

Revisão de Python

A ementa pressupõe que as pessoas inscritas neste módulo já sabem programar e conhecem a linguagem de programação Python. Para alinharmos o conteúdo necessário ao bom desenvolvimento de todos durante o curso, será realizado uma revisão de conceitos e práticas com programação em Python.

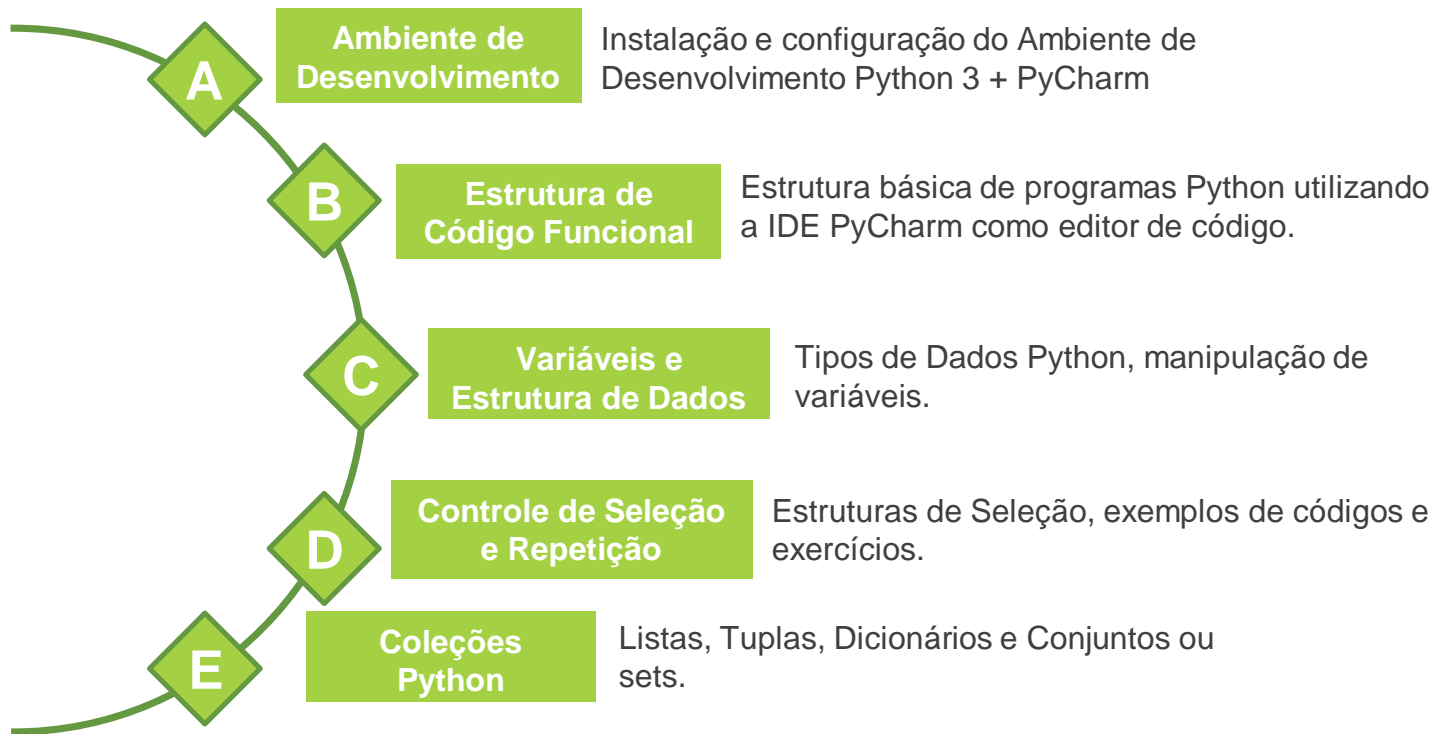
Tópicos da Ementa:

- **Instalação Python 3 e PyCharm**
- **Variáveis e Estrutura de Dados**
- **Funções**
- **Arquivos**
- **Orientação a Objeto**

Aula 19 e 20/07:

- Instalando e configurando o ambiente de desenvolvimento
- Primeiros Códigos Python
- Estrutura do Código, Estrutura de Dados, Controles Condicionais e de Repetição
- Funções e Arquivos

Aula 01 – Revisão de Python

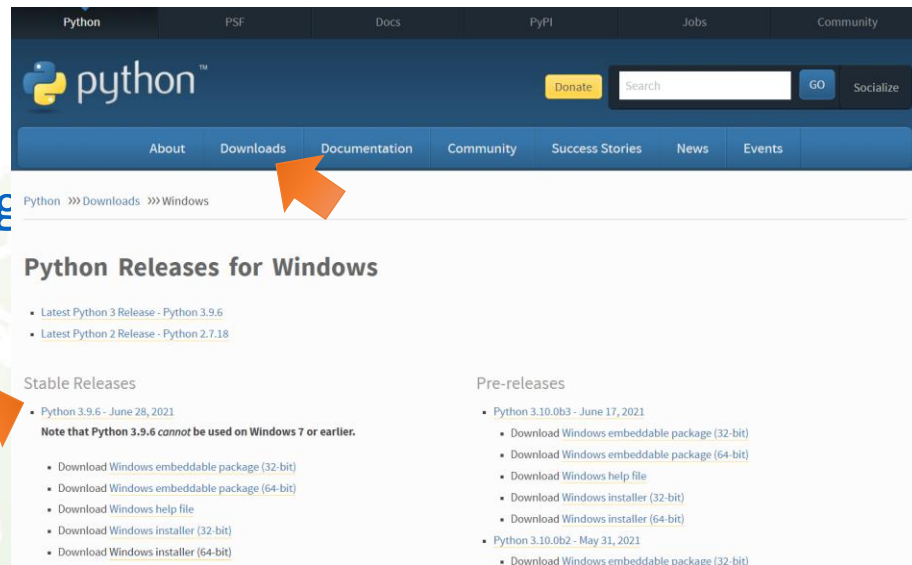


Porquê utilizar Python?

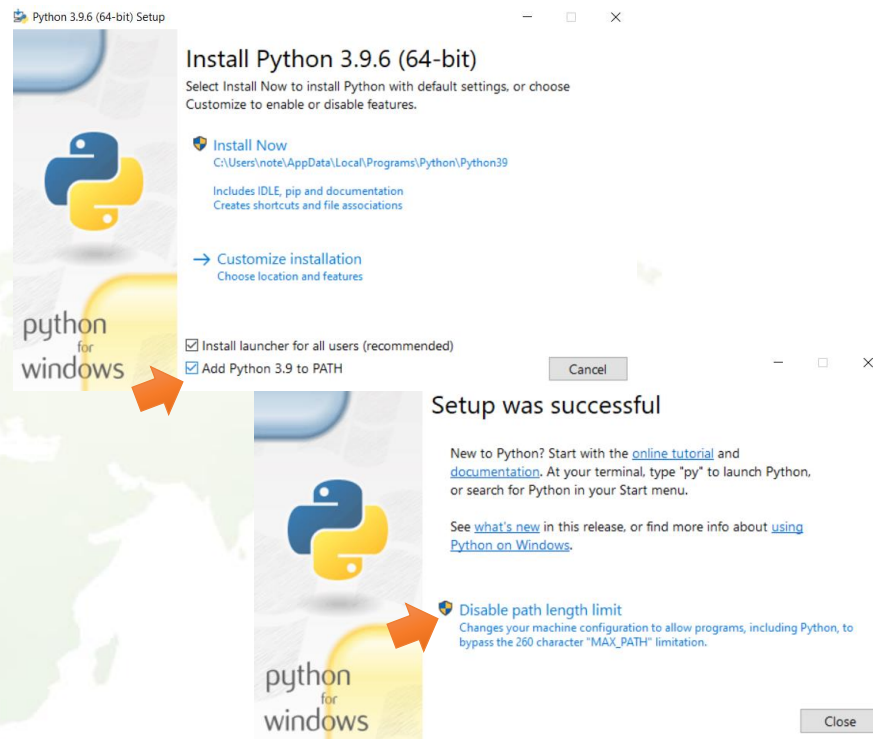
- Antes de iniciarmos nosso módulo de Python em QGIS, iremos revisar as principais sintaxes e como a linguagem pode também ser executada, por usuários de um modo geral;
- Python é amplamente usada em geotecnologias, facilitando a automatização de tarefas repetitivas;
- Possui uma baixa quantidade de palavras reservadas no código, faz o uso de indentação para marcar blocos de comandos e contém um coletor de lixo automático que gerencia a memória - tudo visando facilitar a curva de aprendizagem e a usabilidade para novos usuários;
- Vantagens de usá-la: Fácil aprendizado; Simples de programar; Sintaxe intuitiva; Grande comunidade de usuários; Documentação extensa; *Open Source*; Modularização e Multiplataforma.

