# Começando a Programar com o Python no QGIS

André Silva

GEOeduc

# CAPÍTULO 1:

## Introdução ao Python em QGIS

Atividades em geoprocessamento faz em sua maioria o uso de dados geográficos, e estes dados geográficos são usados para criar mapas com diferentes tipos de detalhamento, ou criar informações. Fazendo o uso de plataformas de geoprocessamento, os usuários podem automatizar muitas rotinas relacionados por exemplo ao software QGIS. A linguagem Python, é uma linguagem que está atualmente muito presente em diversas aplicações do nosso dia a dia, como Youtube, Instagram, Spotfify entre outros. Em geotecnologias está também sendo amplamente usada, facilitando assim a automatização de tarefas repetitivas.

É considerada uma linguagem de alto nível, pois é uma linguagem cuja sintaxe se aproxima mais da nossa linguagem e se distanciam mais da linguagem de máquina. Possui um nível de abstração que possibilita o entendimento do código mais facilmente.

Antes de iniciarmos nossa trilha de aprendizagem com Python em QGIS, iremos conhecer as principais sintaxes e como a linguagem em questão pode também ser executada, por usuários de um modo geral.

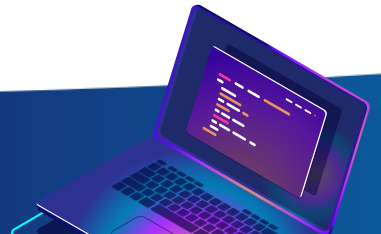
O Python possui uma baixa quantidade de palavras reservadas no código, faz o uso de endentação para marcar blocos (setores) de comandos e contém um coletor de lixo automático que gerencia a memória para os usuários da linguagem. Tudo visando facilitar a curva de aprendizagem e a usabilidade dos novos adeptos.

Seguem 8 vantagens de usar Python:

- Fácil aprendizado
- Simples de programar
- Sintaxe intuitiva
- Grande comunidade de usuários
- Documentação extensa
- Open Source
- Modularização
- Multiplataforma

Objetivos do capítulo:

- Conhecer às principais regras de sintaxe e de comandos dentro do terminal Python
- Apresentação de terminais alternativos ao PyQGIS para desenvolvimento



## O que é um Sintaxe?

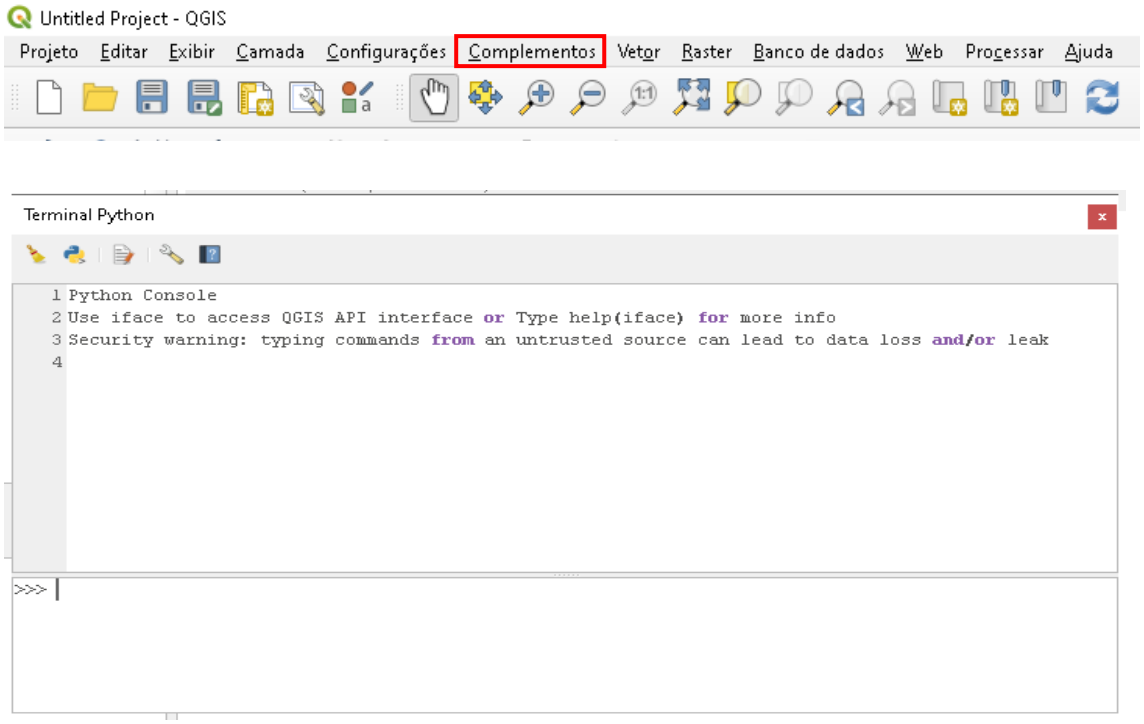
**Sintaxe** em linguagem de programação, está relacionado com as regras que determinam a composição de textos com significado em uma linguagem formal, isto é, os textos para os quais faz sentido definir o significado, ou fornecer uma interpretação.

**Dica:** Não se preocupe neste momento com instalações, o QGIS já vem com o Python e vamos usar o Console Python do QGIS inicialmente. E vamos utilizar o Console Python para aprender os conceitos básicos do Python.

## Console Pyhton

Iremos utilizar o Console Python para aprender os conceitos básicos de Python.

