**Preparatório para Exame 98-381**

**Introdução à programação com o Python**

[**herickfreitas@hotmail.com**](mailto:herickfreitas@hotmail.com)

**27/03/2018**

Sumário – Roteiro de aprendizado

**1.** [**Introdução 1**](#_Toc509940889)

**2.** [**Exibição de texto 1**](#_Toc509940889)

**3.** [**Variáveis de cadeia de caracteres 1**](#_Toc509940889)

**4.** [**Armazenamento de números 1**](#_Toc509940889)

**5.** [**Como trabalhar com datas e horas 1**](#_Toc509940889)

**6.** [**Tomada de decisões com o código 1**](#_Toc509940889)

**7.** [**Decisões complexas com o código 1**](#_Toc509940889)

**8.** [**Repetição de eventos 1**](#_Toc509940889)

**9.** [**Repetição de eventos até a conclusão 1**](#_Toc509940889)

**10.** [**Listas de lembranças 1**](#_Toc509940889)

**11.** [**Como salvar informações em arquivos 1**](#_Toc509940889)

**12.** [**Leitura a partir de arquivos 1**](#_Toc509940889)

**13.** [**Funções 1**](#_Toc509940889)

**14.** [**Manipulação de erros 1**](#_Toc509940889)

**15.** [**Recursos recomendados e próximas etapas..................................................................... 1**](#_Toc509940889)

**16.** [**Orientações de demonstração......................................................................................... 1**](#_Toc509940889)

1. **Introdução.**

**Qual o público alvo deste curso?** Qualquer pessoa interessada em aprender, estudante, profissionais de TI, profissionais de Ciências Exatas.

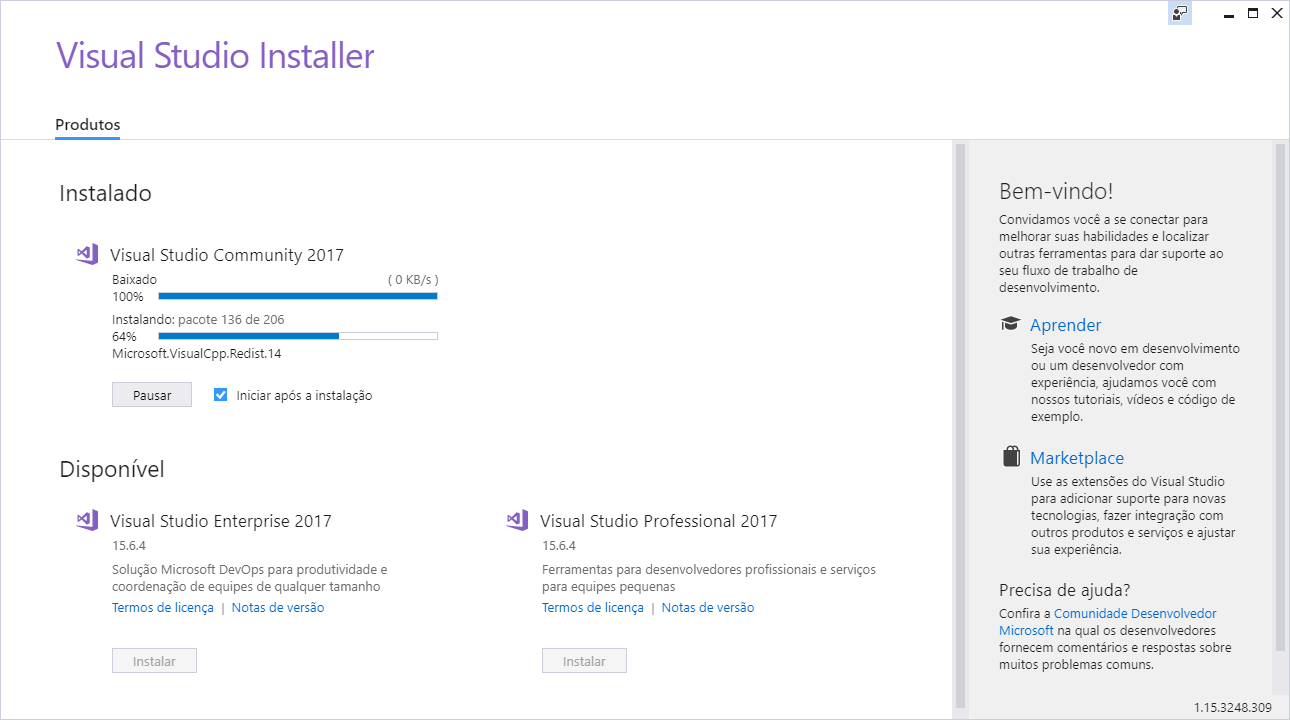
**Por que escolher Python?** Python possui uma sintaxe de baixa complexidade, é gratuito, aplicação leve, possui uma infinidade de recursos e aplicações...

