salário final com a bonificação. O programa deve exibir o valor da variável "novo\_salario".

# Exemplo 5

- Um funcionário de uma empresa recebe R\$ **x.xxx,xx** de salário por mês. Ao atingir sua meta de produtividade esse funcionário receberá uma bonificação de **xx%** do seu salário. Ajude o dedicado funcionário a descobrir quantos reais ele receberá no fim do mês caso consiga atingir sua meta. Escreva um programa em Python que possua uma variável "salario", do tipo float, inicializada com valor **xxxx** e uma variável "novo\_salario" que receberá o salário final com a bonificação. O programa deve exibir o valor da variável "novo\_salario".

```
salario = float(input("Informe seu salário atual: "))
bonificacao = float(input("Informe de quanto será sua bonificação (em %): "))
acrescimo = salario*(bonificacao/100)
novo_salario = salario+acrescimo

print("Seu salário com bonificação será R$ ", novo_salario)
```

# VARIÁVEIS STRING

.... Variáveis do tipo String armazenam cadeias de caracteres como nomes e textos em geral.  
Chamamos de **Cadeia de caracteres** uma sequência de símbolos como letras, números, sinais de pontuação, etc.

Exemplo: João e Maria comem pão



# VARIÁVEIS STRING

- Uma string em Python tem um tamanho associado, assim como um conteúdo que pode ser acessado caractere a caractere. O tamanho de uma string pode ser obtido utilizando a função **len**.

Essa função retorna o número de caracteres na string.

Dizemos que uma função retorna um valor quando podemos substituir o texto da função por seu resultado.

# VARIÁVEIS STRING – função LEN

PE Aula1 [C:\Users\Fabiano\PycharmProjects\Aula1] - C:\Users\Fabiano\.PyCharmEdu2018.3\config\scratches\scratch.py - PyCharm

File Edit View Navigate Code Help

Project ▾

- ▼ Aula1 C:\Users\Fabiano\PycharmProjects\Aula1
  - > venv library root
  - exe000.py
  - > External Libraries
  - > Scratches and Consoles

