Programação Básica - Python

Introdução e Fundamentos aos Principais Conceitos de Programação Utilizando Linguagem Python

Elaborado por:

Gustavo Gil de Oliveira Silva

Outubro – 2018

ORIENTAÇÕES INICIAIS

O objetivo desta apostila é servir de material complementar às aulas do curso de programação básica da GeoDrones EAD. Através da linguagem Python, os conceitos fundamentais de programação serão levantados e trabalhados, para que o aluno seja capaz de desenvolver seus próprios algoritmos e entender como os softwares funcionam. Ao final de cada capítulo uma lista de exercício é apresentada e recomenda-se fortemente que ela seja feita, pois assim o aluno pode aprimorar e desenvolver sua capacidade de programar e aplicar os conhecimentos adquiridos durante o curso.

Sumário[[1]](#footnote-1)

[1. Introdução 7](#_Toc527362119)

[1.1. Função print() 8](#_Toc527362120)

[1.2. Comentários # e ‘’’ ‘’’ 8](#_Toc527362121)

[2. Variáveis 8](#_Toc527362122)

[2.1. Conceitos Exemplos e Observações 8](#_Toc527362123)

[2.2. Função input() 10](#_Toc527362124)

[2.3. Função int(), float() e str() 11](#_Toc527362125)

[2.4. Exercícios Práticos 11](#_Toc527362126)

[2.5. Resolução dos Exercícios Práticos 12](#_Toc527362127)

[3. Condicionais 15](#_Toc527362128)

[3.1. Conceitos, Exemplos e Observações 15](#_Toc527362129)

[3.2. Operadores de Comparação 17](#_Toc527362130)

[3.3. E se não 18](#_Toc527362131)

[3.4. Indentação 19](#_Toc527362132)

[3.5. Exercícios Práticos 19](#_Toc527362133)

[3.6. Resolução dos Exercícios Práticos 20](#_Toc527362134)

[4. Booleanos 23](#_Toc527362135)

[4.1. Conceitos, Exemplos e Observações 23](#_Toc527362136)

[4.2. Números Binários 24](#_Toc527362137)

[4.3. Exercícios Práticos 25](#_Toc527362138)

[4.4. Resolução dos Exercícios Práticos 25](#_Toc527362139)

[5. Laços de Repetição 27](#_Toc527362140)

[5.1. Conceitos, Exemplos e Observações 27](#_Toc527362141)

[5.1.1. While 27](#_Toc527362142)

[5.1.2. For 29](#_Toc527362143)

[5.2. Exercícios Práticos 31](#_Toc527362144)

[5.3. Resolução dos Exercícios Práticos 32](#_Toc527362145)

[6. Funções 34](#_Toc527362146)

[6.1. Conceitos, Exemplos e Observações 34](#_Toc527362147)

[6.2. Exercícios 37](#_Toc527362148)

[6.3. Resolução dos Exercícios Práticos 38](#_Toc527362149)

[7. Python e QGis 39](#_Toc527362150)

[8. Fontes Bibliográficas 42](#_Toc527362151)

Lista de Tabelas[[2]](#footnote-2)

[Tabela 1 - Operadores de Comparação 17](#_Toc527362152)

[Tabela 2 - Tabela de Operadores Lógicos 24](#_Toc527362153)

[Tabela 3 - Tabela de Conversão Decimal para Binário 24](#_Toc527362154)

Lista de Figuras[[3]](#footnote-3)

[Figura 1 - Exemplo da Função print() 8](#_Toc527362155)

[Figura 2 - Exemplo de Comentários 8](#_Toc527362156)

[Figura 3 - Exemplo Simples do Uso de uma Variável 9](#_Toc527362157)

[Figura 4 - Analogia Caixa e Variável 9](#_Toc527362158)

[Figura 5 - Exemplo Simples do Uso de uma Variável (2) 9](#_Toc527362159)

[Figura 6 - Exemplo de Como um Valor se Sobreescreve em Uma Variável 10](#_Toc527362160)

[Figura 7 – Exemplo de Como Realizar Operações de Uma Variável com Ela Mesma 10](#_Toc527362161)

[Figura 8 - Exemplo da Função input() 11](#_Toc527362162)

[Figura 9 - Exemplo das Funções int(), float() e str() 11](#_Toc527362163)

[Figura 10 - Fluxograma Exemplo de Condicionais 15](#_Toc527362164)

[Figura 11 – Exemplo de Condicional 1 16](#_Toc527362165)

[Figura 12 - Resposta do Exemplo de Condicional 1 16](#_Toc527362166)

[Figura 13 - Exemplo de Condicional 2 16](#_Toc527362167)

[Figura 14 - Resposta do Exemplo de Condicional 2 16](#_Toc527362168)

[Figura 15 – Fluxograma Exemplo de Condicionais 2 17](#_Toc527362169)

[Figura 16 - Exemplo de Condicional 3 18](#_Toc527362170)

[Figura 17 - Resposta do Exemplo de Condicional 3 18](#_Toc527362171)

[Figura 18 - Exemplo de Condicional 4 18](#_Toc527362172)

[Figura 19 - Resposta 1 do Exemplo de Condicional 4 18](#_Toc527362173)

[Figura 20 - Resposta 2 do Exemplo de Condicional 4 18](#_Toc527362174)

[Figura 21 - Exemplo do Uso de Código Indentado 19](#_Toc527362175)

[Figura 22 - Exemplo de Valor Booleano True 23](#_Toc527362176)

[Figura 23 - Exemplo de Valor Booleano False 23](#_Toc527362177)

[Figura 24 - Exemplo de Operador Booleano AND 23](#_Toc527362178)

[Figura 25 - Exemplo 1 de Laço de Repetição While 27](#_Toc527362179)

[Figura 26 - Exemplo 2 de Laço de Repetição While Imprimindo a Variável de Controle 28](#_Toc527362180)

[Figura 27 - Fluxograma para Exemplificar Laço de Repetição While 28](#_Toc527362181)

[Figura 28 - Exemplo Laço While de Dizer o Nome 29](#_Toc527362182)

[Figura 29 - Exemplo Laço For 30](#_Toc527362183)

[Figura 30 - Exemplo Equivalente While/For 30](#_Toc527362184)

[Figura 31 – Exemplo Laço For com Intervalo Diferente 30](#_Toc527362185)

[Figura 32 - Exemplo Laço For Incrementando 2 31](#_Toc527362186)

[Figura 33 - Exemplo Laço For Incrementando 31](#_Toc527362187)

[Figura 34 - Exemplo 1 do Uso de Funções 34](#_Toc527362188)

[Figura 35 - Exemplo de Função com Entrada 35](#_Toc527362189)

[Figura 36 - Exemplo de Função com Entrada e Saída 36](#_Toc527362190)

[Figura 37 - Iniciando Console Python na Plataforma QGis 40](#_Toc527362191)

[Figura 38 - Console Python na Plataforma QGis 40](#_Toc527362192)

[Figura 39 - Editor Python na Plataforma QGis 41](#_Toc527362193)

[Figura 40 - Resultado Após O Uso da Função *iface.zoomFull()* 41](#_Toc527362194)

# Introdução

Antes de qualquer informação é interessante ressaltar que aprender a programar vai além de entender o funcionamento de softwares e programas de computadores. Quando você aprende o raciocínio lógico de programação seu raciocínio do cotidiano, sua capacidade de resolver problemas, sua criatividade, entre outros fatores elevam significantemente.