No QGIS, console Python pode ser iniciado usando CTRL+ALT+P ou menu **Complementos > Terminal Python**:



## Fundamentos e sintaxe 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...

```
>>> a = 29
>>> print(a)
29
>>> Praia = 'grande'
>>> print (Praia)
grande
>>> b = 2.4
>>> print(b)
2.4
```

## Regras para nome de variáveis:

Nomes das variáveis podem começar com uma letra, palavra ou um underline, mas **não** com números:

```
>>> x = 4
>>> _x = 2
>>> jose = 'maria'
>>> jose_4 = 'joao'
>>> 8v = 5
Traceback (most recent call last):
  File "D:\Programas\qgis\apps\Python37\lib\code.py", line 63, in runsource
    code = self.compile(source, filename, symbol)
  File "D:\Programas\qgis\apps\Python37\lib\codeop.py", line 168, in __call__
    return _maybe_compile(self.compiler, source, filename, symbol)
  File "D:\Programas\qgis\apps\Python37\lib\codeop.py", line 99, in _maybe_compile
    raise err1
  File "D:\Programas\qgis\apps\Python37\lib\codeop.py", line 87, in _maybe_compile
    code1 = compiler(source + "\n", filename, symbol)
  File "D:\Programas\qgis\apps\Python37\lib\codeop.py", line 133, in __call__
    codeob = compile(source, filename, symbol, self.flags, 1)
  File "<input>", line 1
    8v = 5
     ^
SyntaxError: invalid syntax
```

O nome das variáveis são sensíveis a letras maiúsculas, por exemplo: x é diferente de X:

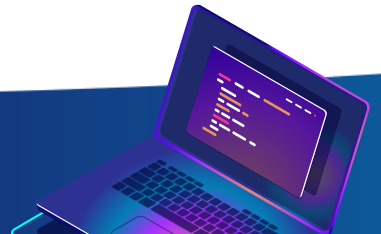
```
>>> x = 28
>>> y  = X*5
Traceback (most recent call last):
  File "D:\Programas\qgis\apps\Python37\lib\code.py", line 90, in runcode
    exec(code, self.locals)
  File "<input>", line 1, in <module>
NameError: name 'X' is not defined
```

Dica: As palavras abaixo são reservadas da linguagem Python e não devem ser usadas como nome de variáveis.

| | | | |
|---|---|---|---|
| and | except | lambda | with |
| as | finally | nonlocal | while |
| assert | False | None | yield |
| break | for | not | |
| class | from | or | |
| continue | global | pass | |
| def | if | raise | |
| del | import | return | |
| elif | in | True | |
| else | is | try | |

Tipos numéricos: Python possui primitivamente os tipos numéricos de números inteiros (int), ponto flutuante (float) e complexo (complex).

```
>>> a = 1
>>> b = 2.4
>>> c = 4 + 3j
```



Para descobrir qual o tipo de variável que está armazenada, use a função *type*.

```
>>> a = 1
>>> b = 2.4
>>> c = 4 + 3j
>>> type(a)
<class 'int'>
>>> type(b)
<class 'float'>
>>> type(c)
<class 'complex'>
```

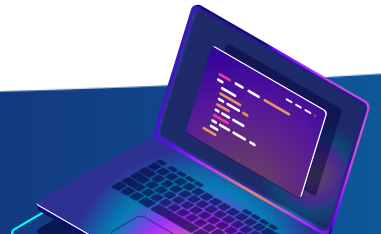
No console do Python no QGIS, assim como também em outros consoles, é possível fazer o uso dele como se fosse uma calculadora.

```
>>> 2 + 2
4
>>> 2**8
256
>>> a = 134
>>> b = 3
>>> print (a + b)
137
>>> (50 - 5*6)/4
5.0
```

Na linguagem Python, assim como em outras linguagens de programação, letras, palavras e textos são conhecidos como sendo do tipo String, ou str, se usada a função *type*.

```
>>> nome = 'Maria'
>>> letra = 'r'
>>> type(nome)
<class 'str'>
>>> type(letra)
<class 'str'>
```

Um objeto da classe string nada mais é do que uma matriz de valores sequenciados onde o primeiro valor (caractere) corresponde ao índice 0 da matriz e o último valor ao índice -1, o penúltimo -2 e assim sucessivamente.



```
>>> nome = 'Maria'
>>> nome[0]
'M'
>>> nome[1]
'a'
>>> nome[2]
'r'
>>> nome[-1]
'a'
>>> nome[-2]
'i'
```

Você pode usar também uma função chamada **len()**, para que possa ver retornado o comprimento de um valor do tipo string.

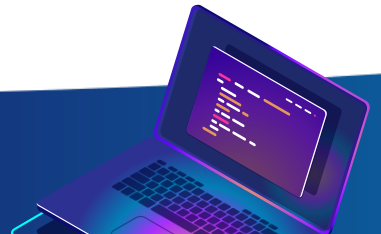
```
>>> frase = "você é uma pessoa muito legal"
>>> len(frase)
29
```

**Listas:** Lista é uma estrutura que faz uso de itens organizados em série, na qual cada item pode ser acessado a partir de um índice, que representa sua posição dentro da lista. Estes índices sempre se iniciam por 0. Listas são representadas por itens separados por vírgula e dentro de colchetes, logo uma lista vazia é representada por colchetes sem nenhum conteúdo [].

```
>>> lista_num = [521,568,952,258]
>>> lista_texto = ['miguel','jose','carro','virus']
>>> texto = ['manga',522,'pao',1001,'interesse']
```

Como já citado anteriormente e como também podemos ver em valores do tipo string, os itens da lista podem ser acessados usando índices.

```
>>> lista_num = [521,568,952,258]
>>> lista_texto = ['miguel','jose','carro','virus']
>>> texto = ['manga',522,'pao',1001,'interesse']
>>> lista_num[2]
952
>>> lista_texto[3]
'virus'
>>> texto[0]
'manga'
>>> texto[1:3]
[522, 'pao']
```



