**PYTHON BACK END, do básico ao avançado com Python OO**

No Python, não é necessário usar ; no fim de cada linha de código

Nas variáveis não são determinados seus tipos, como int, string...

Print (“Olá Mundo”) <- impressão na tela

int, float // str

int= inteiro; float = número flutuante, com decimais; str = string de caracteres

para determinar uma variável constante é necessário que a variável esteja em caixa alta

ex:

preco = 10

print (preco)

preco = float(preco)

print (preco)

10.0 <- como nas variáveis são atribuídos tipos automaticamente pela linguagem, podemos modificar o seu type sempre que necessário, colocando seu novo tipo no valor

//caso utilize divisão em uma operação, é possível colocar duas barras para que o valor da operação já retorne diretamente como inteiro

**Número absoluto**, ignorando sinais de positivo e negativo

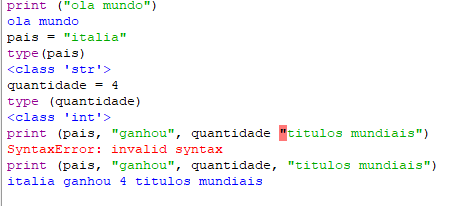
função = abs(variável ou número a ser reduzido)

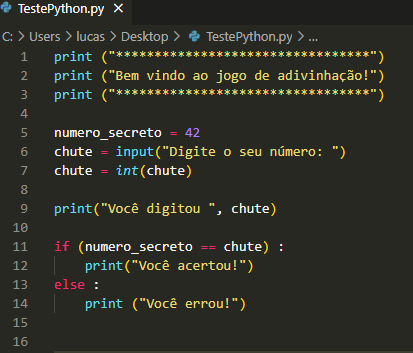
**Entrada de dados**

Nome = input(“Informe o seu nome:”) <- input

*Operadores= print (10%3) + (10\*\*2) <- módulo e exponenciação*

Print (“Brasil”, “ganhou 5 títulos mundiais”, sep= “-“) <- o código sep mostra o que será impresso entre os espaços de strings do print

País = “italia” <- criando variável de valor string Itália



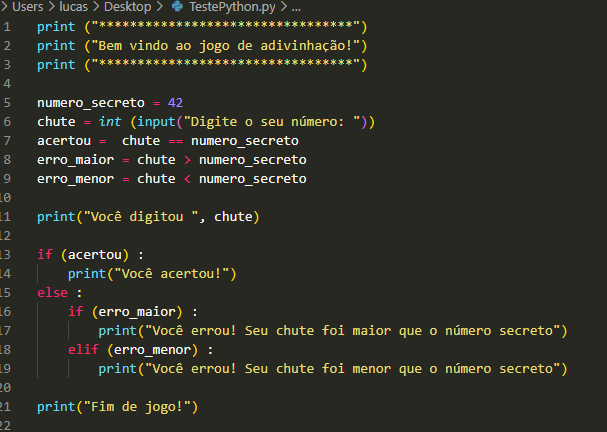
No Python é utilizado **:** para separar funções, *como as chaves {} em outras linguagens*,

Não é necessário fazer o fechamento, apenas usar novamente onde se quer abrir a chave;

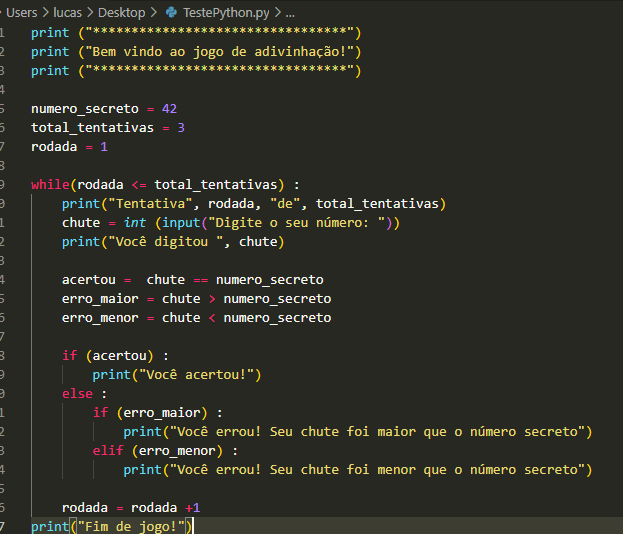
Também foi modificado o tipo da variável *chute* para **Int**, pois o python recebe tipo de valor padrão **String**, se não a operação de igualdade de variáveis retorna **false**