A linguagem de programação trabalhada nesse curso é chamada de Python, que é uma das linguagens que mais vem sendo utilizadas nos últimos anos. É interessante levantar que existem duas versões de Python que são bem utilizadas: Python 2 e Python 3. Nesse curso será utilizada a versão 3, pois é mais atual e moderna, mas não se preocupe, inicialmente a diferença entre essas duas linguagens será irrelevante, pois para os conceitos básicos elas pouco se diferem.

É possível baixar Python gratuitamente para Windows, Mac OS X, Linux entre outros através do site: <https://www.python.org/downloads/>. Após feita a instalação pesquise em sua máquina o arquivo *IDLE (Python 3.X XX-bit),* onde será executado os códigos.

Após feita a instalação, deve-se procurar (pode pesquisar na barra de pesquisa do Windows) o terminal IDLE (*Integrated Development and Learning Environment*) do Python. Esse é o terminal onde vamos executar os códigos e fazer os testes. Nessa mesma plataforma também é possível abrir um novo arquivo onde códigos mais complexos podem ser rascunhados e executados, basta clicar em File>>New File. Para rodar o código é só apertar F5, salvar e ver o resultado no terminal. Para abrir um código já salvo, basta abrir o terminal IDLE do Python clicar em File>>Open e pronto.

Durante as explicações pode ser que apareçam tópicos que vão ensinar alguns comandos extras que devem ser aprendidos para que novas aplicações possam ser criadas. Na seção seguinte, um exemplo de comando (print()) vai ser apresentado para que o curso possa ser continuado.

No final de cada capítulo vai existir uma lista de exercícios, que deve ser feita para que o raciocínio lógico de programação possa ser trabalhado e desenvolvido. O foco na solução dos exercícios é a didática, ou seja, será focado a melhor solução que faça o aluno entender o conceito, assim talvez a solução presente não será a mais eficiente e sim a mais clara. Vale lembrar também que erros em programação são normais, e existem muitas formas de repará-los, como pesquisar no Google a mensagem de erro que está sendo mostrada, perguntar no fórum de discussão da plataforma EAD da GeoDrones ou pesquisar em sites especializados nesse tipo de problema como <https://stackoverflow.com/> e <https://www.reddit.com/>.

## Função print()

Ao digitar o comando *print()* no terminal IDLE do Python tem-se que a informação entre parênteses seja apresentada na tela. Observe o exemplo:

Figura 1 - Exemplo da Função print()



Note que a palavra entre parênteses é mostrada na tela, pois o comando print mostra tudo que está entre aspas. É possível também utilizar aspas simples para mostrar uma mensagem, mas por enquanto será utilizada a dupla.

O comando print também pode imprimir valores que estão armazenados em variáveis, concatenar palavras e letras, entre outras funções, mas isso será trabalhado futuramente, por enquanto o mais importante é saber que o comando print() vai apresentar mensagens na tela.

## Comentários # e ‘’’ ‘’’

É interessante que um código tenha comentários para explicar do que se trata determinada variável ou determinado algoritmo por exemplo. Para isso, pode-se utilizar o # (código) ou ‘’’ (código) ‘’’, quando esses comandos são utilizados o código é ignorada e, assim, passa a ser um comentário, observe o exemplo:

Figura 2 - Exemplo de Comentários



Nos dois casos o comando print não funciona, pois estão no modo de comentário. Posteriormente, alguns códigos serão passados e será possível observar com mais clareza a importância dos comentários.

# Variáveis

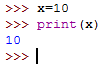
## Conceitos Exemplos e Observações

Variável é um dos conceitos mais importante de programação e é constantemente utilizado por qualquer programador em qualquer linguagem. Dessa forma, é importante que a definição e a forma como utilizar as variáveis fique bem clara para que o próximo módulo possa ser iniciado.

Uma interessante analogia para explicar o que são as variáveis é pensar que elas são como caixas, que servem para armazenar informações. Dessa forma, caso exista um número, uma letra, ou uma palavra que se deseja utilizar mais de uma vez, ou utilizar mais tarde, pode-se usar as variáveis.

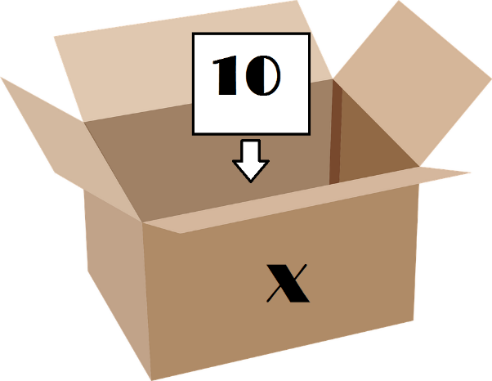
Inicialmente cria-se a variável e então a informação é armazenada em seu interior. Observe o exemplo abaixo:

Figura 3 - Exemplo Simples do Uso de uma Variável



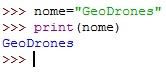
Primeiro uma variável chamada “x” é criada e em seguida o número 10 é colocado em seu interior. Observe que na programação o sinal “=” não é como na matemática, o sinal de igual significa que, neste caso, 10 “está dentro” de “x”, ou seja esse sinal armazena o valor dentro da variável. Seguindo com o exemplo, tem-se neste caso uma caixa chamada “x” em que seu interior tem o número 10:

Figura 4 - Analogia Caixa e Variável



Pode-se armazenar também números decimais, símbolos, letras, palavras e até mesmo frases em uma variável, como nota-se com o código a seguir:

Figura 5 - Exemplo Simples do Uso de uma Variável (2)



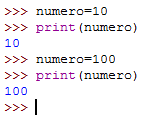
Primeiro uma variável chamada “nome” é criada e em seguida a palavra GeoDrones é colocada em seu interior.

Na programação existem diversos tipos de variáveis. É importante saber esses tipos para que essas variáveis possam ser trabalhadas. Será utilizado, por enquanto, três tipos principais:

* INT: Números inteiros
* FLOAT: Números decimais
* STR ou String: Palavras, símbolos, letras, etc. Geralmente é identificada por estar entre “”.

Deve-se ter em mente que quando um valor é armazenado em uma variável, esse valor permanece até que outro valor seja armazenado em seu local, substituindo-o, como mostra o exemplo a seguir:

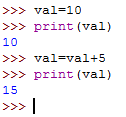
Figura 6 - Exemplo de Como um Valor se Sobreescreve em Uma Variável



Nesse caso o valor 10 dentro de “numero” é substituído por 100.

Além disso, é possível realizar operações da variável com ela mesma, observe o exemplo:

Figura 7 – Exemplo de Como Realizar Operações de Uma Variável com Ela Mesma



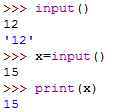
Aqui, a variável “val” é a soma dela mesma com o valor 5, obtém-se então 15 (10+5).

Uma boa prática de programação é sempre lembrar que o nome da variável é muito importante, dessa forma procure colocar nomes sugestivos para suas variáveis, assim você vai saber o que elas representam futuramente.

## Função input()

Ao digitar o comando *input()* no terminal IDLE do Python, o programa espera o usuário digitar algum texto no teclado e então apertar *enter*. Pode-se armazenar esse valor em uma variável, assim o usuário pode interagir com o programa desenvolvido, possibilitando uma programação dinâmica. Observe o exemplo:

Figura 8 - Exemplo da Função input()

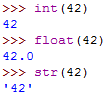


Após acionar o comando input(), o número 12 é digitado e então expressado. Em seguida, o comando é armazenado dentro da variável *x*, e quando essa é impressa na tela tem-se o valor que o usuário digitou.