Os elementos de uma lista podem ser modificados.

```
>>> print(texto)
['manga', 522, 'pao', 1001, 'interesse']
>>> texto[0] = 1250
>>> print(texto)
[1250, 522, 'pao', 1001, 'interesse']
>>> texto[2]='bolo'
>>> print(texto)
[1250, 522, 'bolo', 1001, 'interesse']
```

É possível também adicionar novos itens dentro de uma lista, usando a função **append()**.

```
>>> texto
[1250, 522, 'bolo', 1001, 'interesse']
>>> texto.append('copo')
>>> texto
[1250, 522, 'bolo', 1001, 'interesse', 'copo']
>>> texto.append(145)
>>> texto
[1250, 522, 'bolo', 1001, 'interesse', 'copo', 145]
```

Ao criar uma lista vazia em Python, é possível adicionar itens com o método **extend()**.

```
>>> lista=[]
>>> lista
[]
>>> lista.extend([1,2,3])
>>> lista
[1, 2, 3]
```

Adicionar um item de valor 35 na posição predeterminada (3) com o método **insert()**.

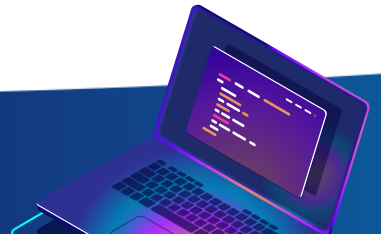
```
>>> lista.insert(3,35)
>>> lista
[1, 2, 3, 35]
```

Remover um item da lista na posição (2) com o método **remove()**.

```
>>> lista.remove(2)
>>> lista
[1, 3, 35]
```

Copiando o conteúdo de uma lista para outra lista, com o método **copy()**.

```
>>> lista2 = lista.copy()
>>> lista2
[1, 3, 35]
```



Reverter a ordem dos itens de uma lista é muito simples, basta usar o método **reverse()**.

```
>>> lista2.reverse()
>>> lista2
[35, 3, 1]
```

É possível reorganizar novamente a lista, por ordem crescente usando o método **sort()**.

```
>>> lista2.sort()
>>> lista2
[1, 3, 35]
```

O método **count()** retorna o número de vezes que um determinado item aparece na lista

```
>>> lista3=[1,2,4,4,4,5,5,6,9,10]
>>> lista3.count(4)
3
```

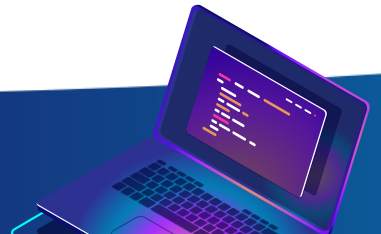
Com o método **clear()**, é possível remover todos os itens da lista.

```
>>> lista3.clear()
>>> lista3
[]
```

É possível também usar a instrução **del** para deletar itens de uma lista usando índices em vez de valores.

```
>>> lista3=[1,2,4,4,4,5,5,6,9,10]
>>> del lista3[2]
>>> lista3
[1, 2, 4, 4, 5, 5, 6, 9, 10]
>>> del lista3[5:8]
>>> lista3
[1, 2, 4, 4, 5, 10]
```

**Tuplas:** é um tipo de lista em que não é possível fazer mudanças. Em outras palavras, diferentemente da estrutura lista, ensinada anteriormente, uma tupla, após ter seus valores definidos, não permite a adição ou remoção de elementos.



```
>>> exem_tupla = 'banana',3,45,False,'Aline'
>>> exem_tupla
('banana', 3, 45, False, 'Aline')
>>> exem_tupla[0]
'banana'
>>> exem_tupla[1:3]
(3, 45)
```

**Sets:** Também conhecidos como conjuntos também é conhecida como uma sequência utilizada na linguagem Python. A sintaxe dos sets são definidas entre chaves {} e que se resume apenas a aparição de cada um dos seus itens, que neste caso não são indexados.

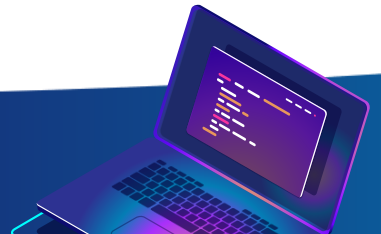
```
>>> grupo = {'teste', 'prova', 'lar', 'cozinha'}
>>> grupo
{'teste', 'prova', 'lar', 'cozinha'}
>>> 'prova' in grupo
True
>>> 'maresia' in grupo
False
```

**Dica:** Os sets não retornam console, valores repetidos.

```
>>> elementos = {'teste','ar','alunos','alunos','ferradura','bosque','bosque'}
>>> elementos
{'alunos', 'ferradura', 'teste', 'ar', 'bosque'}
>>> numeros={1,2,3,3,4,4,5,'a'}
>>> numeros
{1, 2, 3, 4, 5, 'a'}
```

**Dicionários:** é um tipo diferente de coleção, ou seja, um objeto que contém mais que um valor. Diferente das listas em que os dados são acessados por índices, nos dicionários o acesso às informações ocorre através de chaves únicas.

```
>>> dicionario={'nome':'andre', 'sobrenome':'silva'}
>>> dicionario['idade']=51
>>> dicionario
{'nome': 'andre', 'sobrenome': 'silva', 'idade': 51}
>>> del dicionario['idade']
>>> dicionario
{'nome': 'andre', 'sobrenome': 'silva'}
```



## Controles de fluxo em Python

Assim como em outras linguagens de programação, python também utiliza controles de fluxo de programa. Porém antes de aprendermos sobre controles de fluxo, precisamos falar um pouco sobre indentação de script.