* chute = **int** (input(“Digite o seu número:”)) também é possível definir a variável ainda dentro do input

é interessante usar os operadores dentro das variáveis, para uma melhor visualização de código

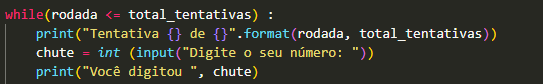


Para criar uma repetição, é usado a estrutura de Laço While

While = enquanto há tentativas/ ou enquanto a condição for true 

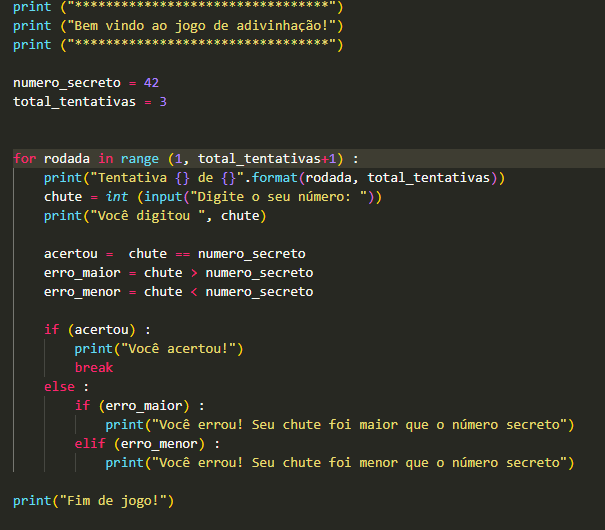
while (condição) :

**String Interpolation** Função Format, usa-se {} chaves para que sejam recebidas variáveis,

 **.**format(variáveis)

**For** *variável* **in range(1,10) :**

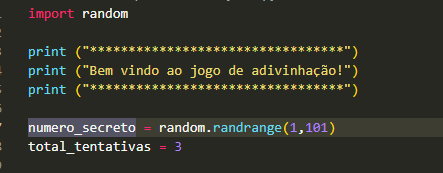
for é usado para laços de sequências, e range define qual é a sequência a ser seguida



A diferença entre o *for* e o *while*, é porque o *for* define uma sequência específica

E consegue lidar melhor com alguns casos de laços e repetições, nesse exemplo reduzimos a necessidade de algumas linhas de códigos com regras, a variável *rodada = 1* já não é mais necessária, pois o *for* se encarrega de definir o número que se inicia dentro do *range*, além da contagem de tentativas até o limite, que seria *total\_tentativas* (veja, o *+1* é usado nesse caso para que substitua um *<=*)

Para sair de um laço de repetição podemos “quebrar” a sequência e levar logo após usando **break**, também é possível usar **continue** para voltar ao início do laço sem prosseguir até o final