## Função int(), float() e str()

Para fazer a soma de dois valores inteiros, por exemplo, é necessário que ambos sejam do tipo int, então se for somado um valor do tipo int e um valor do tipo str um erro será mostrado na tela. Portanto, para que esse erro não ocorra utiliza-se funções de conversão, que nesse caso são as funções *int()*, *float()* e *str()*. Observe os exemplos:

Figura 9 - Exemplo das Funções int(), float() e str()



Observe que o número 42 foi convertido para inteiro, decimal e para *string*. Essa conversão é feita para o valor 42, pois ele está entre parênteses e está sendo enviado como entrada na função. No capítulo 6, o significado de “entrada” será descrito com mais detalhes.

Lembre-se que não se deve realizar operações com variáveis de tipos diferentes e por causa disso essa função é muito importante e deve ser usada sempre que necessário. Esteja atento com os exemplos para ver as melhores formas de utilizar essa função.

## Exercícios Práticos

1. Criar três variáveis, multiplicar todas e mostrar o resultado.
2. Criar quatro variáveis, correspondente à idade de quatro pessoas diferentes. Somar as idades e dividir por quatro, calculando assim a média.
3. Escrever três palavras e concatená-las.
4. Fazer os três exercícios acima porém utilizando *input()* para a entrada de dados.
5. Fazer um algoritmo que calcule a área de um triângulo, quadrado e círculo. Procure, antes de usar o comando de entrada de dados *input()*, informar o usuário qual dado que ele está solicitando e o que será solicitado.
6. Desenvolva um programa que pergunta o nome da pessoa e a data. Depois esse programa exibe a seguinte mensagem: “Olá NOME\_DA\_PESSOA hoje é dia DATA\_DIGITADA”.
7. Desenvolva um programa que armazena o nome e o peso de três pessoas diferentes. Feito isso o programa diz o nome das três pessoas e o peso de cada uma. (peso deve ser do tipo float).
8. Desenvolva um programa que armazena o nome e o peso de três pessoas diferentes. Feito isso o programa diz o nome das três pessoas e a média do peso de cada uma. (peso deve ser do tipo float).
9. Faça um programa que solicita um número. Ao entrar com esse número o programa retorna o valor anterior a este número, o posterior, o dobro dele, o triplo e sua raiz quadrada.
10. Faça um programa que solicita um número. Ao entrar com esse número o programa retorna o valor ao quadrado desse número.
11. (Pergunta Desafio): Faça um programa que realiza conversões numéricas de quilômetro para metro, ou seja, ao entrar com um número em quilometro, o programa deve retornar esse mesmo número em metro.
12. (Pergunta Desafio): Faça um programa que realiza conversões numéricas de real para dolar, ou seja, ao entrar com um número em reais, o programa deve retornar esse mesmo número em dolares.

## Resolução dos Exercícios Práticos

############# EXERCÍCIOS PRÁTICOS – VARIÁVEIS #############

#Exercício 1)

x=1

y=2

z=3

soma=x+y+z

print(soma)

#Exercício 2)

id\_gu=27

id\_nat=22

id\_marcos=26

id\_marcel=23

media=(id\_gu+id\_nat+id\_marcos+id\_marcel)/4

print(media)

#Exercício 3)

palavra1="Ola "

palavra2="Geo"

palavra3="Drones"

conca=palavra1+palavra2+palavra3

print(conca)

#Exercício 4)

x=input()

x=int(x)

y=input()

y=int(y)

z=input()

z=int(z)

soma=x+y+z

print(soma)

id\_gu=input()

id\_gu=int(id\_gu)

id\_nat=input()

id\_nat=int(id\_nat)

id\_marcos=input()

id\_marcos=int(id\_marcos)

id\_marcel=input()

id\_marcel=int(id\_marcel)

media=(id\_gu+id\_nat+id\_marcos+id\_marcel)/4

print(media)

palavra1=input()

palavra2=input()

palavra3=input()

conca=palavra1+palavra2+palavra3

print(conca)

#Exercício 5)

print("Por favor, entre com valor da base do triângulo")

b=int(input())

print("Por favor, entre com valor da altura do triângulo")

h=int(input())

area=b\*h/2

print("A área do triângulo é:",area)

print("Por favor, entre com valor do lado do quadrado")

l=int(input())

area=l\*l

print("A área do quadrado é:",area)

print("Por favor, entre com valor do raio do círculo")

r=int(input())

area=3.14\*r

print("A área do círculo é:",area)

# Condicionais

## Conceitos, Exemplos e Observações

Até o momento foi estudado a programação em sequência, ou seja, o código simplesmente é executado lendo linha por linha. Porém, o mais incrível da programação é a sua capacidade de pular instruções, repeti-las ou até mesmo escolher qual código vai rodar. Para isso, usa-se as condicionais. Com isto, é possível controlar o fluxo do código, ou seja, permitir ou proibir que algo seja executado. Observe a figura a seguir:

Figura 10 - Fluxograma Exemplo de Condicionais

Uma imagem contendo texto, mapa

Descrição gerada com alta confiança

Nessa figura tem-se que ao iniciar o código uma pergunta é feita, caso a resposta seja sim um comando vai ser executado e então o programa é encerrado, caso a resposta seja não outro comando é executado, fazendo outra pergunta, e assim por diante, até que finalmente o programa é encerrado. Note que nesse tipo de programa existe mais de uma maneira do programa começar e terminar, e são as condicionais que verificam se a resposta é sim ou não, que vão controlar qual rumo o código vai tomar, possibilitando o controle do fluxo do programa.

Para utilizar as condicionais o código é simples: basta escrever “if”, a condição e depois colocar “:”, o que deve ser executado caso a condição é satisfeita deve ficar espaçado. Observe os seguintes códigos:

Figura 11 – Exemplo de Condicional 1

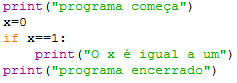


Figura 12 - Resposta do Exemplo de Condicional 1



Figura 13 - Exemplo de Condicional 2

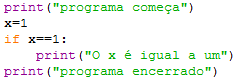


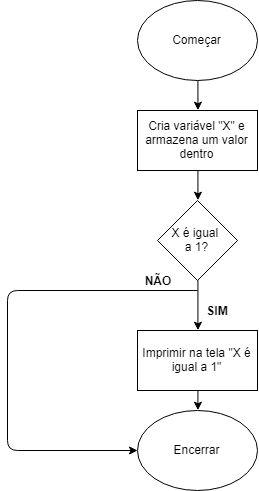
Figura 14 - Resposta do Exemplo de Condicional 2



Primeiro é mostrado na tela a frase “programa começa”, em seguida uma variável x=1 é inicializada. Depois vem a condicional que verifica se x é igual a 1, caso positivo imprime na tela “O x é igual a um”, caso negativo nada é mostrado. Encerrando o código com a frase “programa encerrado”. Observe que o sinal “==” é diferente do sinal “=”, no caso do “==” tem-se que realmente aqui o símbolo é como o da matemática, ou seja, ele verifica se algo é igual a outro.