**Qual software iremos usar para programar?** Visual Studio Community, é gratuito e pode ser encontrado facilmente ne web para download.

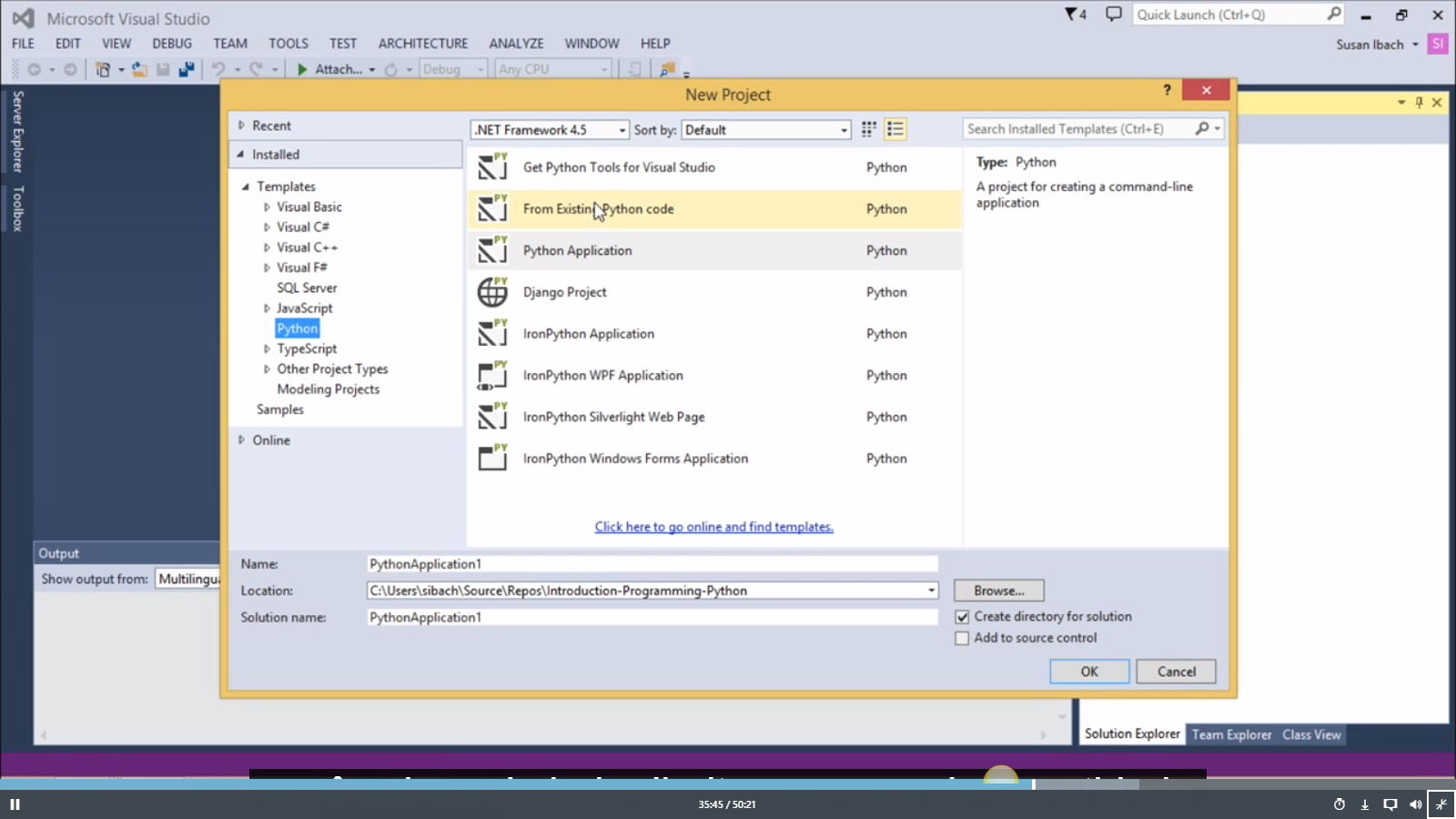
**Qual finalidade de um programa?** Um programa resolve problemas, facilitando a vida.