 Operadores **or** e **and** usam a mesma lógica de OU e E para definir regras

**Import de blibliotecas e módulos**

import random <- import de função

random.random() <- uso da função

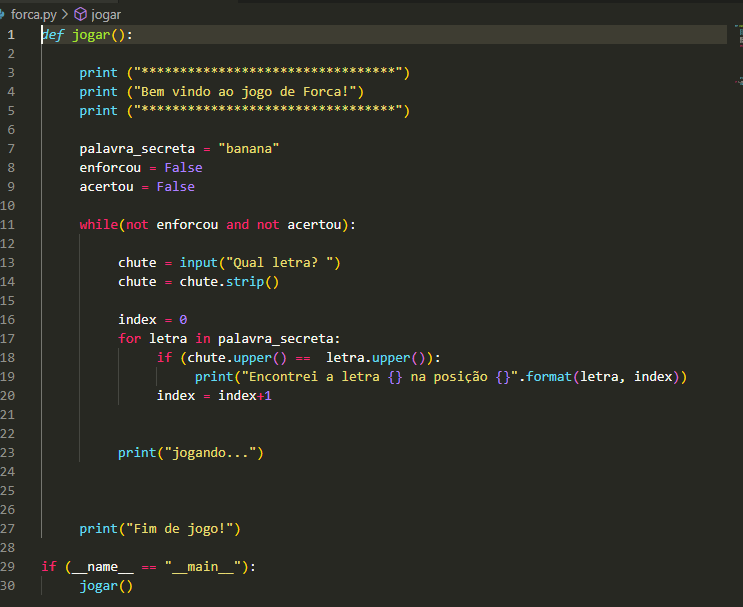
foi importado a biblioteca random, para utilização de números gerados aleatoriamente, a função é chamada através do seu nome de biblioteca .a função desejada; nesse caso o randrange gera números aleatórios em um intervalo definido entre ()

Definir Módulos para que seja possível fazer Import em outros arquivos de códigos,

É utilizado a função= **import** *NomeDoModulo/Pacote*no arquivo a ser usado, mas antes é necessário definir o nome do módulo no arquivo referenciado= **def *NomeDoModulo*():**

Para ser executado usando o próprio código, é possível com a função *NomeDoModulo()*

Função para encontrar letras dentro de palavras



Para fazer o método de procurar letras dentro da ***palavra\_secreta***, foi usado a função ***for*** para definir uma sequência de letras, ***if*** o ***chute == letra***, então vai ser mostrado na tela a letra e também a sua posição, que foi definida como variável de nome index, veja também que foi somado +1 a cada ***for*** para que o código verifique as posições na palavra.

Com essa lógica pode-se ter problemas de letras maiúsculas/minúsculas e espaços, então foi usado a função ***.uppe***r() nas variáveis para que sejam convertidos em caixa alta, e também a ***.strip()*** para que ignore os espaços e quebras de linhas; Note que é utilizado ao longo do código a função ***.append()*** para que seja adicionado algo entre () dentro da variável, e também algumas outras funções que irão ser explicadas ao longo do estudo.

***Sequências:*** *String , Range , List , Tuple*

Range[0,10] <- é impresso uma sequência de 0 até 9

List [] <- tipo de variável para que sejam armazenadas sequências

Tuple () <- tipo de variável para que sejam armazenadas sequências, mas que não se alteram

São grupos de valores, podem ser armazenados uma dentro da outra



É possível abrir documentos salvos no mesmo caminho que está o projeto Python, a partir da função *open(“nome do arquivo”, “qual a forma a ser tratada, r para leitura, w para escrita”)*

Foi então criado um arquivo .txt com palavras secretas para serem trabalhadas no jogo, a função *open* foi usada como leitura R e adicionada na variável de nome *arquivo*.

O laço ***For*** irá fazer uma sequência entre todas as ***linha in arquivo***, mesmo a variável linha não existindo até então, é possível utilizar para que retornem valores do arquivo em sequência.

No final, o ***For*** irá adicionar cada valor do arquivo dentro da lista ***palavras = []*** usando a função .append(); é importante fechar o arquivo sempre que terminar de fazer aberturas.

O ***len()*** consegue visualizar o tamanho do arquivo, nesse contexto seria qual a quantidade de valores há para que o ***random.randrange (0,\*)*** defina a sequência.