Para o segundo caso, é mostrado na tela a frase “programa começa”, em seguida uma variável x=0 é inicializada. Depois vem a condicional que verifica se x é igual a 1, caso positivo imprime na tela “O x é igual a um”, caso negativo nada é mostrado. Encerrando o código com a frase “programa encerrado”. No caso do exemplo 2, o x é igual a 1, então é apresentado a frase “o x é igual a um”, no caso do exemplo 1 o x é igual a zero, então apenas duas frases são apresentadas. O fluxograma a seguir exemplifica melhor o que está acontecendo:

Figura 15 – Fluxograma Exemplo de Condicionais 2



## Operadores de Comparação

Na sessão anterior se observa a comparação de igualdade entre uma variável e um número, porém a comparação pode ser realizada também por outros caracteres, letras, palavras, entre outros. Além disso, pode-se comparar dois itens de 6 formas diferentes, a tabela abaixo mostra os operadores de comparação, onde é possível verificar se dois valores são iguais, diferentes, maiores, menores, maior ou igual e menor ou igual:

Tabela 1 - Operadores de Comparação

|  |  |
| --- | --- |
| **Operador** | **Significado** |
| == | Igual |
| != | Diferente |
| < | Maior |
| > | Menor |
| <= | Maior ou Igual |
| >= | Menor ou Igual |

O exemplo a seguir é o mesmo da figura 10, porém a condição agora é se x é diferente de 1:

Figura 16 - Exemplo de Condicional 3

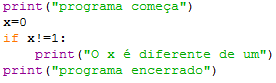


Figura 17 - Resposta do Exemplo de Condicional 3



## E se não

Existe também a possibilidade de se fazer uma ação caso a condição não seja satisfeita, isto é, se a condição atender faça algo, se não faça outro. Para isso, basta escrever “else:” e dizer o que deve ser feito, observe o exemplo:

Figura 18 - Exemplo de Condicional 4

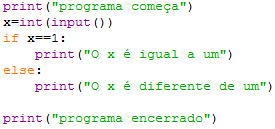


Figura 19 - Resposta 1 do Exemplo de Condicional 4

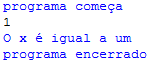
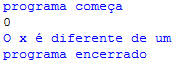


Figura 20 - Resposta 2 do Exemplo de Condicional 4

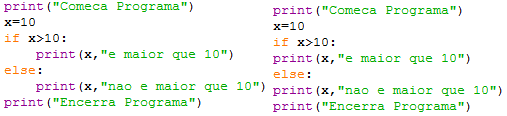


Neste caso, se o x for igual a um ele imprime “o x é igual a um”, se não ele imprime “o x é diferente de um”. Existe também o comando “elif”, que não será abortado por enquanto, mas sinta-se livre para pesquisar.

## Indentação

Para qualquer programa que for escrito, principalmente na linguagem Python, deve-se ficar atento quanto a indentação. Isso é a estrutura que o código tem para que ele seja mais legível e, em Python, um código deve estar sempre indentado, isto é, com os espaçamentos corretos e alinhados para que tudo funcione normalmente. Veja abaixo a forma indentada e correta de se escrever um código e a forma não indentada e incorreta de se escrever:

Figura 21 - Exemplo do Uso de Código Indentado



Os códigos sempre devem ser escritos como no primeiro caso, no segundo caso algumas linguagens até podem funcionar, porém não é a forma mais legível e correta de se desenvolver um programa.

## Exercícios Práticos

1. Faça um programa que cria uma variável, verifica se ela é igual a 100, caso afirmativo ele deve dizer: “variável igual a 100”.
2. Faça um programa que pede para a pessoa dizer um número, o usuário então entra com um número e o programa verifica se o número é menor que 100, se sim ele imprime uma mensagem dizendo “seu número é menor que 100”.
3. Escreva um código em que o usuário vai escrever um nome, se o nome for geodrones ele deve falar “o nome escrito é geodrones”.
4. Escreva um código em que o usuário vai escrever um nome, se o nome for geodrones ele deve falar “o nome escrito é geodrones”, se não ele deve mostrar “o nome escrito não é geodrones”.
5. Escreva um programa que avisa se a pessoa é maior de idade ou não.
6. Escreva um código usando os seis operadores de comparação.
7. (Pergunta Desafio): Escreva um código que pergunta qual área a pessoa deseja calcular: triângulo, quadrado ou círculo. Dependendo da resposta o programa calcula a área correspondente.
8. Crie um código que permite que dois jogadores joguem par ou ímpar. Primeiro o jogador 1 escolhe par ou ímpar, depois escolhe o seu número. Em seguida, o jogador 2 entra com número. Finalmente o computador mostra quem venceu o jogo, se foi o jogador 1 ou o jogador 2.
9. Crie um código que advinha o número. Primeiro o usuário deve entrar com um número entre 0 e 9. Em seguida o computador deve dizer qual número foi digitado sem escrever na tela no número salvo, ou seja, utilizando condicionais.
10. Crie um programa que pede o nome e idade do usuário. Em seguida ele fala qual classificação de filmes ele não pode assistir. Exemplo: se ele tem 8 anos ele não pode assistir filmes que são proibidos para menores de 10 anos, se tem 12 ele não pode assistir filmes que são proibidos para menores de 14 anos, se tem 15 ele não pode assistir filmes que são proibidos para menores de 18 anos, se tem 20 anos, ele pode assistir qualquer filme.

## Resolução dos Exercícios Práticos

############# EXERCÍCIOS PRÁTICOS - CONDICIONAIS #############

#Exercício 1)

x=100

if x==100:

print("variável igual a 100")

#Exercício 2)

print("Diga um número:")

x=int(input())

if x<100:

print("seu número é menor que 100")

#Exercício 3)

print("Escreva um nome:")

nome=input()

if nome=="geodrones":

print("você escreveu geodrones")

#Exercício 4)

print("Escreva um nome:")

nome=input()

if nome=="geodrones":

print("o nome escrito é geodrones")

else:

print("o nome escrito não é geodrones")

#Exercício 5)

print("Escreva sua idade:")

idade = int(input())

if idade>=18:

print("Você é maior de idade")

else:

print("Você não é maior de idade")

#Exercício 6)

print("Escreva um número:")

numero = int(input())

if numero==10:

print("Seu número é igual a 10")

if numero!=10:

print("Seu número é diferente de 10")

if numero>10:

print("Seu número é maior que 10")

if numero<10:

print("Seu número é menor que 10")

if numero>=10:

print("Seu número é maior ou igual 10")

if numero<=10:

print("Seu número é menor ou igual a 10")

#Exercício 7)

print("O que deseja calcular? triângulo, quadrado ou círculo?")

desejo = input()

if desejo == "triângulo":

print("Por favor, entre com valor da base do triângulo")

b=int(input())

print("Por favor, entre com valor da altura do triângulo")

h=int(input())

area=b\*h/2

print("A área do triângulo é:",area)

if desejo == "quadrado":

print("Por favor, entre com valor do lado do quadrado")

l=int(input())

area=l\*l

print("A área do quadrado é:",area)

if desejo == "círculo":

print("Por favor, entre com valor do raio do círculo")

r=int(input())

area=3.14\*r

print("A área do círculo é:",area)