**Onde Python é usado?** O Python é muito utilizado para coleta de dados, mineração de dados, robótica, edição de filmes, construção de sites, etc...

**Como é o processo de instalação?** Pesquise me algum site de busca “download python for visual studio free”, será direcionado para um portal da Microsoft onde será disponibilizado algumas versões para download. Baixe a versão Community e selecione a adição do módulo de programação em Python. Estou usando a versão 2017.



Após a instalação crie um novo projeto em Python. Selecione Python application. Crie um projeto chamado HelloWorld

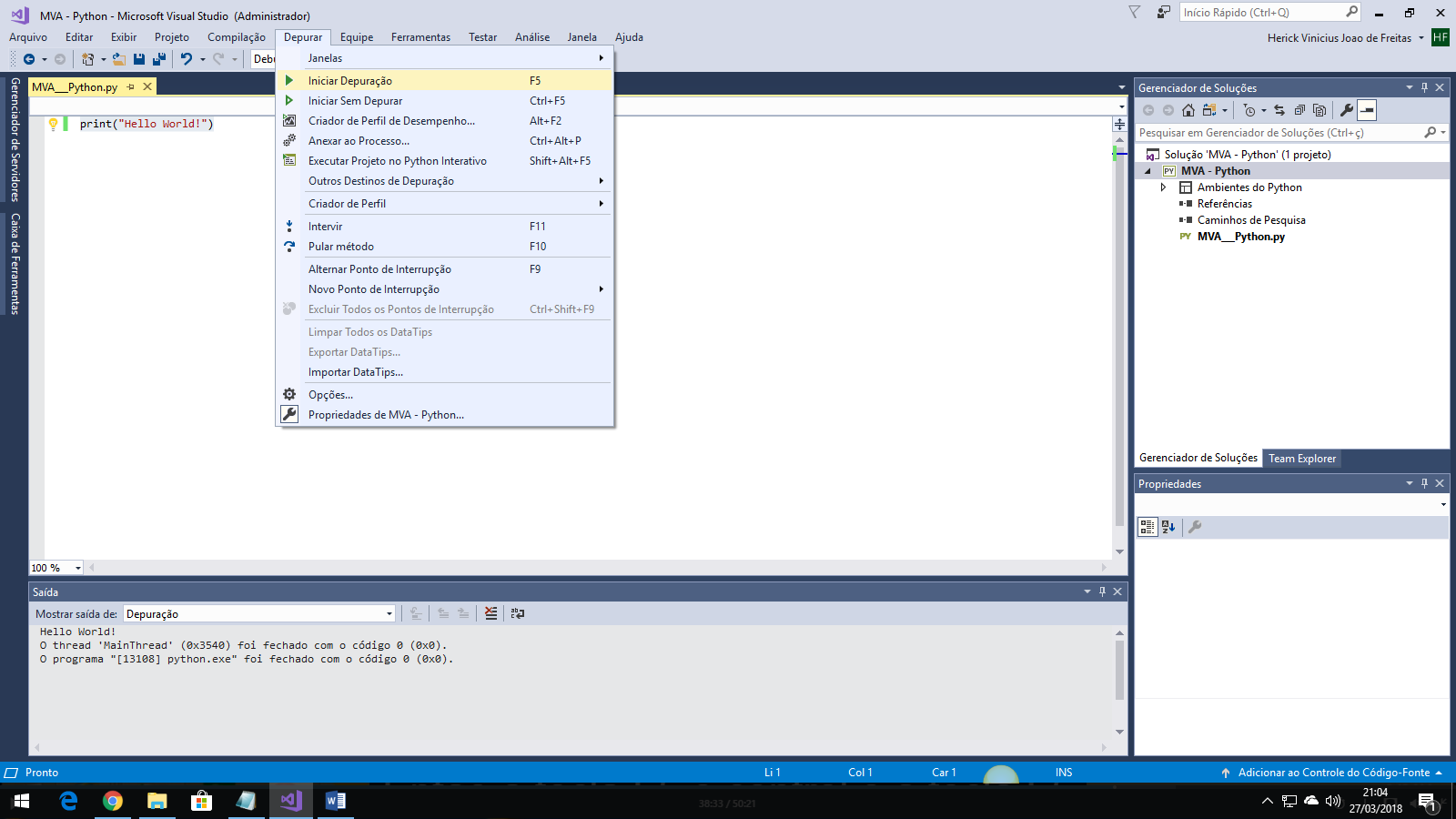


Será criado um arquivo com extensão .py que é a extensão padrão Python.

Crie um código para impressão da mensagem Hello World!.

print("Hello World!")

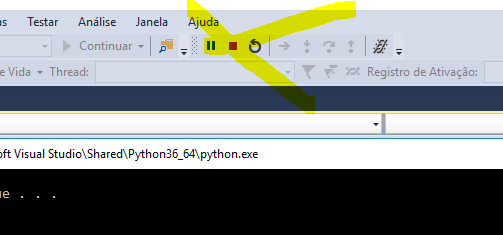
O código poderá ser executado com F5 para que ocorra a depuração da programação.



Como o código representa somente uma saída de dados, será apresentada a mensagem.



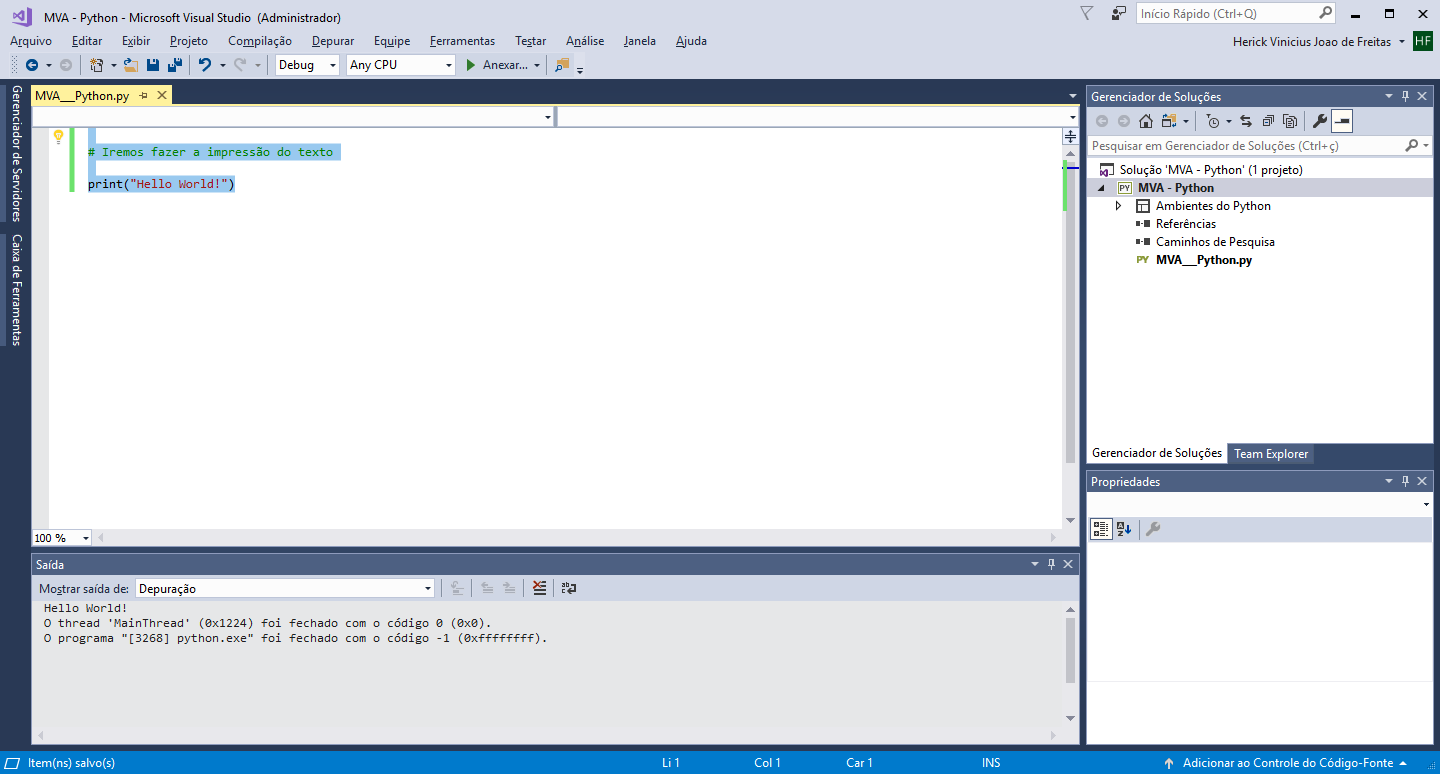
Para encerrar a execução do programa você pode fechar a tela dar Enter ou clicar o Stop na barra superior.