scratch.py x

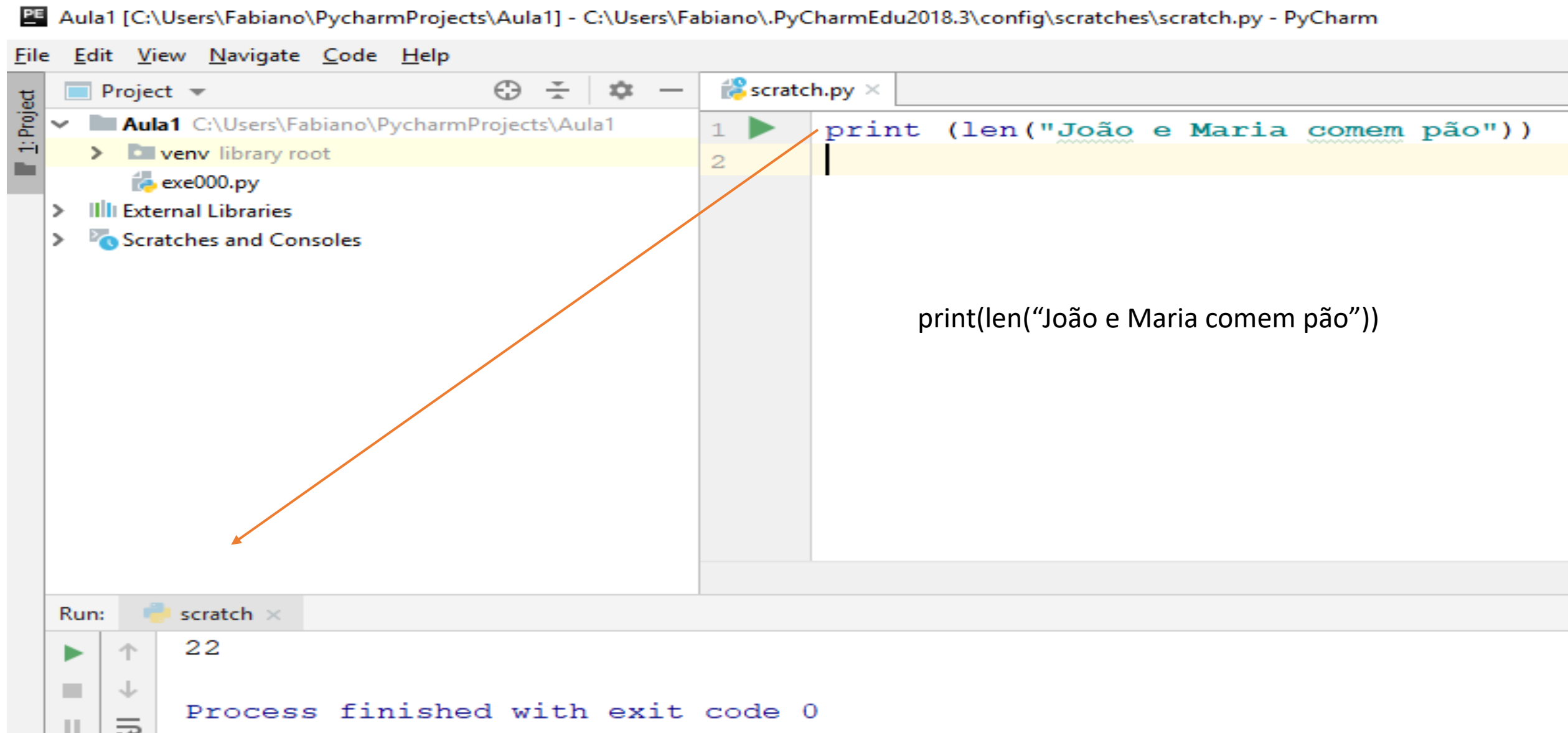
```
1 print (len("João e Maria comem pão"))
2 |
```

print(len("João e Maria comem pão"))

Run: scratch x

22

Process finished with exit code 0



## VARIÁVEIS STRING

String								
0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	H	I

← Índice

← Conteúdo

.... Outra característica de string é poder acessar o seu conteúdo caractere a caractere. Sabendo que uma string tem um determinado tamanho, podemos acessar seus caracteres utilizando um número inteiro para representar sua posição.

Esse número é chamado de índice e ele começa a ser contado de 0.



# VARIÁVEIS STRING

String								
0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	H	I

← Índice

← Conteúdo

Para acessar os caracteres de uma string, devemos informar o índice ou a posição do caractere **entre colchetes []**.

Como o primeiro caractere de uma string é o índice 0, podemos acessar os valores de 0 até o tamanho da string -1.

# VARIÁVEIS STRING

The screenshot shows the PyCharm IDE interface. The top toolbar includes File, Edit, View, Navigate, Code, and Help. The left sidebar shows the Project view with a tree structure: Aula1 (C:\Users\Fabiano\PycharmProjects\Aula1) containing a venv library root, exe000.py, External Libraries, and Scratches and Consoles. The main editor window displays a file named scratch.py with the following code:

```
1 frase="ABCDEFGHI"
2 print (frase[1])
3
```

The bottom panel shows the Run tab with a single run configuration named 'scratch'. The output console displays the result of the execution:

```
B
Process finished with exit code 0
```

A blue arrow points from the index '1' in the code to the output 'B'.

Crie um script em Python que  
Tenha uma variável frase:  
frase="ABCDEFGHI"  
Print(frase[9])

String

0	1	2	3	4	5	6	7	8	← Índice
A	B	C	D	E	F	G	H	I	← Conteúdo

# VARIÁVEIS STRING

Se tentarmos acessar um índice maior que a quantidade de caracteres da **string**, o interpretador emitirá uma mensagem de erro.