# Booleanos

## Conceitos, Exemplos e Observações

Antes de prosseguir com novos conceitos de programação, é muito importante que se saiba o que são os valores booleanos. Diferentemente do tipo *int*, *float* e *string*, que tem uma gama de possíveis valores, os valores booleanos possuem apenas dois: *true* e *false*, ou verdadeiro e falso. Em Python, quando uma comparação é realizada, uma mensagem é enviada dizendo se a comparação foi verdadeira ou falsa, e por isso é importante entender como funciona. Observe o exemplo abaixo:

Figura 22 - Exemplo de Valor Booleano True



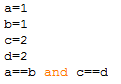
No código acima nota-se que x e y são iguais, logo ao executar o comando “x==y” automaticamente tem-se um retorno True, se eles não fossem iguais seria mostrado False, como mostrado a seguir:

Figura 23 - Exemplo de Valor Booleano False



Além disso, é de fundamental importância saber os operadores booleanos ou operadores lógicos. Simplificando, existem três: *and*, *or* e *not* (*e*, *ou* e *não*) e eles servem para fazer comparações de forma análoga aos operadores de comparação. Analise o exemplo:

Figura 24 - Exemplo de Operador Booleano AND





Uma vez que “a” é igual a “b” E “c” é igual a “d” tem-se uma resposta positiva, ou seja, tem-se um valor verdadeiro. Porém, se for feita a substituição “a=0” a resposta é False, pois apenas “c” é igual a “d”. Assim, pode-se dizer que:

* And – Retorna um valor verdadeiro apenas se receber duas expressões verdadeiras;
* Or – Retorna um valor verdadeiro se uma das possibilidades for verdadeira;
* Not – Retorna um valor verdadeiro se receber uma expressão falsa e retorna falso se receber verdadeira.

Veja a tabela de operadores lógico:

Tabela 2 - Tabela de Operadores Lógicos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Comparação A** | **Comparação B** | **A and B** | **A or B** | **not A** |
| Verdadeiro | Verdadeiro | True | True | False |
| Verdadeiro | Falso | False | True | False |
| Falso | Falso | Falso | Falso | True |
| Falso | Verdadeiro | False | True | True |

## Números Binários

Aproveitando o tópico em questão, é interessante levantar uma observação sobre os números binários, que são basicamente um sistema de numeração em que qualquer número pode ser representado pelos números 1s e 0s. Os computadores geralmente trabalham com dois níveis de tensão, é por isso adotam os sistemas binários em suas configurações. Não vamos entrar em detalhes sobre esse assunto, porém cabe mostrar a tabela de conversão dos números binários:

Tabela 3 - Tabela de Conversão Decimal para Binário

|  |  |  |  |
| --- | --- | --- | --- |
| **Decimal** | **Binário** | **Decimal** | **Binário** |
| 1 | 0001 | 5 | 0101 |
| 2 | 0010 | 6 | 0110 |
| 3 | 0011 | 7 | 0111 |
| 4 | 0100 | 8 | 1000 |

Se esta tabela for analisada cuidadosamente é possível observar um padrão de conversão. Porém, estudar esses números não é o foco do curso e por isso essa conversão ficará apenas a título de curiosidade.

## Exercícios Práticos

1. Faça um código que compara se quatro pessoas tem a mesma idade, caso afirmativo mostrar a mensagem na tela de positivo ou negativo (use os operadores booleanos).
2. Faça um código que compara se pelo menos duas pessoas tem a mesma idade, caso afirmativo mostrar a mensagem na tela de positivo ou negativo (use os operadores booleanos).
3. Faça um programa que compara o nome de quatro pessoas, e ele vai dizer se pelo menos duas pessoas tem o mesmo nome (use os operadores booleanos).
4. Desenvolva um código que verifica senhas, ou seja, se a pessoa digitar uma senha e estiver correto, dizer que a passagem foi permitida, se não dizer que a passagem foi negada.
5. Desenvolva um código que verifica senhas como no exercício anterior, porém dessa vez faça que mais de uma senha seja permitida.
6. Escreva um programa que armazena 10 números e imprime os que são maiores que 10 e os menores que 3.

## Resolução dos Exercícios Práticos

############# EXERCÍCIOS PRÁTICOS - BOOLEANOS #############

#Exercício 1)

print("Entre com quatro idades quaisquer:")

idade1=int(input())

idade2=int(input())

idade3=int(input())

idade4=int(input())

if idade1==idade2 and idade3==idade4:

print("Positivo")

else:

print("Negativo")

#Exercício 2)

print("Entre com quatro idades quaisquer:")

idade1=int(input())

idade2=int(input())

idade3=int(input())

idade4=int(input())

if idade1==idade2 or idade3==idade4:

print("Positivo")

else:

print("Negativo")

#Exercício 3)

print("Entre com quatro nomes quaisquer")

nome1=input()

nome2=input()

nome3=input()

nome4=input()

if nome1==nome2 or nome3==nome4:

print("Positivo")

else:

print("Negativo")

#Exercício 4)

print("Entre com a senha:")

senha=int(input())

if senha==1234:

print("Acesso permitido")

else:

print("Acesso negado")

#Exercício 5)

print("Entre com a senha:")

senha=int(input())

if senha==1234 or senha==4321:

print("Acesso permitido")

else:

print("Acesso negado")

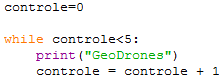
# Laços de Repetição

## Conceitos, Exemplos e Observações

### While

Como já mencionado, o mais interessante da programação é a capacidade de se controlar o fluxo dos códigos. Neste capítulo será trabalhado os laços de repetição ou *loops*. Como o próprio nome já diz, os laços de repetição permitem que uma determinada parte do código seja executado quantas vezes for necessário. Existem inúmeras formas para fazer isso, mas inicialmente vamos trabalhar com o *while*. Este vai verificar uma condição e se ela for verdadeira o código se repete, assim enquanto a condição for verdadeira ele se repete. Analise o exemplo a seguir:

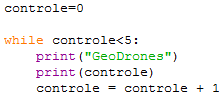
Figura 25 - Exemplo 1 de Laço de Repetição While

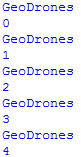




Observe que a palavra “GeoDrones” repetiu 5 vezes, porém o comando print() foi escrito apenas uma vez. Isso ocorre devido ao laço de repetição “while”, que repete a mesma ação indentada enquanto a condição é satisfeita. Observe agora o mesmo código, porém a variável “controle” será mostrada na tela juntamente com a palavra GeoDrones:

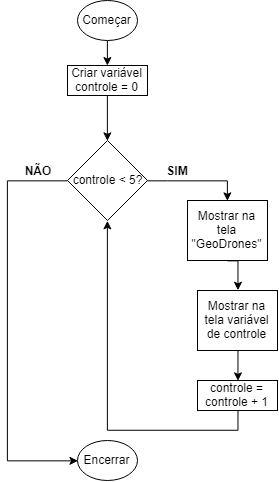
Figura 26 - Exemplo 2 de Laço de Repetição While Imprimindo a Variável de Controle





O fluxograma abaixo mostra o que está acontecendo:

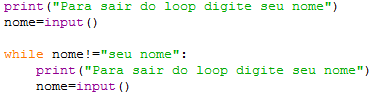
Figura 27 - Fluxograma para Exemplificar Laço de Repetição While



Ao inicializar o código, cria-se uma variável chamada controle e armazena em seu interior o valor zero. Depois a condição *controle<5* é verificada, como é uma condição verdadeira entra-se no laço de repetição e mostra a palavra “GeoDrones” e depois o valor da variável controle. Soma-se 1 no valor de controle e então a condição é verificada novamente, como a variável controle agora é igual a 1, tem-se uma condição verdadeira, logo entra-se no laço de repetição e mostra a palavra “GeoDrones” e depois o valor da variável controle. Soma-se 1 no valor de controle e então a condição é verificada mais uma vez. Agora o valor é igual a 2, tem-se uma condição verdadeira novamente, então o processo se repete. Entretanto, quando o valor de controle é igual a 5 o código para de se repetir, pois a condição é satisfeita, assim o programa é encerrado.