Ambiente de Desenvolvimento Python

- Para os usuários Mac e Linux o Python já vem pré-instalado, no caso do Windows precisamos realizar a instalação.
- Para instalar o interpretador Python vamos acessar: <https://www.python.org>
- Acessar a aba **Downloads** e acessar a opção **Python Releases for Windows**.
- Faça download da última versão disponível do **Python 3** para o seu sistema operacional.



- Clicar no Executável que foi salvo na máquina no download para iniciar a Instalação.
- Marcar a opção **Add Python 3.v to Path** e clique em **Install Now**
- O Windows vai solicitar a autorização para prosseguir e finalizará em alguns segundos\minutos
- Ao final clique em **Disable path length limit**, o Windows irá solicitar autorização para prosseguir novamente e sua instalação será concluída com sucesso.



IDEs Python

IDE - Ambiente de Desenvolvimento Integrado

IDLE	PyCharm	Komodo	SPE	Spyder	Eclipse
Já vem instalada junto com o Python.	Disponibilizada pela JetBrains.	Ótima opção de editor, com muitas variedades em recursos.	Desenvolvido com wxPython.	Poderosa IDE para a linguagem Python com edição avançada.	Software pesado, e grande, mas muito poderoso.
Bem simples de ser usada.	Análise do código, depurador gráfico, testador de unidade integrado e Web.	Vi emulation, Emacs key bindings e outros.	Funcionalidades adicionais com o wxGlade, plugin para desenho de telas gráficas	Testes interativos, recursos de depuração e introspecção	Feito em Java e é ideal para desenvolvimento Java. Mas existem plugins para desenvolver em Python.
python.org.br	lifewire.com	malvida.com	old.zope.org	opensource.com	www.eclipse.org
-	-	-	-	-	-

Instalando a IDE PyCharm

<https://www.jetbrains.com/pt-br/pycharm/download>

JET
BRAINS

Para desenvolvimento

Para equipes

Para aprendizado

Soluções

Loja



PyCharm

Previsto para 2021.2

Novidades

Recursos

Aprenda

Comprar

Baixar



Versão: 2021.1.3

Build: 211.7628.24

29 de junho de 2021

[Requisitos do sistema](#)

[Instruções de instalação](#)

[Outras versões](#)

Baixar PyCharm

Windows

macOS

Linux

Professional

Para desenvolvimento Web com Python e desenvolvimento científico. Com suporte para HTML, JS e SQL.

Baixar

Avaliação gratuita

Community

Para o autêntico desenvolvimento Python

Baixar

Open source gratuito

Feedback

Instalando a IDE PyCharm

Agradecemos pelo seu download do PyCharm!

Seu download começará em breve. Se isso não acontecer, use o link abaixo para baixar o arquivo. Baixe e verifique a [soma de verificação SHA-256](#) do arquivo.

[Softwares de terceiros usados pelo PyCharm Community Edition](#)

Introdução

Envie-me materiais educativos úteis durante meu período de avaliação

Insira seu endereço de email para receber dicas e truques

PyCharm Community Edition Setup

Welcome to PyCharm Community Edition Setup

Setup will guide you through the installation of PyCharm Community Edition.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

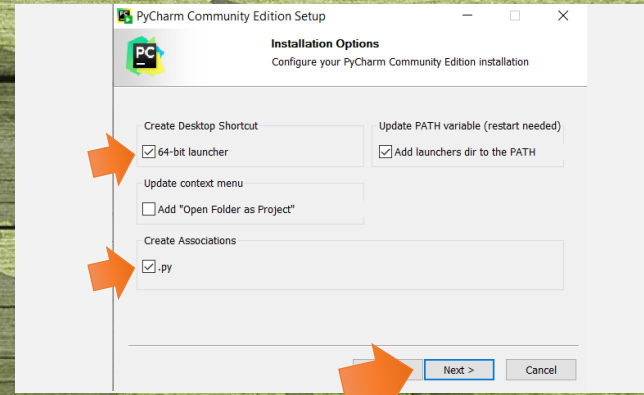
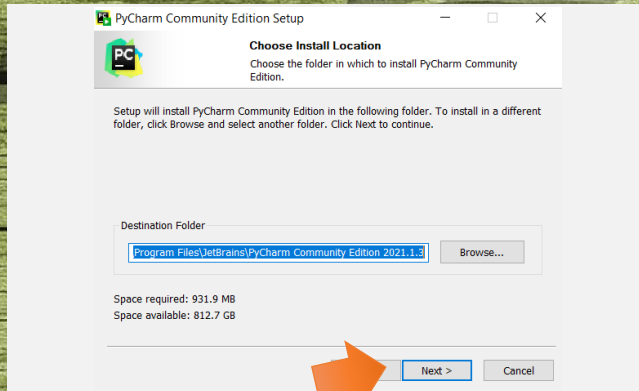
Click Next to continue.

Next >

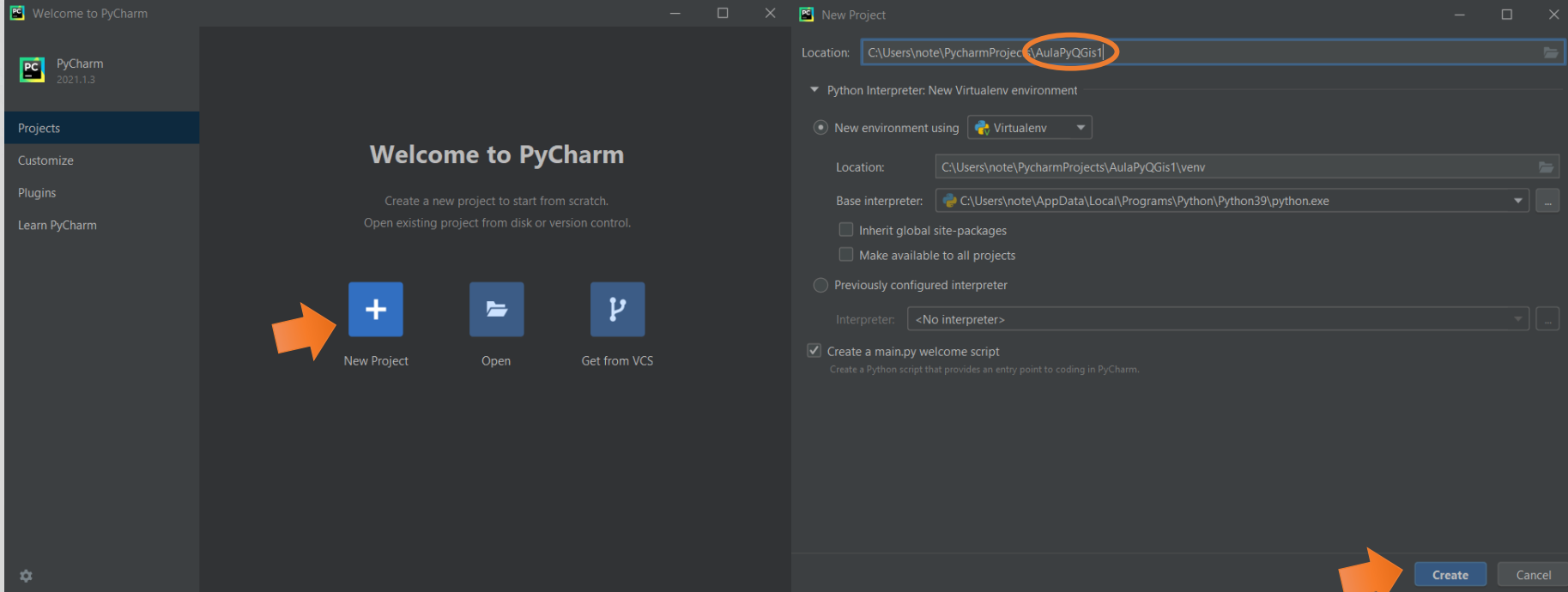
Cancel

Instalando a IDE PyCharm

Detalhes de Configuração



Depois clicar em Install e o executável iniciará a instalação, quando concluir clicar em Finish.
O processo pode levar alguns segundos/minutos a depender da máquina.