Desenvolva bons hábitos desde o início, comente sempre seu código.

# Iremos fazer a impressão do texto

print("Hello World!")



Use o início rápido no canto superior direito, para fazer pesquisas e sanar dúvidas sobre sua programação.

1. **Exibição de texto.**

O Python possui uma abordagem muito simples quando trabalhamos com textos, basicamente um texto é identificado quando está entre aspas(‘ ‘) ou aspas duplas(“ “). Então conforme o projeto Olá mundo quando associamos um texto ao comando print() efetuamos uma impressão.

Efetuando impressão de várias linhas, o comando print() fará uma impressão contínua de todo o texto contido entre parênteses. Então uma maneira menos prática seria uma print() por cada linha que deseja imprimir.

print("Além destas, outras cinco pessoas são alvos de prisão temporária – que valem por cinco dias, prorrogáveis por mais cinco. No total,")

print("dos 11 pedidos de prisão, seis foram expedidos a professores. A operação cumpre, ainda, 12 mandados de busca a apreensão.")

Uma segunda opção e mais prática é usar um texto inteiro em um só print(), efetuando a quebra de linha com \n.

print("Além destas, outras cinco pessoas são alvos de prisão temporária – que valem por cinco dias, prorrogáveis por mais cinco. No total,\n dos 11 pedidos de prisão, seis foram expedidos a professores. A operação cumpre, ainda, 12 mandados de busca a apreensão.")

Também há a opção de informar o texto entre aspas triplas, tanto faz se é simples ou duplas. Assim o texto será impresso exatamente com a quebra de linha que for escrito.

print("""Além destas, outras cinco pessoas são alvos de prisão temporária – que valem por cinco dias, prorrogáveis por mais cinco. No total,

dos 11 pedidos de prisão, seis foram expedidos a professores. A operação cumpre, ainda, 12 mandados de busca a apreensão.""")

Um outro recurso é concatenar texto com ( + ), assim dois ou mais textos podem ser representados de maneira única.

print("Além destas, outras cinco pessoas" + "são alvos de prisão temporária – que valem por cinco dias, prorrogáveis por mais cinco.")

Há um recurso que pode ser aplicado quando precisamos usar aspas duplas e simples no mesmo texto. Use \ antes das aspas para que elas sejam abordadas como texto.

print("Além destas, outras cinco pessoas \"são alvos de prisão temporária\" – que valem por cinco dias, prorrogáveis por mais cinco.")

Se precisar aplicar uma tabulação no texto use \t

print("2kg\tFeijão")

Quando precisar inserir no texto alguma palavra com \ antecedendo use \\ para não gerar quebra ou tabulação no texto.

print("Acesse o site usando \\news")

1. **Variáveis de cadeia de caracteres.**

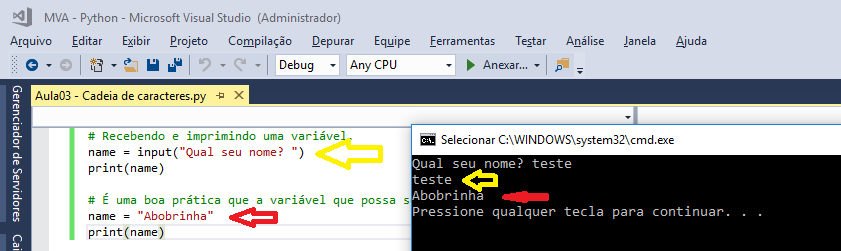
Toda programação parte de uma ação ou interação com mundo exterior, seja através de uma entrada de dado textual ou coleta de dados através de imagens, áudios, websites, IoT, robótica, etc....