Observe outro exemplo utilizando o comando “while”:

Figura 28 - Exemplo Laço While de Dizer o Nome



Analisando este código é possível dizer o que vai acontecer? Veja que enquanto a pessoa não digitar “seu nome” o código vai se repetir para sempre, pois só quando a variável nome for igual a “seu nome” o laço vai parar de ser executado.

É importante ressaltar que deve-se ter cuidado para não criar laços de repetição infinita, isto é, laços em que a condição nunca será insatisfeita, pois quando um laço desse roda ele não para mais.

### For

Existe também outra maneira, mais prática, de se criar laços de repetição. Para isso, basta usar o comando “for”. O laço de repetição “while” verifica a condição constantemente, enquanto o “for” roda o código um determinado número de vezes através da função range(). Analise o exemplo para uma melhor compreensão:

Figura 29 - Exemplo Laço For





Note que o código funcionou por 5 vezes. A função “for” executou os seguintes passos: Primeiro foi criada uma variável “i” com valor zero, a função print(), então, é chamada, depois o código volta para o topo e a função “for” incrementa 1 na variável “i”. O código se repete até que a variável “i” não se inclua mais dentro de “range(5)”, neste caso o programa encerra.

É possível escrever um código equivalente utilizando o laço “while”, veja:

Figura 30 - Exemplo Equivalente While/For





É possível também começar, interromper e pular argumentos utilizando a função range():

Figura 31 – Exemplo Laço For com Intervalo Diferente





Nesse caso, a função é executada apenas entre o intervalo de X até Y, pois foi o intervalo delimitado pela função range().

Figura 32 - Exemplo Laço For Incrementando 2





Aqui, após delimitar os espaços da função, colocamos quanto vamos incrementar na variável, neste caso 2, logo ela vai incrementar 2 em “i” ao invés de 1.

Figura 33 - Exemplo Laço For Incrementando





Também pode-se decrementar ao invés de incrementar, basta colocar o intervalo e quanto se deseja decrementar, como mostrado acima.

## Exercícios Práticos

1. Desenvolva um algoritmo que peça para a pessoa digitar o número zero, e o programa vai pedir para isso ser digitado repetidamente, até que zero seja realmente digitado.
2. Escreva um código que mostra na tela os números de 0 até 500 (use o laço “While”).
3. Faça um código que mostra na tela os números de 0 até 500, porém apresente apenas os números pares.
4. Programe um código que mostra na tela os números de 500 até 0, sempre decrementando 1.
5. Programe um código que mostra na tela os números de 500 até 0, sempre decrementando 3.
6. Repita os programas dos itens de 2 a 5, porém dessa vez use o laço for.
7. Faça um programa que imprima a tabuada de 3.
8. Faça um programa que imprima a tabuada de 5 (use o laço while).
9. Faça um programa que imprima a seguinte sequência utilizando os laços de repetição for e while: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048.
10. Faça um programa que calcule a sequência Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21 e 34.
11. Crie um laço de repetição infinito, ou seja, que nunca será interrompido.

## Resolução dos Exercícios Práticos

############# EXERCÍCIOS PRÁTICOS - LAÇOS DE REPETIÇÃO - WHILE #############

#Exercício 1)

print("Digite o numero zero")

zero=input()

zero=int(zero)

while zero!=0:

print("Digite o numero zero")

zero=input()

zero=int(zero)

#Exercício 2)

numero=0

while numero<501:

print(numero)

numero=numero+1

#Exercício 3)

numero=0

while numero<501:

print(numero)

numero=numero+2

#Exercício 4)

numero=500

while numero>0:

print(numero)

numero=numero-1

#Exercício 5)

numero=500

while numero>0:

print(numero)

numero=numero-3

############# EXERCÍCIOS PRÁTICOS - LAÇOS DE REPETIÇÃO - FOR #############

#Exercício 1)

print("Digite o numero zero")

zero=input()

zero=int(zero)

while zero!=0:

print("Digite o numero zero")

zero=input()

zero=int(zero)

#Exercício 2)

for numero in range(501):

print(numero)

#Exercício 3)

for numero in range(0,501,2):

print(numero)

#Exercício 4)

for numero in range(500,-1,-1):

print(numero)

#Exercício 5)

for numero in range(500,-1,-3):

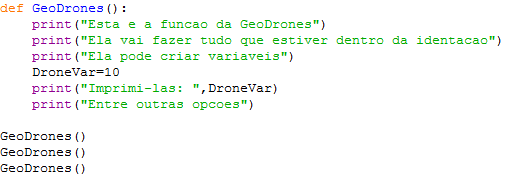
print(numero)

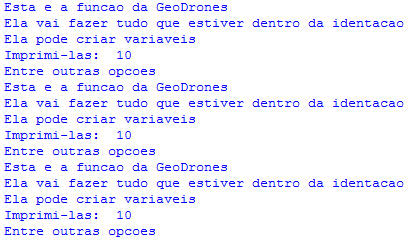
# Funções

## Conceitos, Exemplos e Observações

Em Python, como já mencionado antes, existem várias funções prontas, como print() e input(), porém também é possível criar funções. Este basicamente são “miniprogramas com programas internos”, para um melhor entendimento veja os exemplos abaixo:

Figura 34 - Exemplo 1 do Uso de Funções



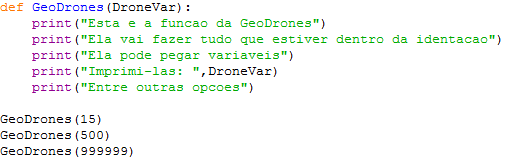


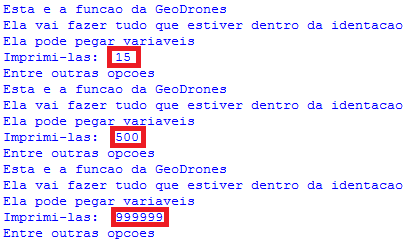
Na primeira linha deve-se escrever “def” e então definir o nome da funão, que neste caso é GeoDrones. O código que segue com a indentação é o corpo da função, e este código é executado quando a função é chamada. É muito importante observar que esse código só é executado quando a função é chamada, ou seja, nas sete primeiras linhas o código não é executado.

Após definido o que será a função, nota-se que ela é chamada 3 vezes, e por isso o código roda três vezes. O uso das funções começa a ficar claro neste ponto, pois se algum código mais complexo precisar ser utilizado mais de uma vez, ao invés de escrevê-lo várias e várias vezes, pode-se utilizar as funções, em que o código é desenvolvido apenas uma vez e se repete quantas vezes for necessário. Note que o uso das funções pode tornar seu código mais fácil, mais organizado e mais sucinto.