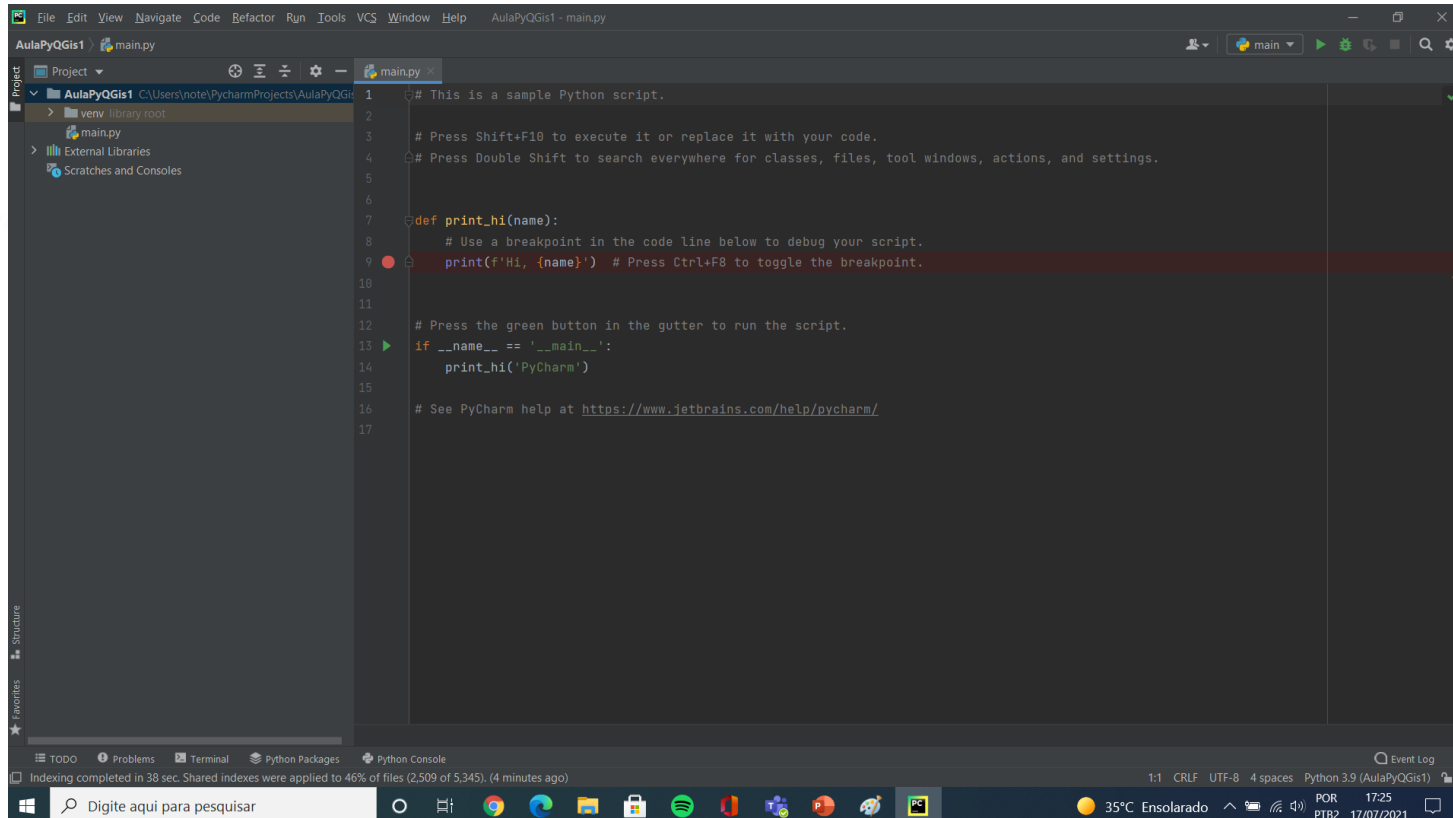


Iniciando um Projeto Python

O Pycharm é usado especificamente para Python e foi desenvolvido para rodar nos principais sistemas operacionais do mercado. Para saber mais, confira os tutoriais oficiais: <https://www.jetbrains.com/help/pycharm/quick-start-guide.html> Mais IDEs: <https://blog.geekhunter.com.br/ides-e-editores-de-codigo-em-python-para-2021/>

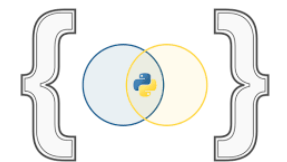
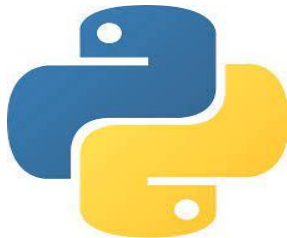
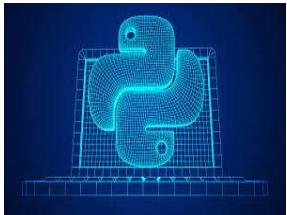
PyCharm

Vamos Codar?



Revisão

Conceitos básicos de Python



Variáveis

Variáveis são espaços alocados na memória para armazenar dados que serão utilizados durante a execução do programa. Estes espaços na memória, podem guardar valores de diversos tamanhos e tipos, tais como números inteiros, números reais, caracteres, textos, frase....

Regras para Nomes de Variáveis

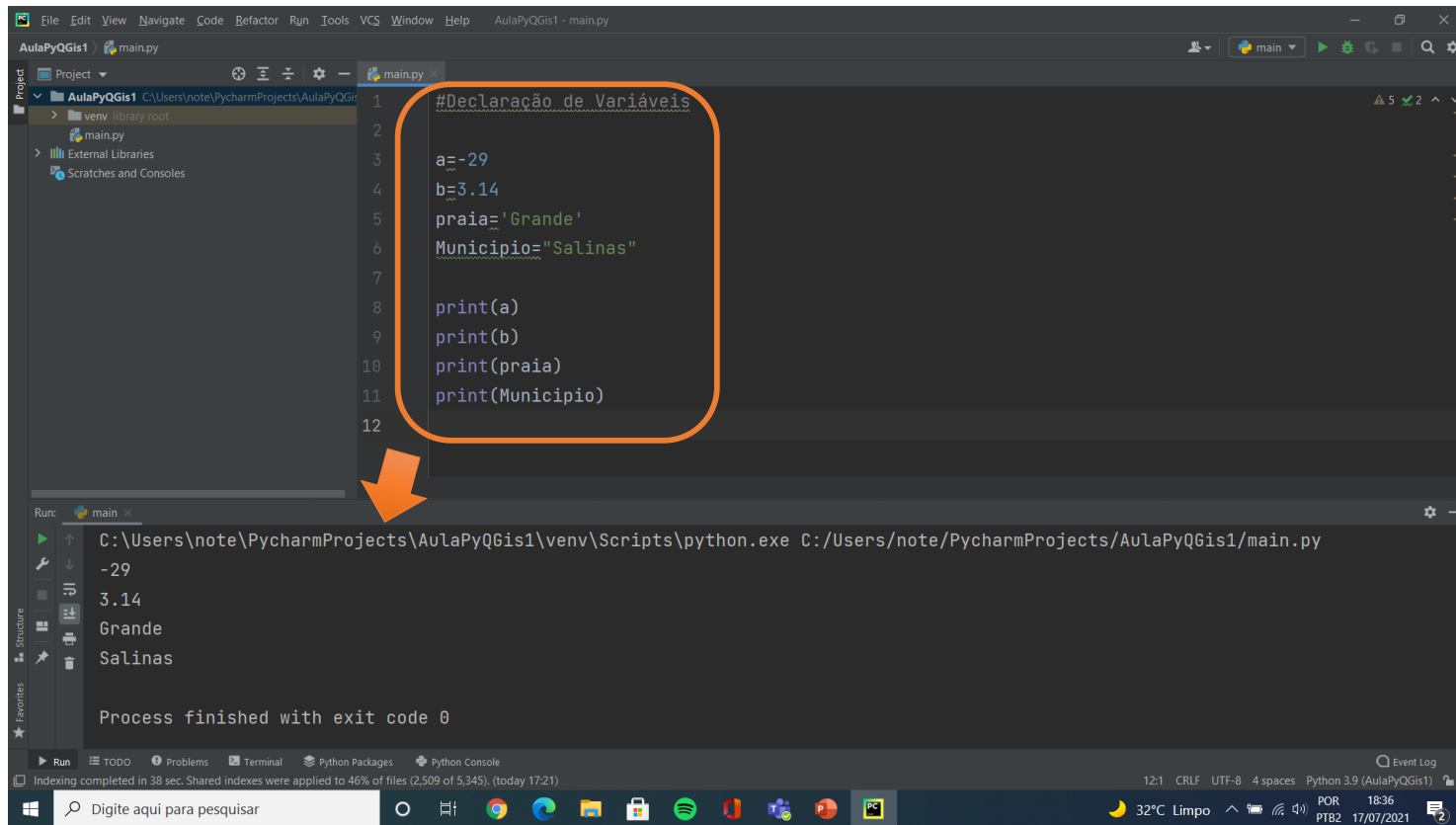
Nomes das variáveis podem começar com uma letra, palavra ou um *underline*, mas não com números. O nome das variáveis são sensíveis a letras maiúsculas, por exemplo: x é diferente de X. Existem palavras reservadas que não devem ser usadas.

Tipo de Dados

Python possui primitivamente os tipos numéricos de números inteiros (*int*), ponto flutuante (*float*) e complexo (*complex*). Letras, palavras e textos são conhecidos como sendo do tipo *String*, ou *str*. Além de outros tipos de dados que veremos na prática.