The screenshot shows the PyCharm IDE interface. The top toolbar includes 'File', 'Edit', 'View', 'Navigate', 'Code', and 'Help'. The left sidebar shows the project structure for 'Aula1' at 'C:\Users\Fabiano\PycharmProjects\Aula1', with subfolders 'venv library root', 'exe000.py', 'External Libraries', and 'Scratches and Consoles'. The main editor window displays a file named 'scratch.py' with the following code:

```
1 frase="ABCDEFGHI"
2 print (frase[9])
3
```

The bottom panel shows the 'Run' output for 'scratch'. It displays a traceback (most recent call last) with the following details:

- File "[C:/Users/Fabiano/.PyCharmEdu2018.3/config/scratches/scratch.py](#)", line 2, in <module>
- print (frase[9])
- IndexError: string index out of range

At the bottom of the Run panel, it states: 'Process finished with exit code 1'.

```
# Solicita ao usuário para inserir uma string  
texto_usuario = input("Digite um texto: ")
```

```
# Calcula o comprimento da string utilizando a função  
len()
```

```
comprimento = len(texto_usuario)
```

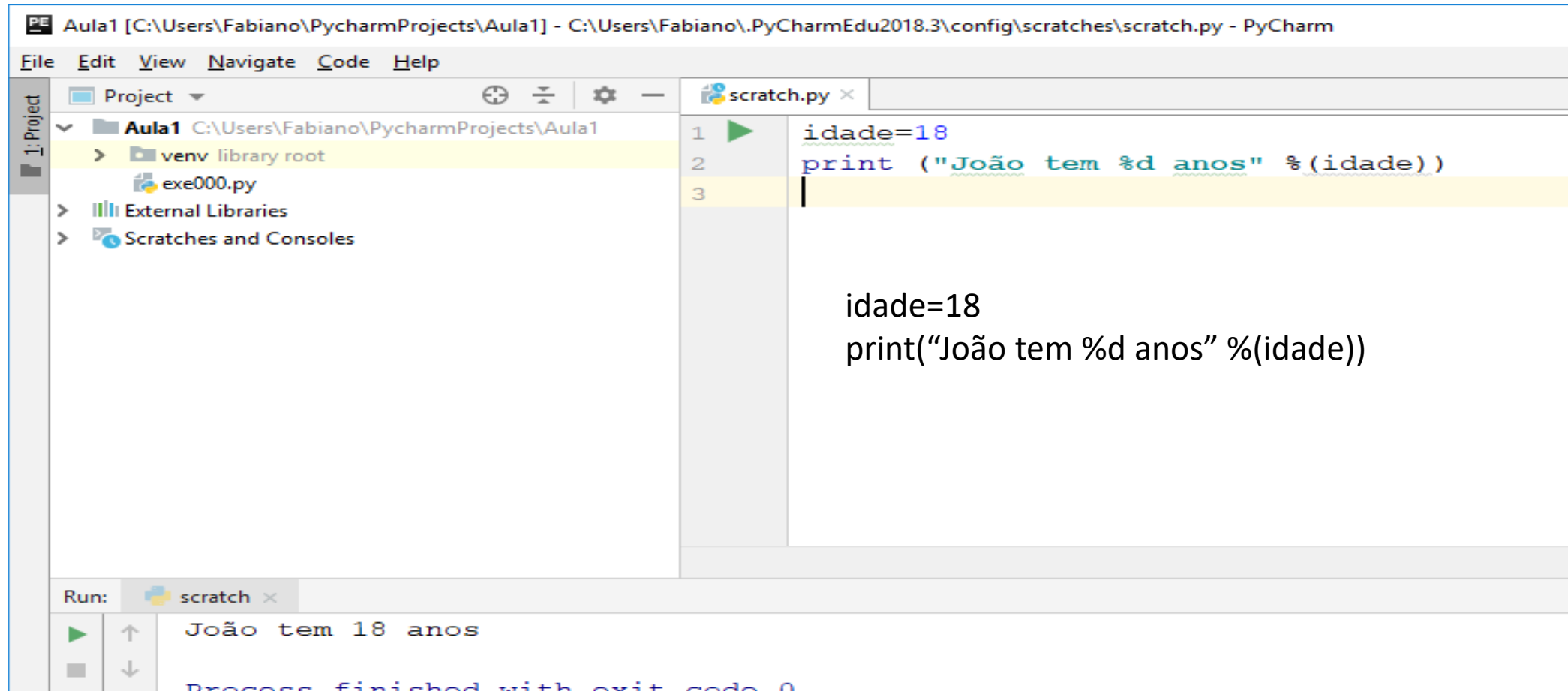
```
# Exibe o comprimento da string  
print(f"O texto digitado possui {comprimento}  
caracteres.")
```