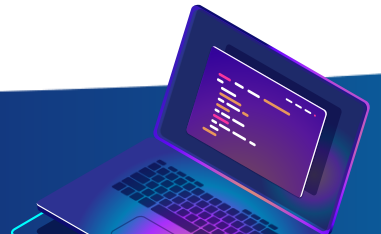
Em Python é necessário o uso de indentação de blocos, uma vez que os blocos de códigos não são separados por chaves {}, então em python, uma classe, função, ou controle de fluxo, deve possuir endentação. A indentação em Python, permite também obter um código mais organizado e intuitivo de se compreender, o que facilita também o uso da linguagem.

**If, elif e else:** O if é uma instrução das mais conhecidas em programação. Ela 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).

```
x = int(input("Diga um número inteiro: "))

Digite um número inteiro:  4

if x<0:
    print('O número é negativo')
elif x>0:
    print('O número é positivo')
else:
    print('O número é zero')

O número é positivo
```

**For:** A instrução **for** se caracteriza como "executar/repetir as instruções nos termos predefinidos". Como é possível verificar abaixo a repetição é feita para cada item da lista e a palavra e o comprimento dela é retornado como resultado.

```
strings = ['Oi!','Vamos','python','agora']
for palavra in strings:
    print(palavra,len(palavra))
>>> exec(open('C:/Users/ANDRSO~1/AppData/Local/Temp/tmpxo6o3sgu.py'.encode('utf-8')).read())
Oi! 3
Vamos 5
python 6
agora 5
```



**While:** é definido com “enquanto o que foi predefinido não ocorrer, vai executando ou repetindo”.

```
conta = 0
while(conta <= 10):
    conta += 1
    print(conta)

```

```
1
2
3
4
5
6
7
8
9
10
11
```

**Range:** forma de interação que Python possui com uma série numérica é usando a função range().

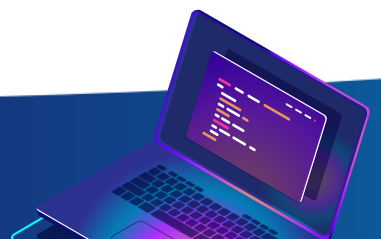
```
for i in range (10):
    print(i)

```

```
0
1
2
3
4
5
6
7
8
9
```

**Dica:** Digamos que você não sabe a quantidade de itens em uma lista, mas queremos listar cada um destes itens. Use o range para lhe auxiliar.

```
lista = ['Oi','Vamos','começar','python?']
for i in range(len(lista)):
    print(lista[i],i)

```

```
Oi 0
Vamos 1
começar 2
python? 3
```



É possível também definir o início e o final do range() e também o passo.

```
>>> list(range(100,115))
[100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114]
>>> list(range(10,5,-1))
[10, 9, 8, 7, 6]
>>> list(range(-100,-50,10))
[-100, -90, -80, -70, -60]
```

**Break, else em loops e continue:** O comando **break** quebra/pára a execução for ou while. As repetições também podem ter uma instrução **else**; ela é executada quando a repetição termina, mas quando ela é interrompida por uma instrução **break**.

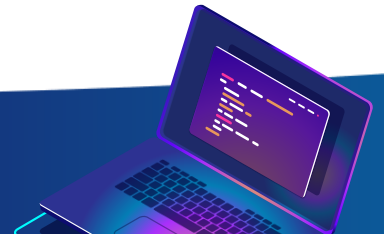
```
for n in range(11,21):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'é igual a', x, '*', n//x)
            break
    else:
        print(n, 'é um número primo')
```

```
11 é um número primo
12 é igual a  2 * 6
13 é um número primo
14 é igual a  2 * 7
15 é igual a  3 * 5
16 é igual a  2 * 8
17 é um número primo
18 é igual a  2 * 9
19 é um número primo
20 é igual a  2 * 10
```

Já na instrução **continue**, avança para a próxima interação da repetição:

```
for num in range(2, 9):
    if num % 2 == 0:
        print("Numero par", num)
        continue
    print("Número ímpar", num)
```

```
Numero par 2
Número ímpar 3
Numero par 4
Número ímpar 5
Numero par 6
Número ímpar 7
Numero par 8
```



**Funções:** é denominado como um conjunto de instruções reusáveis e organizados para executar uma tarefa. Na linguagem de programação Python, é definido uma função usando **def** seguido do nome da função e dos argumentos ou parâmetros passados dentro de parênteses.

```
def tabuada7()
    n=1
    while n < 11:
        print(n*7, end =' ')
        n=n+1

tabuada7()

7 14 21 28 35 42 49 56 63 70
```

## Terminais alternativos ao PyQGIS para desenvolvimento

Terminais alternativos são também conhecidos como IDEs, e são também conhecidos como um ambiente de desenvolvimento integrado. Em outras palavras é um software para criar aplicações, combinando ferramentas comum de desenvolvimento em uma interface gráfica.

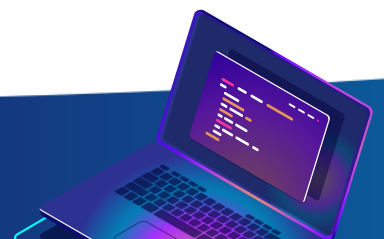
Uma IDE em geral é composta por três partes:

**Editor de código fonte:** É também um editor de textos que auxilia na criação dos códigos e scripts, realçando de modo intuitivo as palavras reservadas da linguagem (sintaxe), capacidade de verificação de bugs e preenchimento automático.