Variáveis

Declaração de Variáveis / Tipos de Dados



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python file named `main.py` with the following code:

```
1 #Declaração de Variáveis
2
3 a=-29
4 b=3.14
5 praia='Grande'
6 Municipio="Salinas"
7
8 print(a)
9 print(b)
10 print(praia)
11 print(Municipio)
12
```

An orange rounded rectangle highlights the variable declarations (lines 3-6). An orange arrow points from this rectangle to the Run console output.

The Run console at the bottom shows the execution command and output:

```
Run: C:\Users\note\PycharmProjects\AulaPyQGis1\venv\Scripts\python.exe C:/Users/note/PycharmProjects/AulaPyQGis1/main.py
-29
3.14
Grande
Salinas

Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, 4 spaces, and Python 3.9 (AulaPyQGis1).

Nomes de variáveis validos:	Nomes de variáveis inválidos:
tempo	2tempos (não pode começar com numeros)
area_triangulo	area triangulo (não é permitido espaço em branco)
samuel_john	samuel_&_john (não são validos caracteres especiais)
_inicio	total-val (não são valido hifens)

Como representar as Variáveis

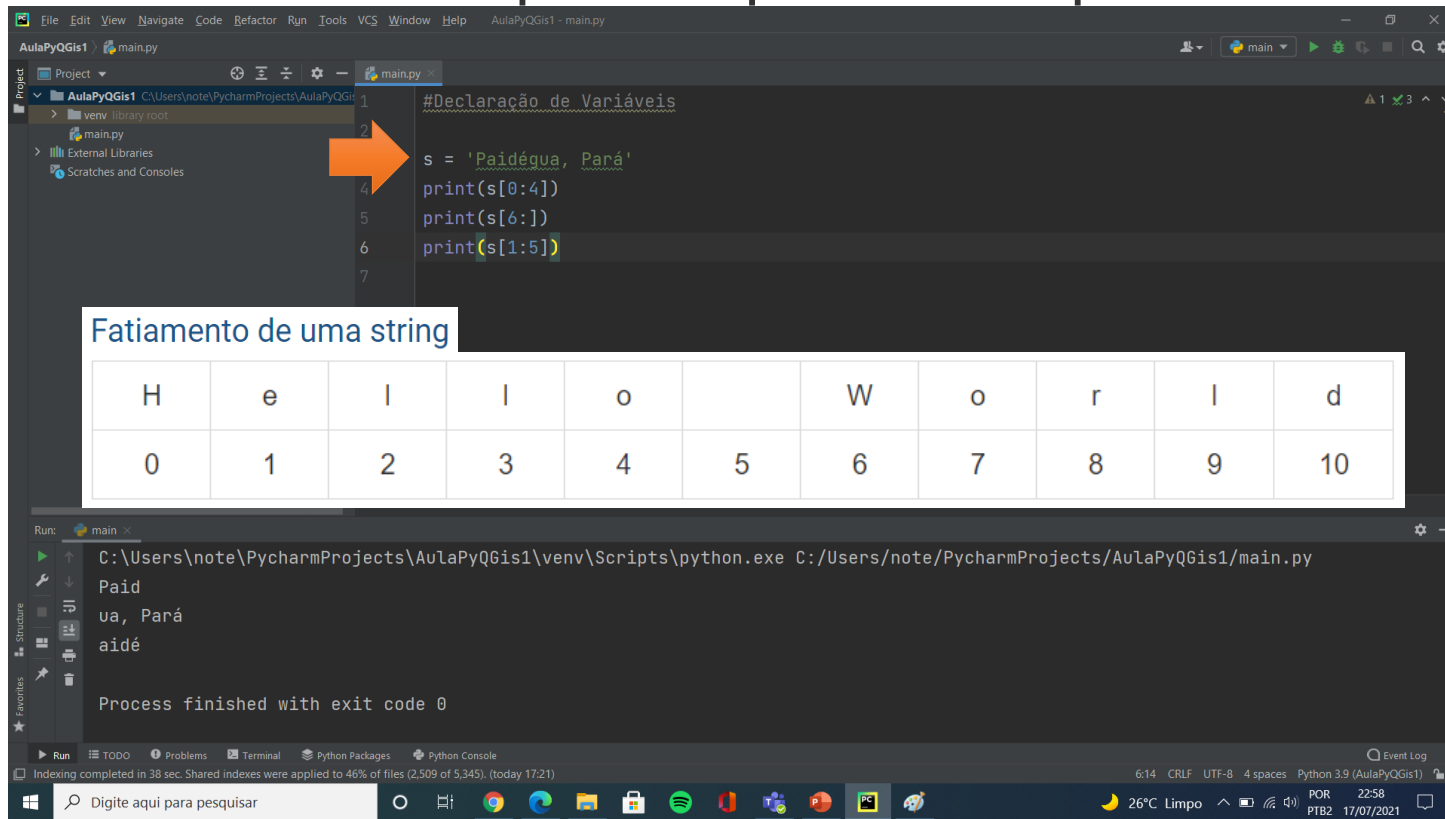


Case sensitive

Python é *case sensitive* então ele diferencia nomes de variáveis com letras maiúsculas e minúsculas exemplo: AREA, Area e area são três variáveis diferentes.

String

Uma *string* é qualquer cadeia de caracteres UTF8 colocada entre aspas simples ou duplas.



The screenshot shows the PyCharm IDE with a Python file named `main.py`. The code defines a string `s` and prints its slices. An orange arrow points from the project view to the code. Below the code, a table illustrates string slicing for the string "Hello World". The bottom panel shows the terminal output of the program.

```
#Declaração de Variáveis
1 s = 'Paidégua, Pará'
2
3 print(s[0:4])
4
5 print(s[6:])
6
7 print(s[1:5])
```

Fatiamento de uma string

H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

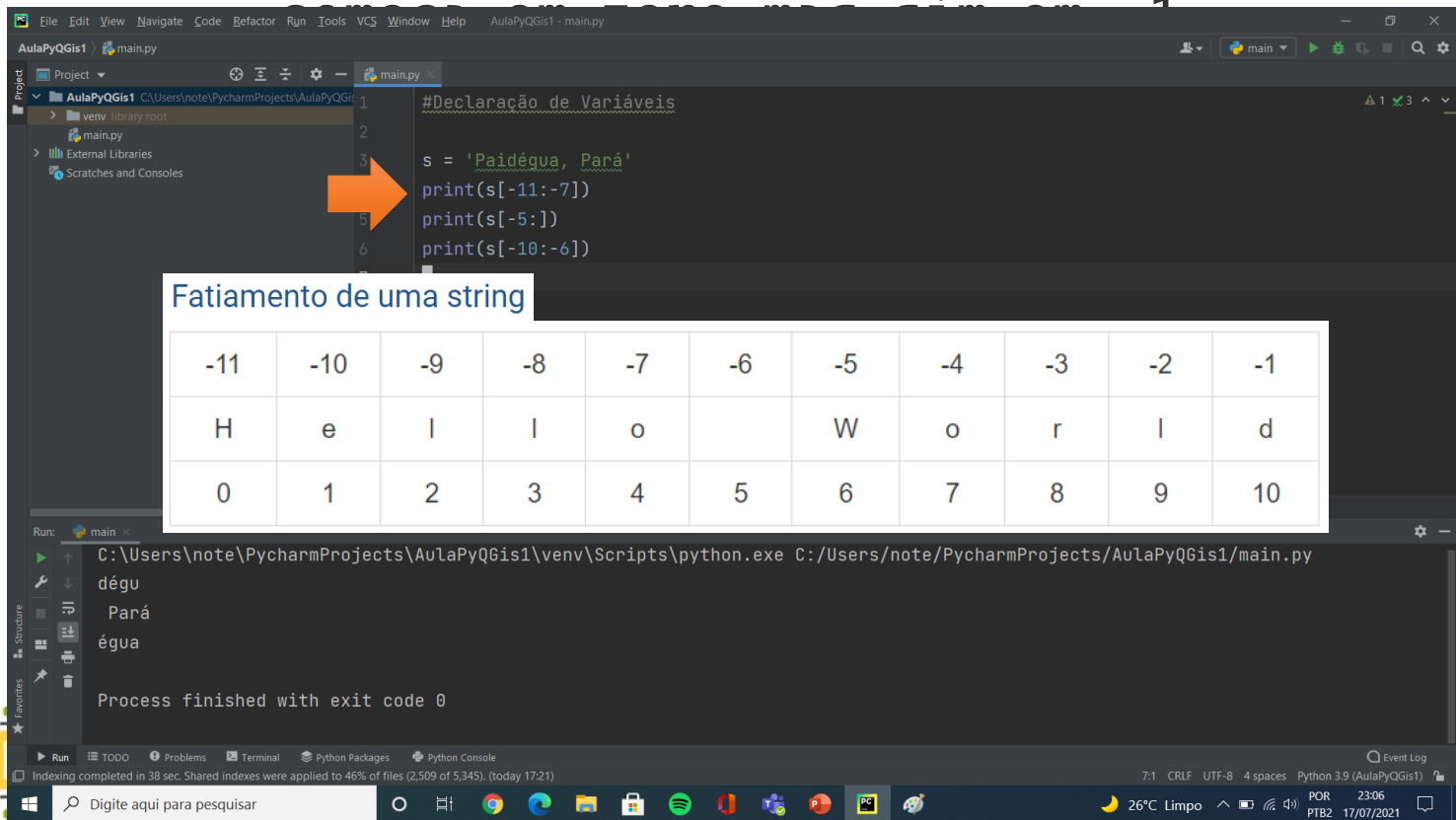
Run: `C:\Users\note\PycharmProjects\AuLaPyQGis1\venv\Scripts\python.exe C:/Users/note/PycharmProjects/AuLaPyQGis1/main.py`