Então, esta entrada se dá através de um input de informações, este input será armazenado em uma variável e esta variável será utilizada pelo programa.

# Recebendo e imprimindo uma variável.

name = input("Qual seu nome? ")

print(name)



É muito importante que se uma variável possa ser usada mais de uma vez ela deva ser limpa ou resetada para que não ocorra erros.

As regras de criação de variáveis são: Não pode conter espaços, CaseSensitive e não pode começar com número.

Boas práticas:

Tenha objetividade na escolha de variáveis exemplo: Rua já referencia o cadastro da variável rua, não use RuaDoCliente ou RDC.

Use variáveis com nomes que as referenciem evite X, Y, etc... Uma variável pode conter até 82 caracteres. Escolha um esquema como padrão, use letras maiúsculas ou minúsculas, exemplo DataNascimento ou datanascimento.

Trabalhando com variáveis.

Vamos criar duas variáveis e concatena-las.

nome = input("Qual seu nome? ")

pais = input("Qual Pais você vive? ")

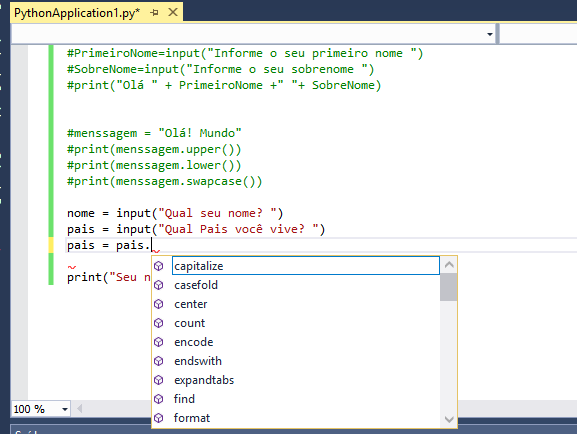
pais = pais.upper()

print("Seu nome é " + nome + " e você vive em " + pais)

Funções de cadeias de caracteres

Serve para manipular o texto, alterando para maiúscula ou minúscula, dentre outros recursos.

O Visual Studio possui recurso de sugestão durante a digitação do código chamado de IntelliSense, assim você receberá algumas dicas dos recursos disponíveis ou use Ctrl + J ou Ctrl + SPACE



Não é possível memorizar todos os recursos, então use as ferramentas disponíveis. F1 é o help, use o intellisense, documentação, sites de busca, etc...

Uma boa prática de programação é sempre declarar as variáveis. Informando-a(as) zerada(s) ou vazia(s).

nome = ""

pais = ""

nome = input("Qual seu nome? ")

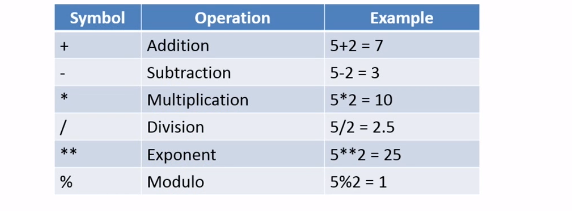
pais = input("Qual Pais você vive? ")

pais = pais.upper()

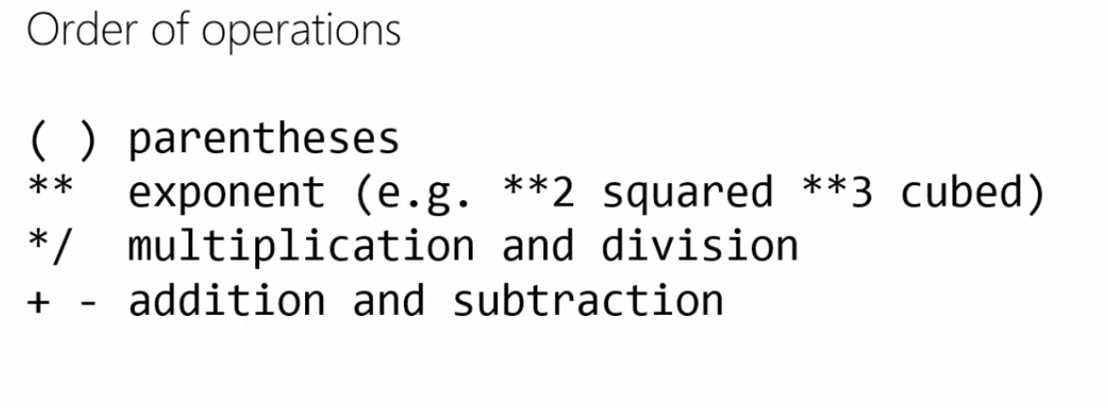
print("Seu nome é " + nome + " e você vive em " + pais)