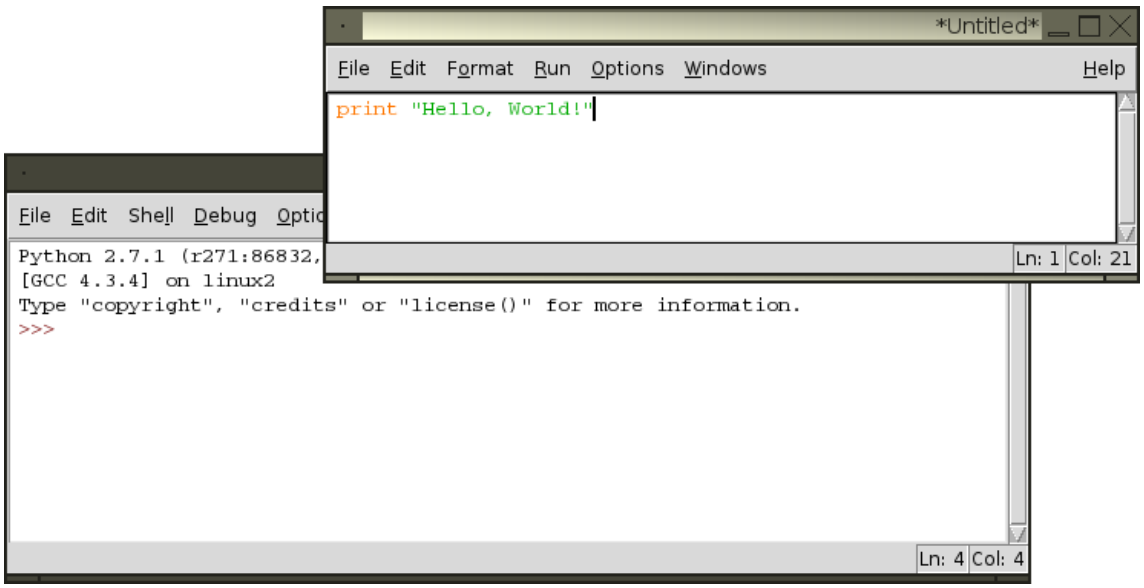
**Automação de compilação local:** Automatiza tarefas simples e repetitivas, responsável também por compilar um código-fonte em código binário, em outras palavras traduzir uma linguagem humana(scripts de uma linguagem) em código de máquina.

**Debugger:** Usado para testar outros programas ou scripts, e mostrar graficamente a localização do bug no código atual.

Algumas IDEs que podem ser usadas alternativas ao PyQGIS:

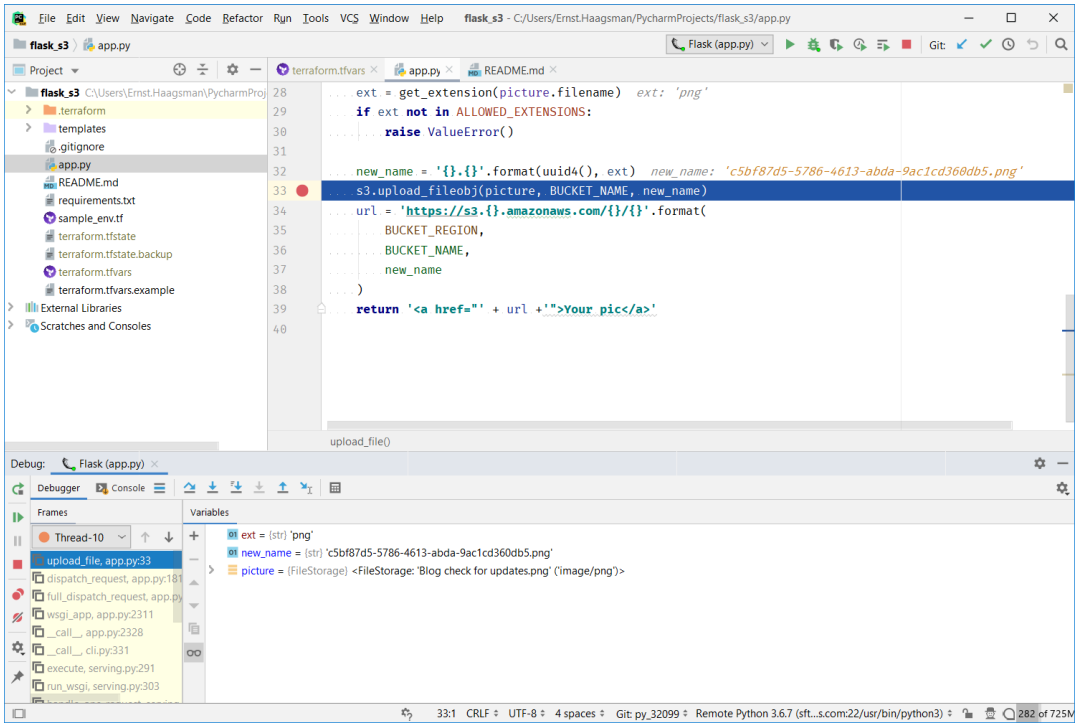


**IDLE:** É uma plataforma que já vem instalada junto com o Python. É bem simples de ser usada também.

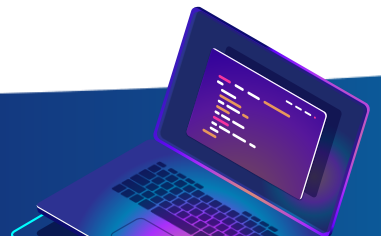



IDLE Python – Fonte: Python.org.br

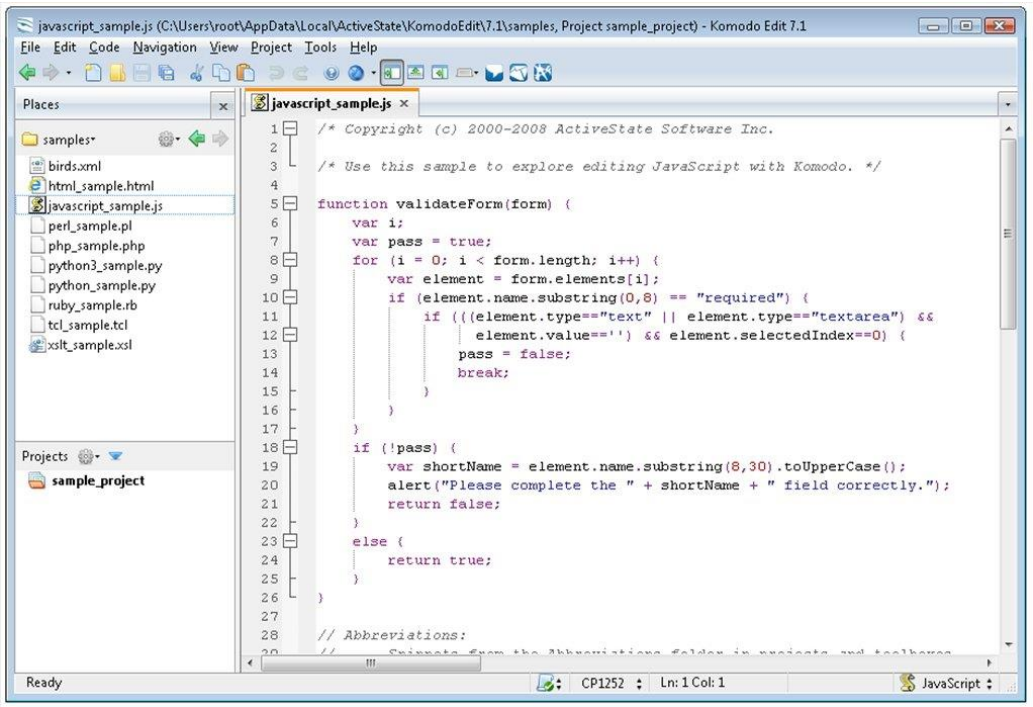
**Pycharm Community:** Disponibilizada pela JetBrains. Essa é uma IDE que proporciona análise do código, depurador gráfico, testador de unidade integrado e também suporta desenvolvimento para Web.



Pycharm – Fonte: lifewire.com

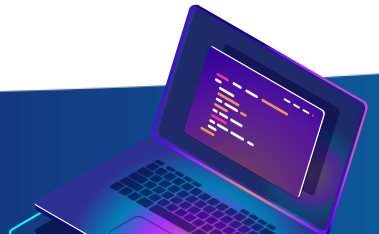


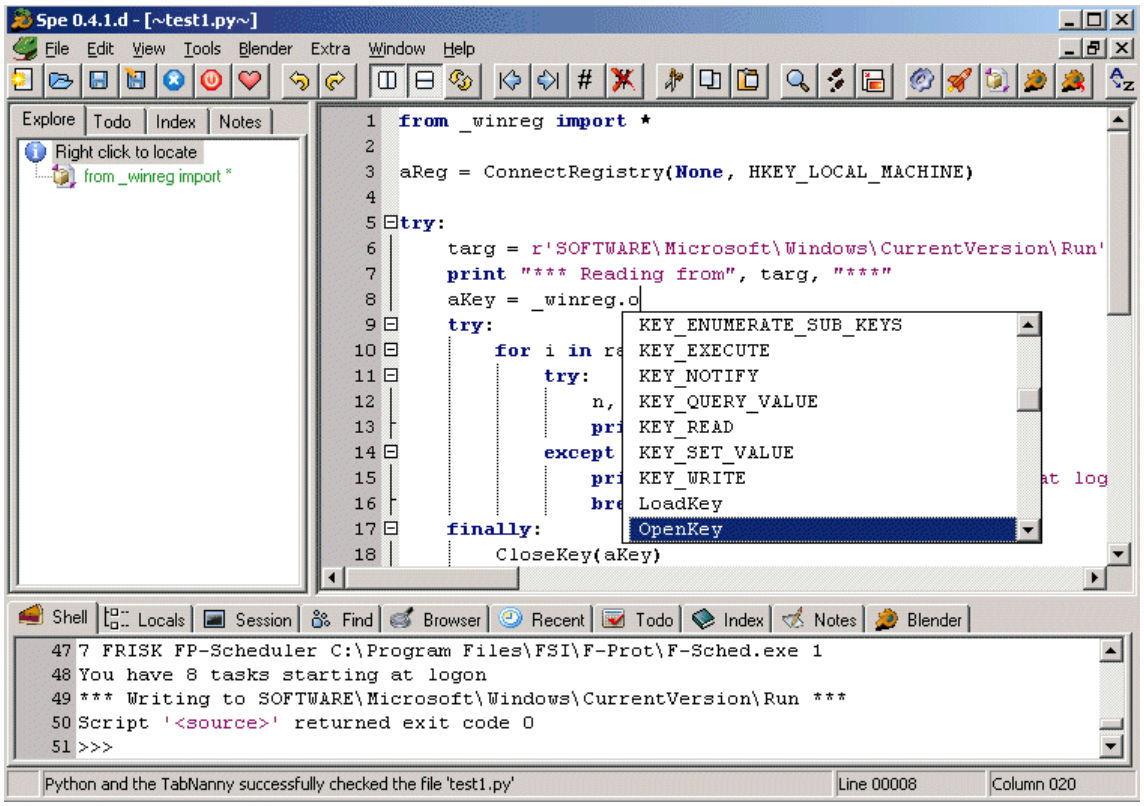
**Komodo Edit:** é uma ótima opção de editor, com muitas variedades em recursos tais como autocomplete, calltips, multi-language file support, syntax coloring, syntax checking, Vi emulation, Emacs key bindings e outros.