```
Paid
ua, Pará
aidé

Process finished with exit code 0
```


String

Podemos também fatiar uma *string* de trás para frente usando índices negativos, nesses casos a contagem não



```
#Declaração de Variáveis
s = 'Paidégu, Pará'
print(s[-11:-7])
print(s[-5:])
print(s[-10:-6])
```

Fatiamento de uma string

-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

Run: C:\Users\note\PycharmProjects\AulaPyQGis1\venv\Scripts\python.exe C:/Users/note/PycharmProjects/AulaPyQGis1/main.py

dégu
Pará
égua

Process finished with exit code 0

Funções (Métodos) para Strings

Python *strings* também são objetos então na verdade ao invés de funções o que realmente existe são métodos para strings e como a maioria das linguagens Orientadas a Objetos (OO).

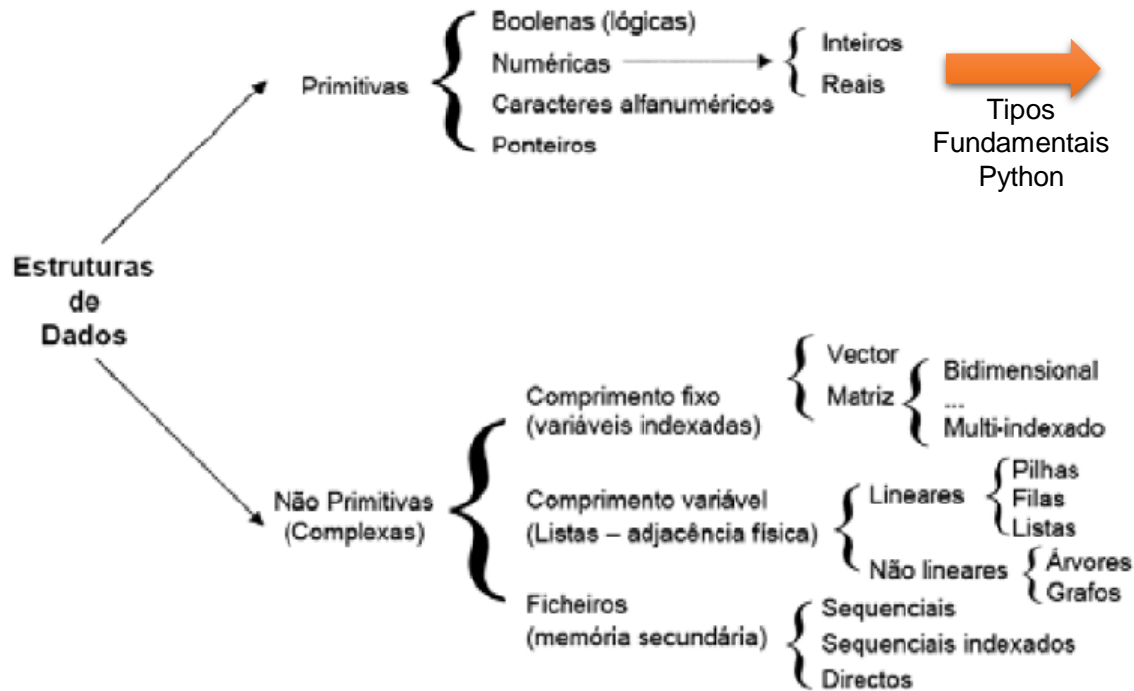
Sintaxe	Descrição
<code>s.capitalize()</code>	Retorna uma nova string com a primeira letra em maiúsculo.
<code>s.center(width, char)</code>	Retorna uma nova string no tamanho width centralizada com espaços ou char preenchendo ambos o lados.
<code>s.ljust(width, char)</code>	Retorna uma nova string no tamanho width alinhada a esquerda e com espaços ou char preenchendo o restante à direita.
<code>s.rjust</code>	Retorna uma nova string no tamanho width alinhada a direita e com espaços ou char preenchendo o restante à esquerda.
<code>s.count(t, end)</code>	Retorna o número de ocorrências de t em s ou na fatia de s finalizada em end.
<code>s.find(t, start, end)</code>	Retorna a posição de t mais a esquerda em s ou na fatia começando por start e terminado em end. Se nada for encontrado retorna -1.
<code>s.rfind(t, start, end)</code>	O mesmo que find() porém buscando a partir da direita.
<code>s.isalnum()</code>	Retorna True se s não for vazia e cada caractere de s é alfanumérico.
<code>s.isalpha()</code>	Retorna True se s não for vazia e cada caractere de s é uma letra.
<code>s.isdecimal()</code>	Retorna True se s não for vazia e cada caractere de s é um numérico Unicode.
<code>s.isdigit()</code>	Retorna True se s não for vazia e cada caractere de s é um numérico ASCII.
<code>s.isidentifier()</code>	Retorna True se s não for vazia e um identificador válido.
<code>s.islower()</code>	Retorna True se s não for vazia e todos os caracteres estão em minúsculas.
<code>s.isupper()</code>	Retorna True se s não for vazia e todos os caracteres estão em maiúscula.



```
1  #Métodos de Strings
2
3  frase='Curso PyQGIS - SEMAS'
4  print(frase.count('o'))
5  print(frase.upper())
6  print(len(frase))
7  print(len(frase.strip()))
8  print("QGIS" in frase)
9  print(frase.find('Py'))
10 print(frase.replace('PyQGIS', 'Python no QGIS'))
11 print(frase)
12 frase=(frase.replace('PyQGIS', 'Python no QGIS'))
13 print(frase)
```

Estruturas de Dados

Visão Geral



Tipos
Fundamentais
Python



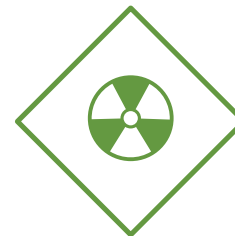


Operadores Aritméticos

Operador	Descrição
+	soma
-	subtração
*	multiplicação
/	divisão
//	Divisão trunca a parte fracionaria
%	Produz o resto da divisão
**	Exponenciação
abs(x)	Retorna o valor absoluto de x
pow(x, y)	O mesmo $x^{**}y$
round(x, n)	Retorna um int ou float arredondado para n casas decimais se n for dado

Revisão

Operadores do Python



Operadores de Comparação

Operador	Descrição
<	menor que
<=	menor ou igual a
>	maior que
>=	maior ou igual a
==	igual
!=	diferente
is	Verifica se duas variáveis apontam para o mesmo objeto
in	Verifica se a variável ou o objeto a esquerda fazem parte de uma sequencia ou coleção (veremos mais tarde o real significado disso) a direita
not in	Ao contrario de in verifica se a variável ou o objeto a esquerda não fazem parte de uma sequencia ou coleção





Revisão

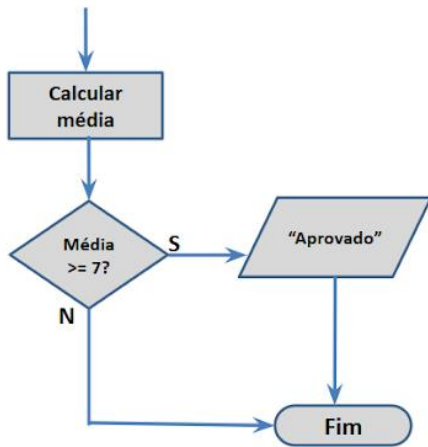
Operadores do Python

Operadores Lógicos

Operador	Descrição	
and	P and Q	Retorna True só se P e Q forem verdadeiros se não retorna False
or	P or Q	Retorna False só se P ou Q forem falsos se não retorna True
not	not P	Se P é verdadeiro retorna False se P falso retorna True



Simples If Desvios Condicionais



```
1 print('Programa para calcular se uma pessoa pode votar')
2 print()
3
4 nome = input('Entre com o nome da pessoa: ')
5 print()
6 a1 = int(input("Entre com o ano de nascimento: "))
7 print()
8 a2 = int(input("Entre com ano atual: "))
9 print()
10
11 idade = a2 - a1
12 if idade > 16:
13     print(nome, 'pode votar')
14
```

Run console output:

```
Entre com o nome da pessoa: Danielle
Entre com o ano de nascimento: 1978
Entre com ano atual: 2021
Danielle pode votar
```

Exemplo if simples que verifica se uma pessoa pode ou não votar

A estrutura de controle if analisa uma ou mais condição e conforme essa condição seja verdadeira ela realiza um bloco de instruções imediatamente após essa confirmação.