Todas as funções utilizadas até o momento terminam com o uso de parênteses “()”. Esses parênteses servem para indicar se a função tem alguma entrada. Analise o exemplo abaixo:

Figura 35 - Exemplo de Função com Entrada

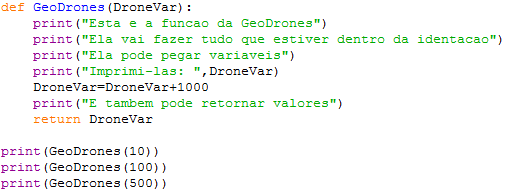


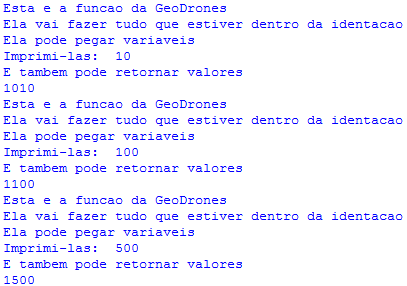


Note que as funções foram chamadas e um valor foi inserido dentro dos parênteses. Esses valores foram para dentro da variável “DroneVar”, e foram todas impressas. Assim, é possível alterar e controlar o que as funções irão executar. Note que a função int(), float() e str() utilizam valores de entrada para realizar a conversão.

Um último conceito que se deve aprender sobre funções são os retornos, ou “return”. A função depois de executada ela pode retornar algo, ou seja, o código usado pode se transformar naquilo que é retornado, analise o código abaixo:

Figura 36 - Exemplo de Função com Entrada e Saída





Note que agora a função GeoDrones está sendo executada e além disso nota-se uma entrada para print(), ou seja, o valor da função GeoDrones está sendo apresentado na tela. E esse valor é a variável “DroneVar”. Essa variável foi alterada no código e por isso está diferente do que foi enviado. Em outras palavras, a função GeoDrones se torna o que ela retorna em seu corpo.

## Exercícios

1. Crie uma função que diga os números de 1 a 10 (use laço de repetição). Depois chame essa função.
2. Crie uma função que recebe um número par e transforma esse número em um número ímpar.
3. Crie uma função que calcula a área de um quadrado.
4. Crie uma função que recebe um número e faz uma contagem regressiva dele até 0.
5. Desenvolva um código que pergunta por um número. Então esse número é enviado para uma função que dobra ele, então o valor é retornado e impresso na tela.
6. Crie duas funções uma que verifica se o número é maior que 100 e outra se é menor do que 100. Então, crie um laço de repetição que sempre pergunta um número e fala se esse número é maior ou menor que 100 chamando as funções criadas.
7. Crie três funções, uma que calcula área de triângulo, retângulo e círculo. Então o programa pergunta o que se deseja calcular, e de acordo com a resposta calcula-se o que se pede. Faça o programa perguntar o que se deseja calcular 3 vezes.
8. Crie uma função que calcula o consumo de um carro. Então pergunte quantos quilômetros o carro andou e depois pergunte quantos litros foi gasto, então a função é chamada e o valor do consumo é informado na tela.
9. Crie uma função que calcula a sequência Fibonacci. Então, pergunte quantos números da sequência se deseja e então retorna os número.

## Resolução dos Exercícios Práticos

############# EXERCÍCIOS PRÁTICOS - FUNCOES #############

#Exercício 1)

def diganumero():

for numero in range(10):

numero=numero+1

print(numero)

diganumero()

#Exercício 2)

def par2impar(par):

impar=par+1

print(impar)

print("diga um numero par:")

par=int(input())

par2impar(par)

#Exercício 3)

def quadrado(lado):

area=lado\*lado

print(area)

print("diga o tamanho do lado do quadrado:")

lado=int(input())

quadrado(lado)

#Exercício 4)

def diganumero(numG):

for numero in range(numG,-1,-1):

print(numero)

print("diga um numero:")

numG=int(input())

diganumero(numG)

#Exercício 5)

def dobra(num):

num=num\*2

return num

print("diga um numero:")

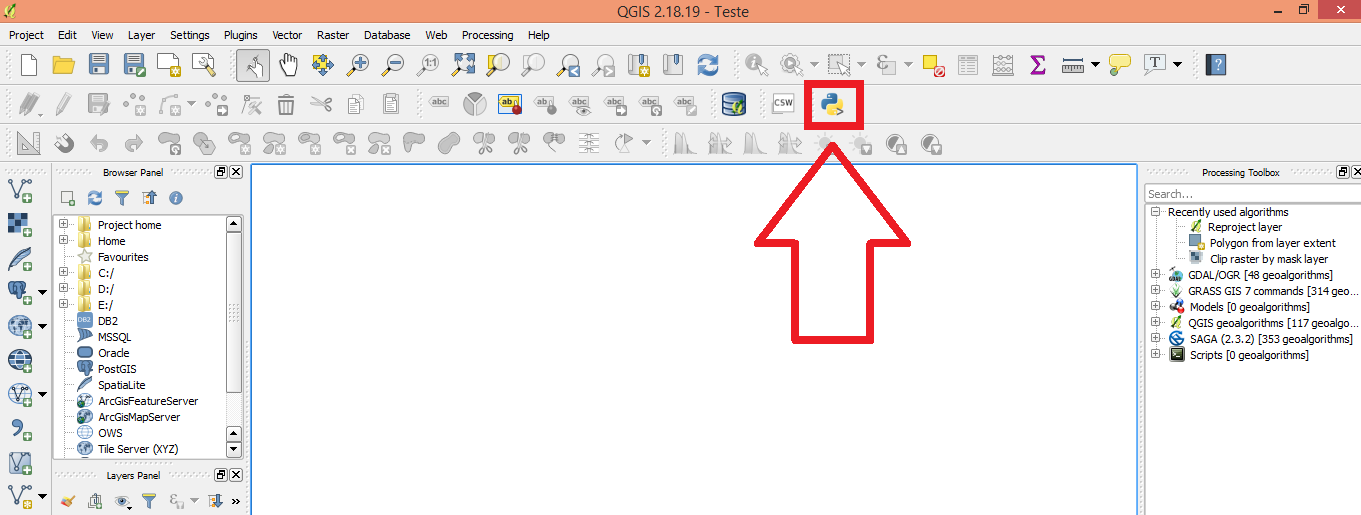
num=int(input())

print(dobra(num))

# Python e QGis

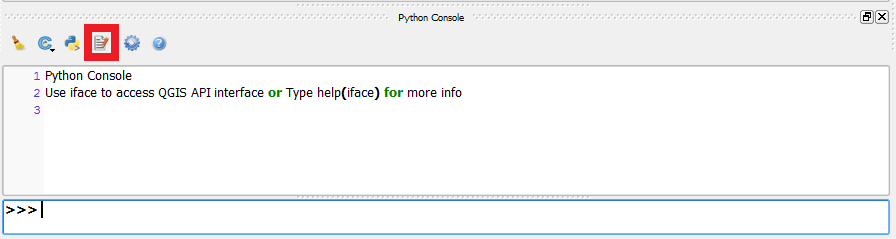
A plataforma QGis possui uma interface que permite que suas funcionalidades possam ser estendidas para scripts, isto é, é possível realizar e automatizar tarefas através do uso da linguagem Python no software QGis. Para acionar o *Console Python,* basta clicar no ícone python como mostra a figura abaixo:

Figura 37 - Iniciando Console Python na Plataforma QGis



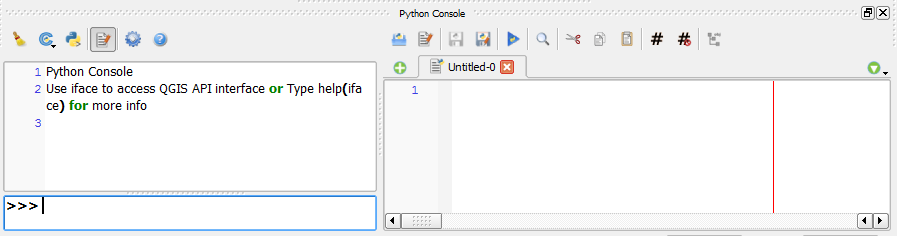
Com isso, um terminal será exibido onde comandos funções podem ser utilizadas. Na figura abaixo observa-se esse terminal:

Figura 38 - Console Python na Plataforma QGis



Clicando no ícone em destaque, observa-se que uma nova janela se abre, onde é possível escrever códigos maiores e mais complexos:

Figura 39 - Editor Python na Plataforma QGis



Para exemplificar como esse console funciona, observe o exemplo a seguir. Primeiro adicione uma *Layer* Vetorial no QGis utilizando o seguinte comando:

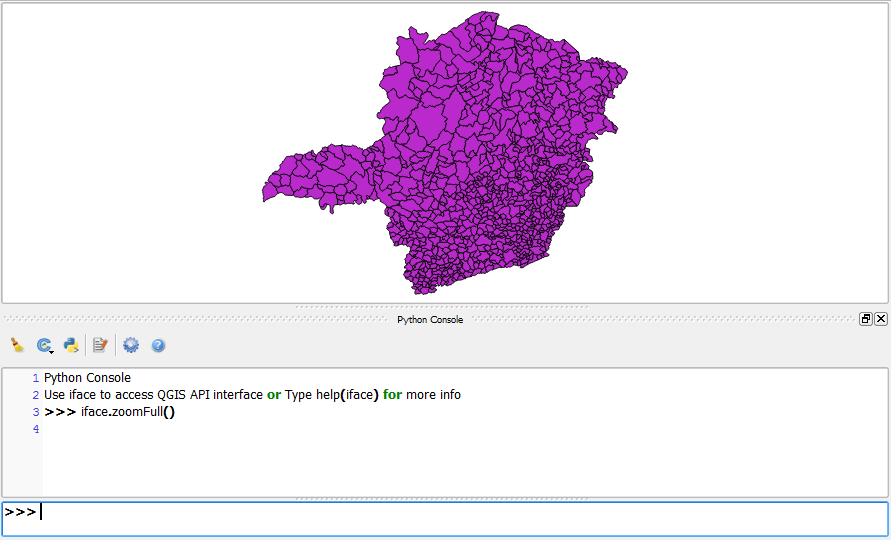
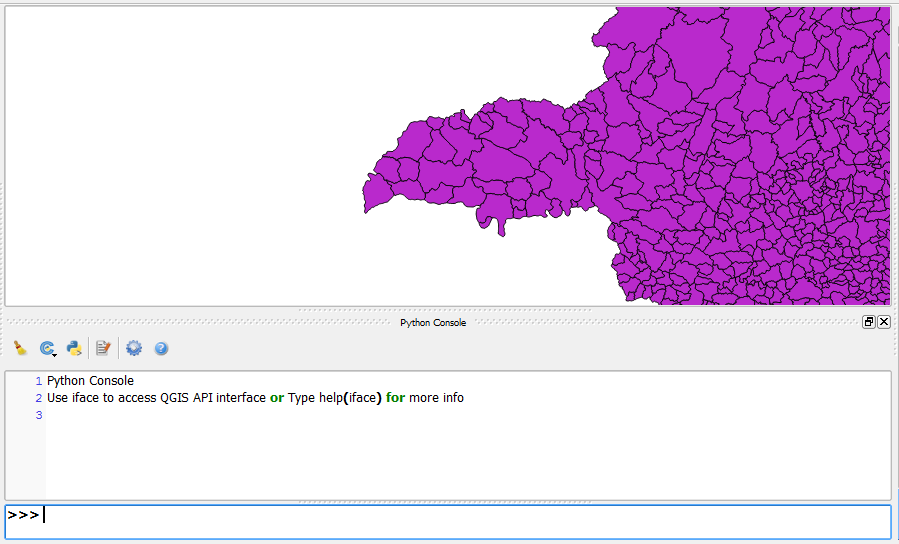
layer = iface.addVectorLayer("C:/CaminhaQualquer/Arquivo.shp", "NomeQualquer","ogr")  
if not layer:

print("Layer falhou ao carregar!")

Se colocar o caminho corretamente uma Layer Vetorial irá abrir automaticamente, se houver algo errado uma mensagem dizendo “Layer falhou ao carregar” será exibida.

Em seguida, escreva a seguinte função: *iface.zoomFull().* Essa função vai expandir sua Layer para até o tamanho da sua janela, observe a imagem abaixo antes e depois do uso dessa função:

Figura 40 - Resultado Após O Uso da Função *iface.zoomFull()*



Outras funções de zoom podem ser utilizadas através do console Python, tais como iface.zoomToActiveLayer(), que tem a tela ampliada na layer ativa; iface.zoomToPrevious() que amplia a próxima layer em questão; entre outras funções.

A plataforma QGis já possui muitas funções e é possível realizar muitas tarefas com elas, porém ao se utilizar Python é possível realizar qualquer tarefa e ir além das já propostas pelo software. O curso tem como objetivo passar os conceitos básicos de programação para que eles possam ser aplicados em softwares como o QGis, ArcGis, entre outros. Nas aulas pode-se encontrar mais exemplos utilizando Python na plataforma QGis, então não deixe de assisti-las. Mais informações sobre o uso de Python na plataforma QGis pode ser encontrada na página oficial indicada na seção Referências Bibliográficas.

# Fontes Bibliográficas

* Sweigart, A.: Automate the Boring Stuff with Python - Vol. 1. Editora No Starch Press.
* Borges, L. E.: Python para Desenvolvedores - Vol. 1, 2ª edição. Edição do Autor, 2010.
* Severance, C. R.: Python for Everybody Exploring Data Using Python 3 - Vol. 1, 2ª edição. Editora Elliott Hauser, Sue Blumenberg, 2016.
* PYTHON. 2018. Último acesso: 23 set. 2018. Disponível em:<https://www.python.org/>.
* W3SCHOOLS. 2018. Último acesso: 23 set. 2018. Disponível em:<https://www.w3schools.com/python/default.asp>.
* PyQGIS. 2018. Último acesso: 10 out. 2018. Disponível em: <https://docs.qgis.org/testing/en/docs/pyqgis\_developer\_cookbook/index.html>.

1. Para atualizar as páginas, basta clicar com o botão direito do mouse sobre o sumário e selecionar “Atualizar campo”. [↑](#footnote-ref-1)
2. Desde que os títulos das tabelas estejam utilizando o “Estilo Rápido” “Lista Tabelas”, basta clicar com o botão direito do *mouse* sobre a lista e selecionar “Atualizar campo” para atualizar essa lista. [↑](#footnote-ref-2)
3. Desde que os títulos das tabelas estejam utilizando o “Estilo Rápido” “Lista Figuras”, basta clicar com o botão direito do *mouse* sobre a lista e selecionar “Atualizar campo” para atualizar essa lista. [↑](#footnote-ref-3)