Komodo – Fonte: malvida.com

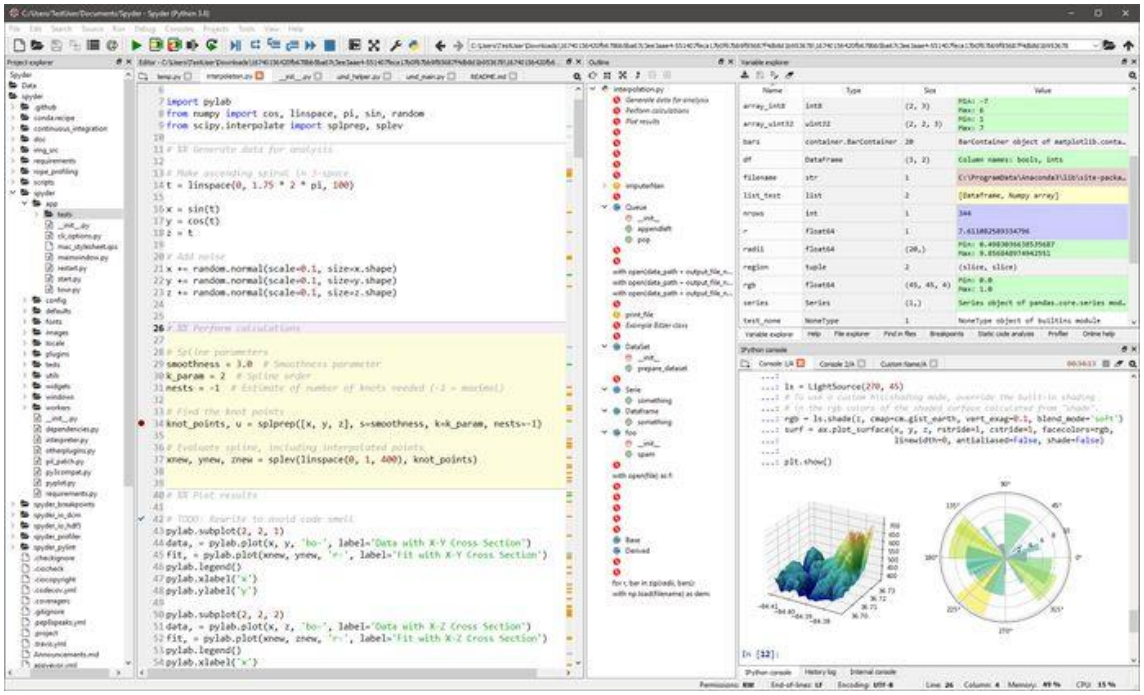
**SPE:** Desenvolvido com wxPyhton e possui algumas funcionalidades adicionais com o wxGlade, plugin para desenho de telas gráficas.



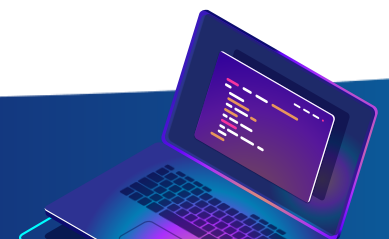



SPE – Fonte: old.zope.org

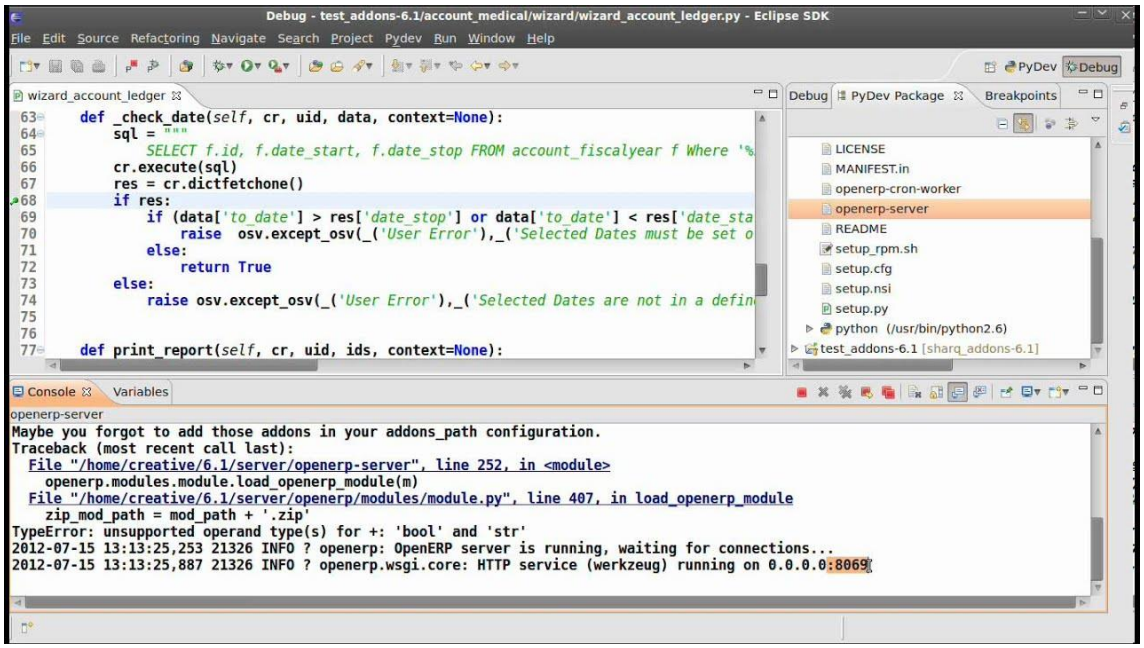
**Spyder:** Também conhecido como Pydee, é uma poderosa IDE para a linguagem Python com edição avançada, testes interativos, recursos de depuração e introspecção.