```

1
2 x=int(input("Digite um número inteiro: "))
3
4 if x<0:
5     print("O número é negativo.")
6 elif x>0:
7     print("O número é positivo.")
8 else:
9     print("O número é zero.")
10

```

```

# Programa para calcular a media de candidato
print('Programa para verificar se um candidato passou em pr

print()
nome = input('Entre com o nome do candidato: ')
print()
nota1 = float(input("Entre com a nota do primeiro avaliador: "))
print()
nota2 = float(input("Entre com a nota do segundo avaliador: "))
media = (nota1 + nota2)/2

print()
if media >= 7:
    print('{0} aprovado com media = {1:4.2f}'.format(nome, media))
elif media < 7 and media >= 3:
    print('{0} vai para a lista de espera = {1:4.2f}'.format(nome, media))

```

Exercício If, elif e else

Verifica se (if) uma condição ou se outras condições (elif) são atendidas. Se nenhuma das condições forem atendidas, podemos também instruir que alguma outra coisa seja feita (else).

```

# Programa para calcular a media de candidato
print('Programa para verificar se um candidato passou em processo seletivo')

print()
nome = input('Entre com o nome do candidato: ')
print()
nota1 = float(input("Entre com a nota do primeiro avaliador: "))
print()
nota2 = float(input("Entre com a nota do segundo avaliador: "))
media = (nota1 + nota2)/2

```

```

    aprovado com media = {1:4.2f}'.format(nome, media))
    media >= 3:
    para a lista de espera = {1:4.2f}'.format(nome, media))
    ou reprovado com media = {1:4.2f}'.format(nome, media))

```

```

(input("Digite um número inteiro: "))

```

```

3
4 if x<0:
5     print("O número é negativo.")
6 elif x>0:
7     print("O número é positivo.")
8 else:

```

Estruturas de Repetição

A estrutura de controle *while* e *for* servem para repetir um bloco de instruções.

while

A estrutura de controle *while* serve para repetir todo um bloco de instruções enquanto sua condição for verdadeira.

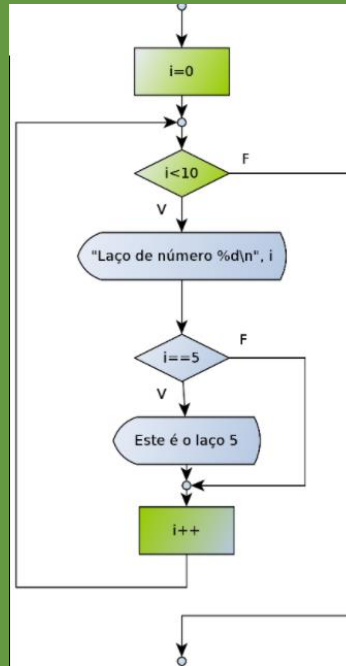
for

For é basicamente feito para percorrer as coleções do Python. Para realizar um loop controlado por uma série numérica devemos usar a função `range()`.



Cuidado!

Prestar bastante atenção para que a condição de parada do loop seja satisfeita em algum momento, ou teremos um *loop* infinito.



```

flag = True
# Aqui o usuário tem três tentativas para entrar com o número certo
for i in range(0, 3):
    print()
    numero = int(input('Entre com um número inteiro entre 1 e 9: '))
    if numero < 1 or numero > 9:
        print()
        print('O número deve ser número inteiro entre 1 e 9')
        flag = False
    else:
        flag = True
        break
if flag:
    print('\nTabuada de', numero)
    print()
    for contador in range(1, 10):
        resultado = numero * contador
        print("{0} x {1} = {2}".format(numero, contador, resultado))
else:
    print('\nVocê excedeu o número de tentativas para entrar com um número')
    print()

```

For

O loop *for* do Python é basicamente feito para percorrer os tipos de dados de coleções (que iremos ver daqui a pouco) do Python.

While

A estrutura de controle *while* serve para repetir todo um bloco de instruções enquanto sua condição for verdadeira.

```

# Programa que imprime a tabuada de um número entre 1 e 9

numero = 0
contador = 1
while numero < 1 or numero > 9:
    print()
    numero = int(input('Entre com um número inteiro entre 1 e 9: '))

    if numero < 1 or numero > 9:
        print()
        print('O número deve ser número inteiro entre 1 e 9')
print('\nTabuada de', numero)
print()
while contador <= 9:
    resultado = numero * contador
    print("{0} x {1} = {2}".format(numero, contador, resultado))
    contador += 1
print()

```

and appeal to your Presentations.

Tipos de Dados

Tipos Coleções - são os tipos que podem armazenar objetos de outros tipos além deles mesmos.



Listas

Sequencia ordenada com 0 ou + objetos aceitando o fatiamento com colchetes [].

Tuplas

São sequencias ordenadas assim como listas, e assim como strings são imutáveis.

Listas



Tuplas



Dicionários



Conjuntos



Dicionários

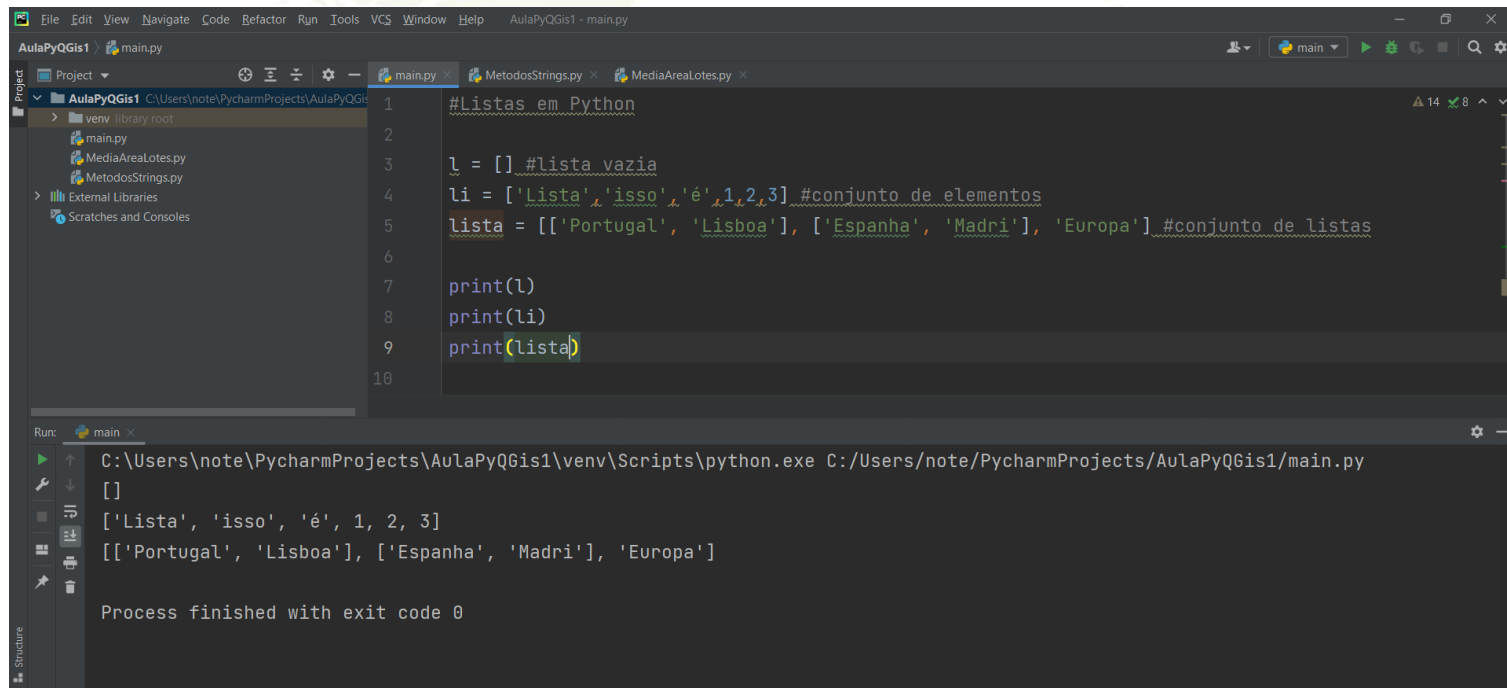
Coleção desordenada com chaves para os objetos, valor onde a chave é usada para referenciar um determinado valor.

Conjuntos

Conjuntos só aceitam tipos de dados imutáveis como inteiros, floats, tuplas e strings.

Listas

Uma lista é uma sequência ordenada com zero ou mais objetos sendo também do tipo sequencial aceitando o fatiamento com colchetes [] e também a função len() e o comparador in e not in.



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project name is 'AulaPyQgis1'. The left sidebar shows the project structure with folders like 'venv' and 'library root', and files like 'main.py', 'MediaAreaLotes.py', and 'MetodosStrings.py'. The main editor window displays a Python script titled 'main.py' with the following code:

```
1 #Listas em Python
2
3 l = [] #lista vazia
4 li = ['Lista', 'isso', 'é', 1, 2, 3] #conjunto de elementos
5 lista = [['Portugal', 'Lisboa'], ['Espanha', 'Madri'], 'Europa'] #conjunto de listas
6
7 print(l)
8 print(li)
9 print(lista)
10
```

The bottom panel shows the Run output for the script:

```
C:\Users\note\PycharmProjects\AulaPyQgis1\venv\Scripts\python.exe C:/Users/note/PycharmProjects/AulaPyQgis1/main.py
[]
['Lista', 'isso', 'é', 1, 2, 3]
[['Portugal', 'Lisboa'], ['Espanha', 'Madri'], 'Europa']

Process finished with exit code 0
```

Exercícios Práticos

Listas em Python

Acessando itens de uma lista

Não confundir o fatiamento de uma lista [start:stop:step] com o elemento de uma lista [índice].

Inserir itens no fim da lista

Podemos simplesmente concatenar objetos ao fim da lista usando o operador de concatenação (+).

Percorrendo lista com For

Na primeira maneira trazemos os elementos da lista através de seus índices, na segunda maneira buscamos diretamente os elementos através do operador *in*.

Alterando os itens

Basta acessar esse elemento pelo índice e atribuir o novo valor.

Adicionando com método `append`

Adiciona um único elemento como uma simples *string* até um objeto dos mais variados tipos, sendo que ele só aceita um tipo de objeto por vez.

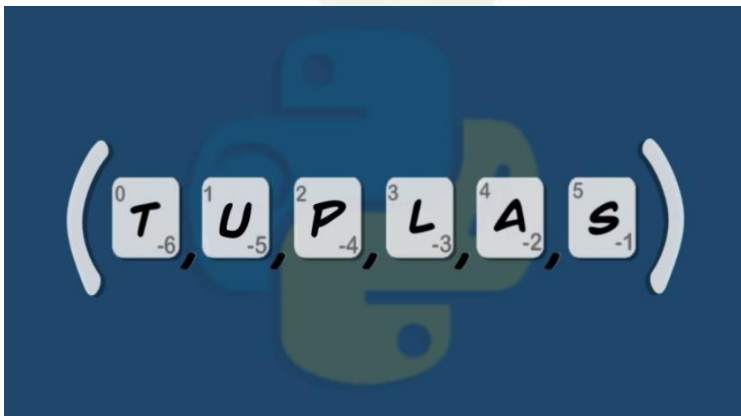


Tuplas

Assim como listas tuplas são sequencias ordenadas com zero ou mais objetos, porém, listas assim como *strings* são imutáveis.



50%



- ✓ Normalmente *tuplas* são criadas com os objetos entre parênteses () separados por vírgulas, ou somente com objetos separados por vírgulas
- ✓ O uso do parênteses () é o mais indicado para criação de *tuplas* por não causar ambiguidades e de melhor identificação visual.
- ✓ Os itens de uma tupla podem ser acessados da mesma maneira que os itens de uma lista usando os colchetes [].
- ✓ Tentar modificar um item de uma tupla gera erro, só é possível alterar um objeto dentro de uma tupla desde que esse objeto seja modificável

Exercícios Práticos

Tuplas em Python

Observação para uso de tuplas

As tuplas são sequências imutáveis, normalmente usadas para armazenar coleções de dados heterogêneos (como a tupla produzida pela função `enumerate` nativa).

Criando Tuplas

Normalmente tuplas são criadas com os objetos entre parênteses () separados por vírgulas, ou somente com objetos separados por vírgulas.

Acessando os itens de uma tupla

Os itens de uma tupla podem ser acessados da mesma maneira que os itens de uma lista usando os colchetes [].



Observação para uso de tuplas

As tuplas também são usadas para casos onde uma sequência imutável de dados homogêneos é necessária (como permitir o armazenamento em `set` ou `dict`).

Dicionários

Dicionários em Python

01

Dicionários são um coleção desordenada de objetos representados na forma de chave, valor onde a chave é usada para referenciar um determinado valor.

02

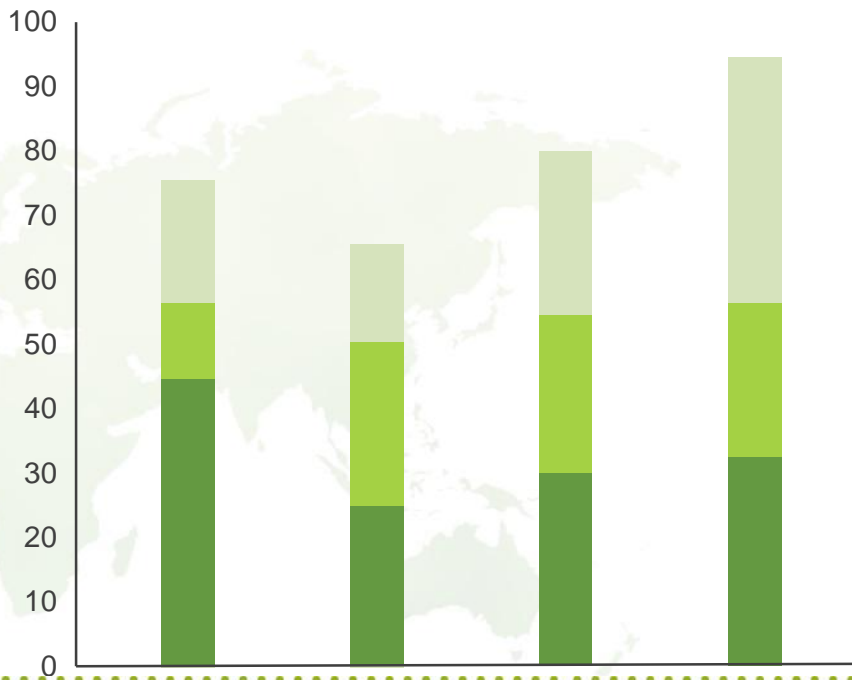
As chaves de um dicionário só podem ser de um tipo imutável como inteiros, *floats* e *strings*.

03

Dicionários não possuem uma noção de índice e não podem ser fatiados.

04

Dicionários são mutáveis de forma que a qualquer momento você pode inserir ou remover itens.



Exercícios

Dicionário

- Exemplo em Python:

```
1 d = {}  
2  
3 d['joao'] = 20  
4 d['maria'] = 30  
5 d['pedro'] = 25  
6  
7 print(d)
```

- Imprime:

- `{'maria': 30, 'pedro': 25, 'joao': 20}`



85%

60%

35%

Criando Dicionários

Dicionários são criados colocando os pares chave: valor entre chaves { }.

Acessando itens

Para acessar um item do dicionário devemos usar sua chave entre colchetes [].

Inserindo e removendo

Para inserir um item basta declarar o dicionário colocando entre colchetes a nova chave e atribuindo um valor. Use del aplicado ao nome do dicionário com a chave entre colchetes para apagar um par chave.

Conjuntos ou *sets*

Conjuntos em Python

```
# Exemplo de criação de sets.
```

```
numeros = [1, 2, 2, 3, 3, 3]
```

```
numeros_distintos = set()
```

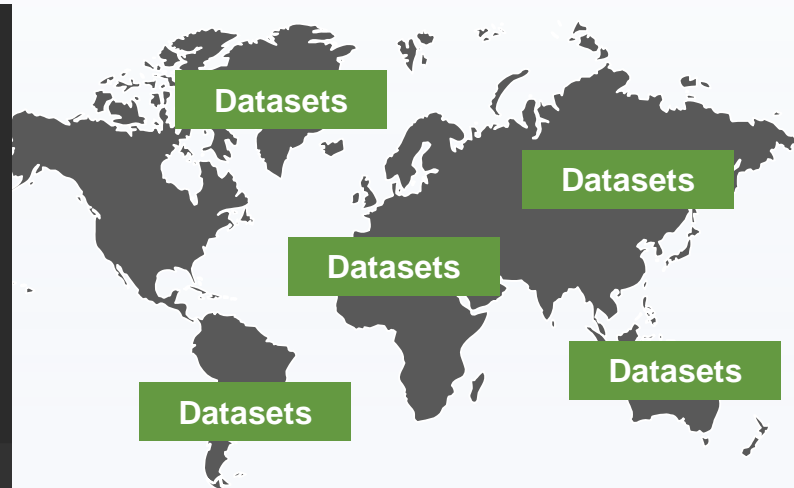
```
for num in numeros:
```