Excluindo espaços ao contar caracteres: Se você deseja contar o número de caracteres excluindo espaços, pode usar a função `replace()` antes de `len()`.

```
comprimento_sem_espacos = len(texto_usuario.replace(" ", ""))  
print(f"O texto digitado possui {comprimento_sem_espacos} caracteres, excluindo espaços.")
```

# VARIÁVEIS STRING - Composição

Juntar várias strings para construir uma mensagem nem sempre é prático. Ex.: João tem X ano. Onde X é uma variável numérica.



The screenshot shows the PyCharm IDE interface. The top toolbar includes 'File', 'Edit', 'View', 'Navigate', 'Code', and 'Help'. The left sidebar shows the 'Project' view with a tree structure: 'Aula1' (C:\Users\Fabiano\PycharmProjects\Aula1) containing 'venv library root', 'exe000.py', 'External Libraries', and 'Scratches and Consoles'. The main editor window displays a file named 'scratch.py' with the following code:

```
1 idade=18
2 print ("João tem %d anos" %(idade))
3
```

Below the code editor, the 'Run' tab is active, showing the output of the script:

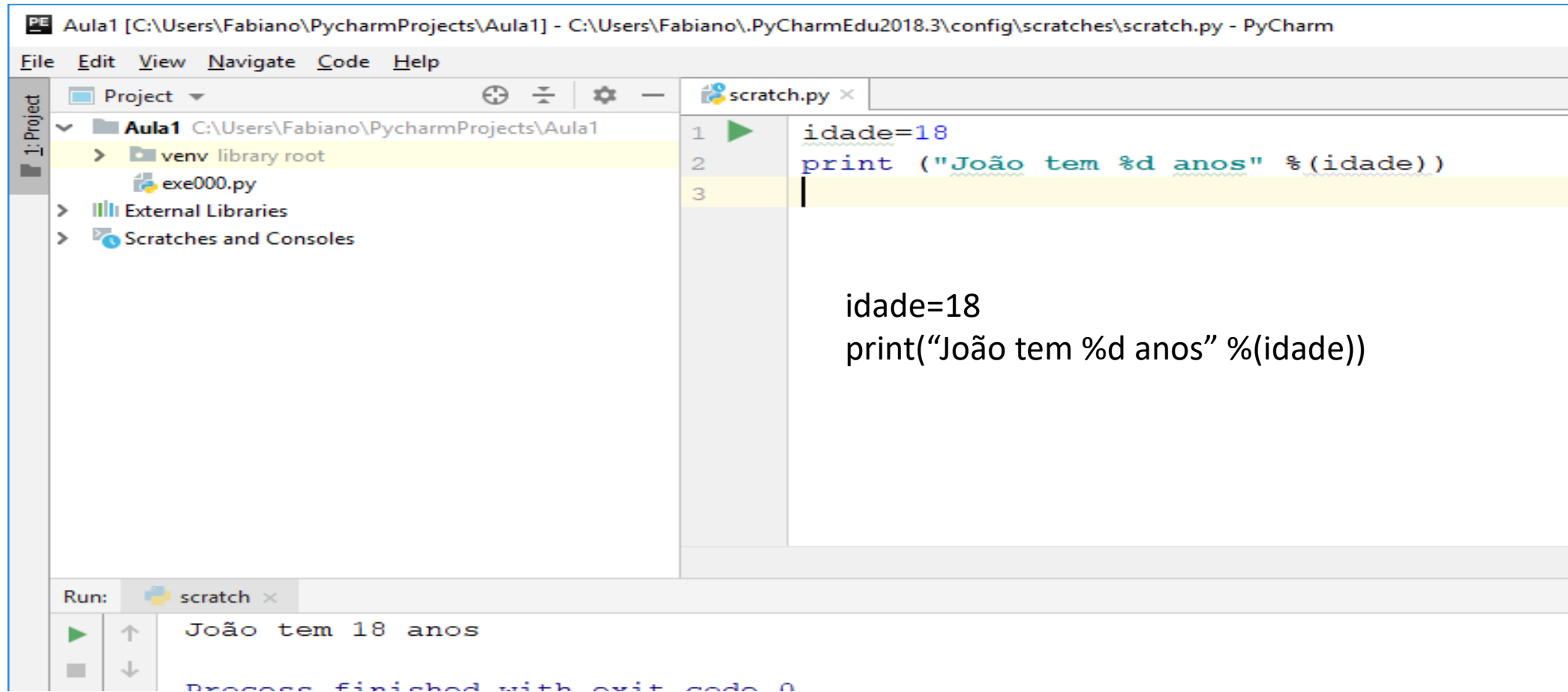
```
João tem 18 anos
Process finished with exit code 0
```