Spyder – Fonte: opensource.com

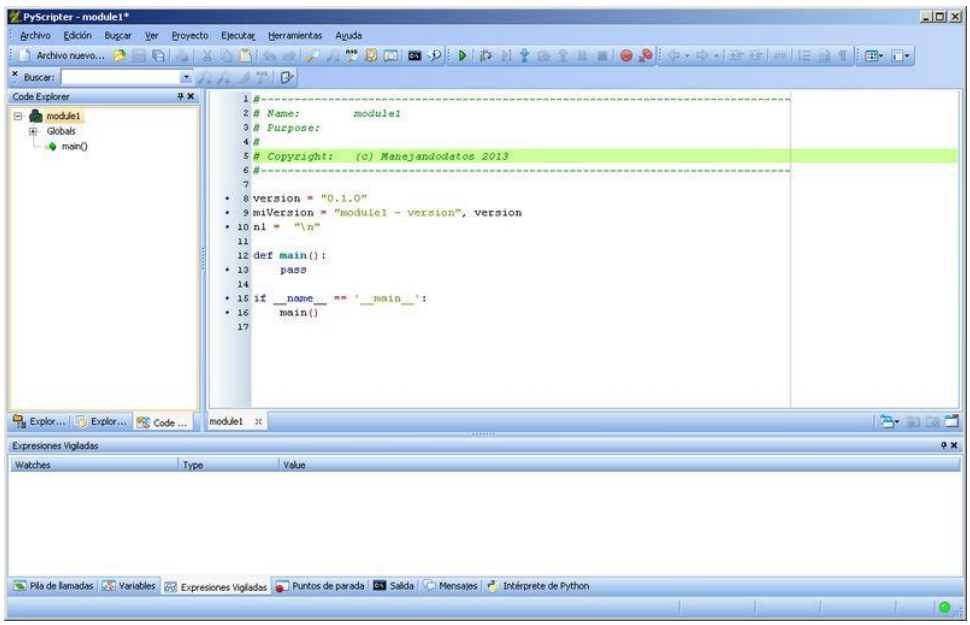


**Eclipse:** Diferente de todos os outros. É um software pesado, e grande, mas muito poderoso. É feito em Java e é ideal para desenvolvimento Java. Mas existem plugins para se desenvolver em Python.

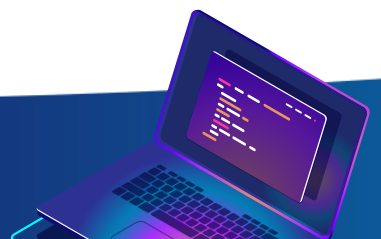



Eclipse – Fonte: youtube.com

**PyScripter:** Acompanha o conjunto de componentes python para Delphi (embora não requeira que o Delphi esteja instalado para funcionar). Suporta debugging, auto-complete, navegação no código entre outros recursos.




PyScripter – Fonte: pinterest.com



## EXERCÍCIO 1A: FUNCIONALIDADES BÁSICAS DO QGIS

Este exercício tem como finalidade verificar a aprendizagem do primeiro capítulo. Durante essa jornada, foi possível verificar a importância das variáveis, conhecer estruturas de repetição, estruturas condicionais, funções e entre outros. Abaixo vamos verificar como foi este processo de absorção de conhecimento.

### Passo 1 – Verdadeiro ou Falso

1) As linguagens de programação, no tocante a sintaxe, não diferem uma das outras.

   a) Verdadeiro
   b) Falso

2) O QGIS já possui um console python devidamente instalado.

   a) Verdadeiro
   b) Falso

3) Uma variável pode ser iniciada com um número.

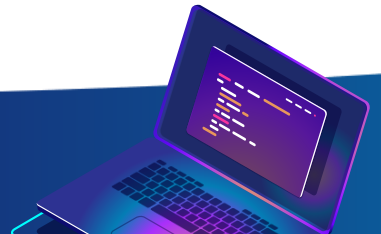
   a) Verdadeiro
   b) Falso

4) Na linguagem python, o números sem casas decimais, são chamados de float.

   a) Verdadeiro
   b) Falso

5) Para que eu possa descobrir o tipo de uma variável em Python, eu devo usar uma função chamada reload().

   a) Verdadeiro
   b) Falso



## Passo 2 – Listas

1) Na lista chamada *frutas = [‘maça’, ‘banana’, ‘mamão’, ‘melão’],* como é possível acessar o valor *mamão*?

2) Se referindo a mesma lista acima, como eu poderia acessar o valor de do último elemento?

*3)* No nome *José,* a letra *s*, ou seja a terceira letra do nome possui um índice de qual valor*?*

4) Qual a função responsável por fazer a contagem das letras de uma palavra, frase ou texto.

5) Qual a função do método extend()?

6) Qual a sintaxe ou seja qual código é necessário para incluir um número na posição 5 de uma lista chamada *numbers*?

7) Qual a função pode ser usada para organizar números inteiros em ordem decrescente?

8) Qual a função que pode ser usada para limpar uma lista?

9) O que são sets e como eles são representados?

10) Em quais estruturas o comando **Break** pode ser utilizado?

11) Para que server um Debugger em uma IDE (Ambiente de desenvolvimento)?