```
    numeros_distintos.add(num)
```

```
print("Números: ", numeros)
```

```
print("Números distintos: ", numeros_distintos)
```

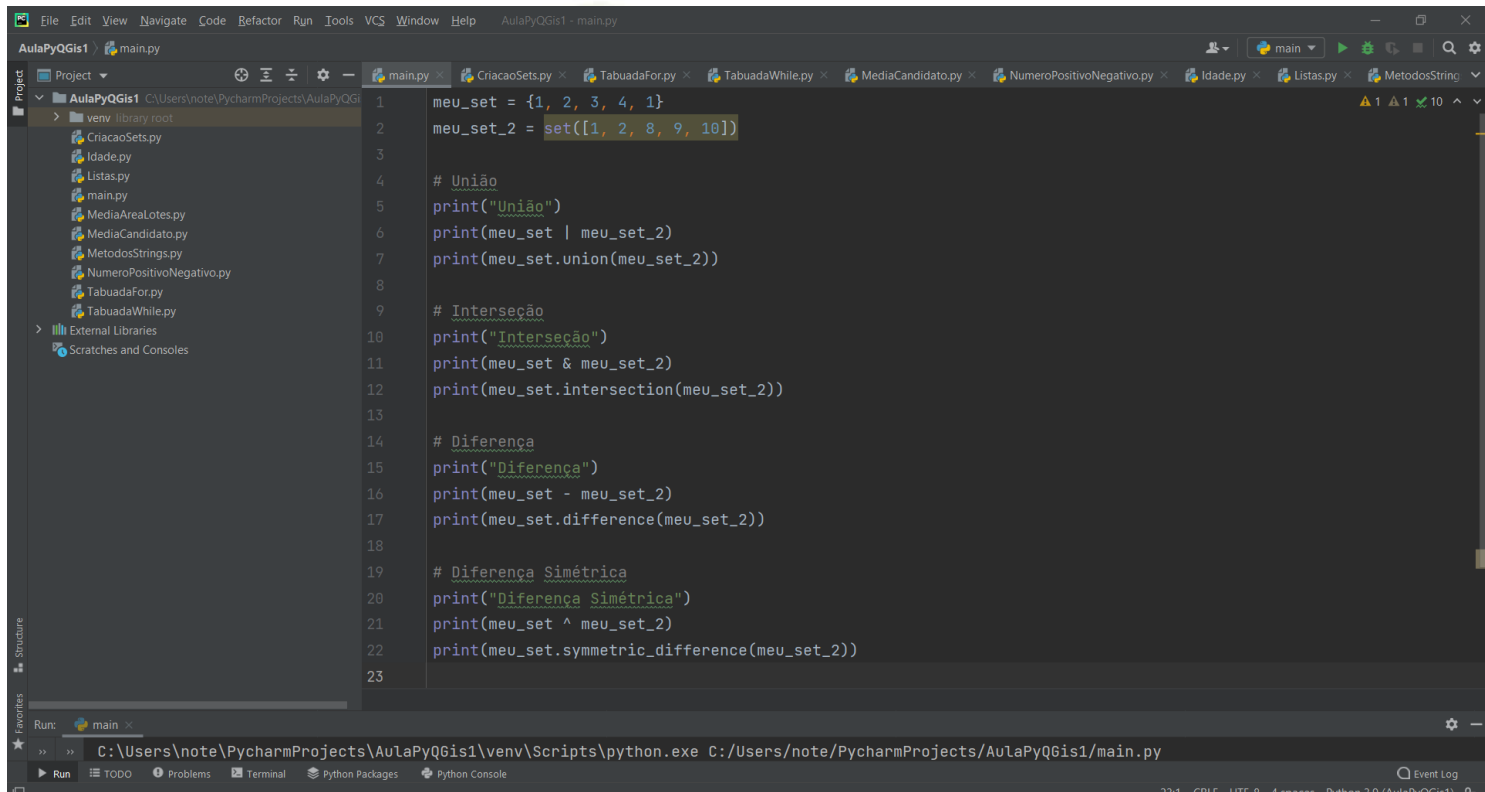
```
|
```



Um conjunto (set) é um tipo de dados de coleção, suportando o operador de associação in, a função *len()* e é iterável. Conjuntos não possuem noção de ordem por isso seus elementos não podem ser acessados com colchetes [] nem podem ser fatiados. Os conjuntos (set) não aceitam valores repetidos ao tentar criar um conjunto com valores repetidos eles serão descartados só sobrando um valor do mesmo.

Exercícios com sets

Os sets permitem operações matemáticas como união, interseção, diferença e diferença simétrica.



The screenshot shows a PyCharm IDE window titled 'AulaPyQGIS1 - main.py'. The left sidebar displays a project structure for 'AulaPyQGIS1' with files like 'CriacaoSets.py', 'Idade.py', 'Listas.py', 'main.py', 'MediaAreaLotes.py', 'MediaCandidato.py', 'MetodosStrings.py', 'NumeroPositivoNegativo.py', 'TabuadaFor.py', and 'TabuadaWhile.py'. The main editor area shows the code in 'main.py' with line numbers 1 through 23. The code defines two sets, 'meu_set' and 'meu_set_2', and performs various set operations: union, intersection, difference, and symmetric difference. The bottom status bar shows the command being executed: 'C:\Users\note\PycharmProjects\AulaPyQGIS1\venv\Scripts\python.exe C:/Users/note/PycharmProjects/AulaPyQGIS1/main.py'.

```
1 meu_set = {1, 2, 3, 4, 1}
2 meu_set_2 = set([1, 2, 8, 9, 10])
3
4 # União
5 print("União")
6 print(meu_set | meu_set_2)
7 print(meu_set.union(meu_set_2))
8
9 # Interseção
10 print("Interseção")
11 print(meu_set & meu_set_2)
12 print(meu_set.intersection(meu_set_2))
13
14 # Diferença
15 print("Diferença")
16 print(meu_set - meu_set_2)
17 print(meu_set.difference(meu_set_2))
18
19 # Diferença Simétrica
20 print("Diferença Simétrica")
21 print(meu_set ^ meu_set_2)
22 print(meu_set.symmetric_difference(meu_set_2))
23
```

Funções



Funções em Python são blocos de código que executarão algum tipo de tarefa ou manipulação de dados, podendo ou não receber: dados de entrada (parâmetros/argumentos). Parâmetros: são os nomes dados aos atributos que uma função pode receber.



Declarar função

Uma função em Python é criada através da declaração `def`.



Função é a primeira coisa a ser definida em programas Python.



Parâmetros

Podem receber valores denominados parâmetros (argumentos da função) e também retornar um valor.

Existem *docstrings* e tem um objetivo especial que é documentar os códigos, muito utilizado em modularização.

File Edit View Navigate Code Refactor Run Tools VCS Window Help AulaPyQGis1 - main.py

AulaPyQGis1 > main.py

Project

- AulaPyQGis1
- venv
- library
- root
- CriacaoSets.py
- Idade.py
- Listas.py
- main.py
- MediaAreaLotes.py
- MediaCandidato.py
- MetodosStrings.py
- NumeroPositivoNegativo.py
- OperacoesConjuntos.py
- TabuadaFor.py
- TabuadaWhile.py

External Libraries

Scratches and Consoles

```
1 # Programa que calcula a área de um retângulo, círculo ou triângulo
2
3 def retangulo(lado_a, lado_b):
4     """Calcula a área de um retângulo"""
5     area = lado_a * lado_b
6     return area
7
8 def triangulo(base, altura):
9     """Calcula a área de um triângulo"""
10    area = (base * altura) / 2
11    return area
12
13 def circulo(raio):
14    """Calcula
15    area = 3.14
16    return area
17
18 opcao = -1
19 while opcao !=
20     print('Esco
21     print()
22     print('1 -
23     print('2 -
24     print('3 -
```



Exercícios de Funções

Arquivos

Arquivos em Python

90%

Para trabalharmos com arquivos no Python usaremos o objeto *file*. Os objetos *file* contêm métodos e atributos que podem ser usados para coletar informações e manipular um arquivo.

Imagem



Existem bibliotecas específicas da linguagem Python, para usar funções prontas para fazer manipulação de imagens.

Vídeo



MoviePy é uma ótima biblioteca para criar seu próprio software de edição de vídeo.

Geolocalização



[GeoPandas](#) é um projeto de código aberto para facilitar trabalhos com dados geoespaciais em Python dos tipos geométricos.

Texto



Vamos ver como abrir arquivos de texto para ler os dados neles agora com exercícios.

Exercícios com Arquivos

Criar arquivos e seus diferentes modos de uso, para realizar as primeiras manipulações com arquivos textos.



Uma coisa fundamental e lembrarmos de fechar o arquivo se não ao sairmos do interpretador ou do programa em execução iremos perder todas as escritas feitas no arquivo.



Próxima Aula