*Index += 1 é o mesmo que escrever Index = +1*

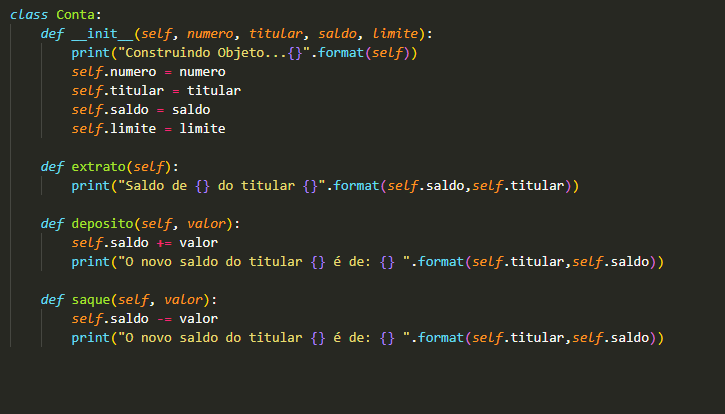
Para melhor organização de códigos, é um padrão positivo e de fácil entendimento à terceiros o uso de funções como forma de separar os “tópicos” que estão sendo tratados ao longo do funcionamento, note que em alguns casos é necessário ***return variável***, para que as demais funções obtenham os valores tratados de volta, ou até mesmo a implantação de variáveis em () para que sejam tratados de forma isolada na sua função definida.

**Iniciando com Orientação a Objetos Python**

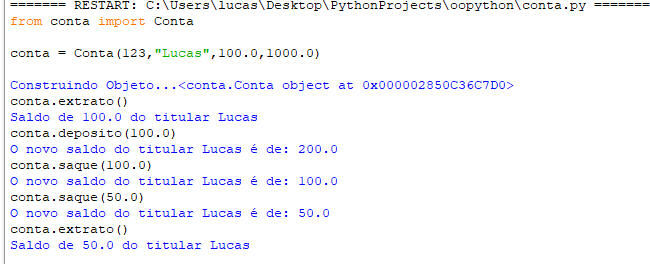
As classes podem ser tratadas com ***class NomeDaClasse***,

Os atributos dos objetos são definidos a partir de funções construtoras: **\_\_init\_\_(self)**

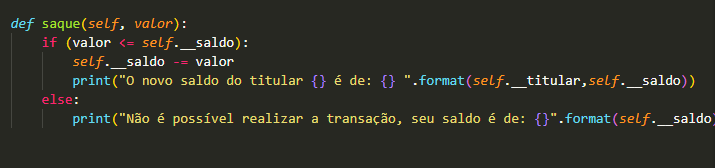
Onde o ***self*** é referente ao caminho onde está armazenado o objeto e é utilizado para criar os atributos dos objetos = ***self.nome do atributo = valor a ser atribuído***

******

Veja que alguns atributos são referenciados com self. titular e saldo para que esses métodos tenham acesso aos valores dos objetos; também é possível definir uma variável juntamente da função criada para que receba valores e possam manipular, como no exemplo de ***valor*** nas funções de ***deposito e saque***

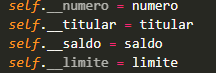
Terminal IDLE do Python, foi buscado o arquivo ***conta*** e ***import*** a ***class*** ***Conta***, além de criado o objeto de nome ***conta*** e passado respectivos valores para serem tratados de exemplos nas funções.

Note que é possível acrescentar uma condição de limite para a função ***def saque***, onde ***if (valor <= self.saldo):***, então ***self.saldo -= valor***

****

**Encapsulamento**

Quando há necessidade de deixarmos atributos privados de acesso à métodos que não sejam por funções definidas, é possível a partir de dois **\_\_** underline no início do nome do atributo, exemplo: ***self.\_\_numero = numero ; self.\_\_titular = titular; self.\_\_saldo = saldo***

***Private nome da variável = \_\_nome do atributo***

**Getters e Setters** utilizado em atributos, para que seja possível realizar retorno ou modificações diretamente

Getter retorna valores

Setter atribui ou modifica valores

Getter de forma reduzida/abreviada =

@property

def saldo(self):

return self.\_\_saldo

**Herança, Polimorfismo**