# VARIÁVEIS STRING - Marcadores Composição

Marcador	Tipo
%d	Números inteiros
%s	strings
%f	Números decimais

# VARIÁVEIS STRING - Composição

O símbolo de % é utilizado para indicar a composição da string como o conteúdo da variável **idade**. O %d dentro da primeira string é o que chamamos de marcador de posição. O marcador indica que naquela posição será colocado um valor inteiro, daí o %d



The screenshot shows the PyCharm IDE interface. The top toolbar includes 'File', 'Edit', 'View', 'Navigate', 'Code', and 'Help'. The left sidebar shows the 'Project' view with a tree structure containing 'Aula1', 'venv library root', 'exe000.py', 'External Libraries', and 'Scratches and Consoles'. The main editor window displays a file named 'scratch.py' with the following code:

```
1 idade=18
2 print ("João tem %d anos" %(idade))
3
```

Below the code editor, the 'Run' tab is active, showing the output of the script:

```
João tem 18 anos
Process finished with exit code 0
```



## VARIÁVEIS STRING - Desafio

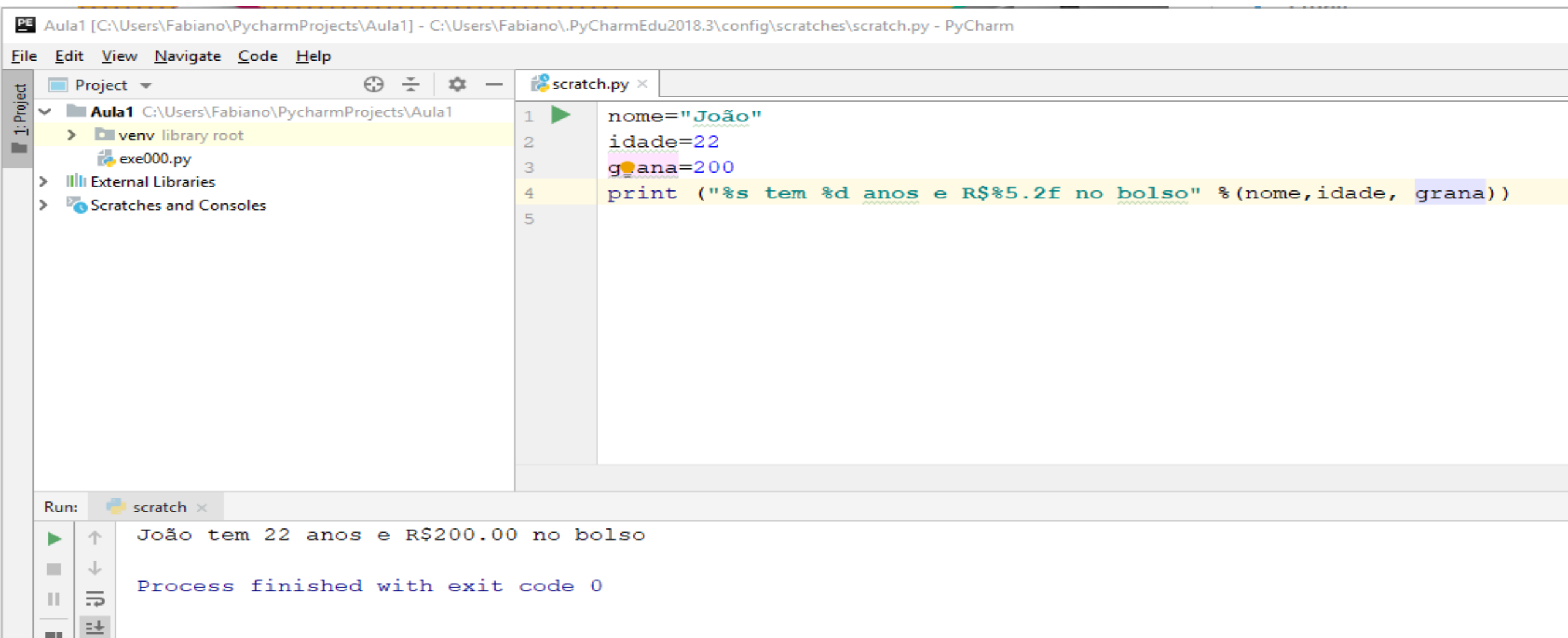
Faça a composição para a seguinte frase

**João tem 22 anos e R\$200.00 no bolso.**

# VARIÁVEIS STRING - Desafio

Faça a composição para a seguinte frase:

**João tem 22 anos e R\$200.00 no bolso.**



The screenshot shows the PyCharm IDE interface. The top toolbar includes buttons for Project, Run, Debug, and Settings. The main editor window displays a file named 'scratch.py' with the following Python code:

```
1 nome="João"
2 idade=22
3 grana=200
4 print ("%s tem %d anos e R$%5.2f no bolso" %(nome,idade, grana))
5
```

The bottom panel shows the 'Run' output for the 'scratch' process. It displays the output of the print statement and the exit code.

```
Run: scratch x
João tem 22 anos e R$200.00 no bolso
Process finished with exit code 0
```

# VARIÁVEIS STRING - Desafio

Desenvolva um script para o seguinte problema:

```
"""
```

```
Tendo como dados de entrada a distância total (em km) percorrida por um  
automóvel e a quantidade de combustível (em litros) consumida para  
percorrê-la, calcule e imprima o consumo médio de combustível.
```

```
"""
```

# VARIÁVEIS STRING - Desafio

Desenvolva um script para o seguinte problema:

```
"""
Tendo como dados de entrada a distância total (em km)
percorrida por um automóvel e a quantidade de combustível
(em litros) consumida para percorrê-la,
calcule e imprima o consumo médio de combustível.
"""

distancia=float(input("Digite a distância percorrida pelo Carro: "))
litros=float(input("Digite a quantidade de litros gastos: "))

print("O carro fez %.2f Km/Litros"%(distancia/litros))
```