1. **Armazenamento de números.**

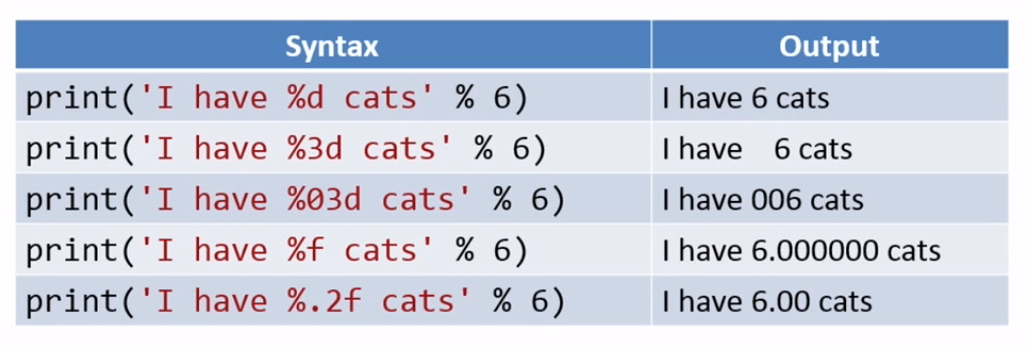
Assim como possuímos variáveis do tipo String, também possuímos variáveis numéricas. Certamente necessária para maior parte das aplicações de programação em Python. De manera nativa o Python reconhece os operadores matemáticos e realiza suas operações.



Durante a execução das equações matemáticas há uma ordem de execução, segue uma representação de como o Python trata essas operações.



Então não será eventual a necessidade de associarmos na saída de dados informações referentes a variáveis de tipos diferentes, além disso representando informações numéricas como inteiros ou pontos flutuantes, etc...



Salvo que na representação a informação d representa decimal e f representa ponto flutuante.

# Manipulação de números

# Calculando área de um triangulo

area = 0

largura = 10

altura = 10

area = (largura \* altura) / 2

# impressão decimal

print("A área do tringulo é %d" %area)

# impressão flutuante

print("A área do tringulo é %f" %area)

# impressão flutuante com duas casas decimais

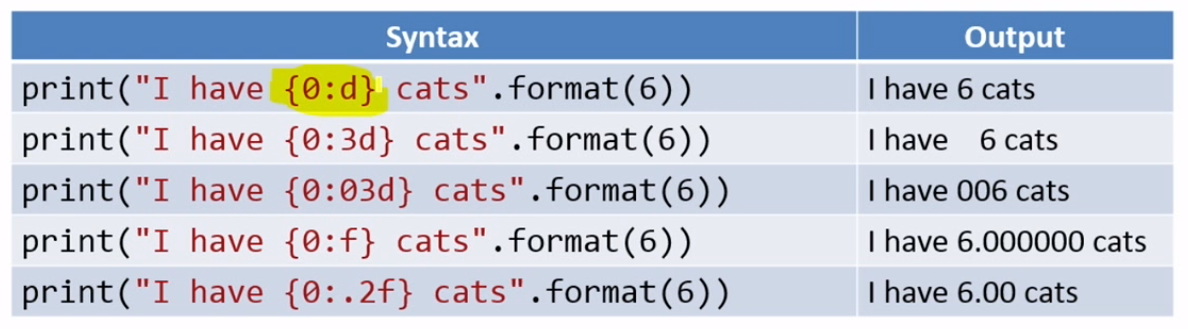
print("A área do tringulo é %.2f" %area)

# impressão preenchendo com zeros a esquerda, muito útil para

# relatórios, refere-se a quantidade de espaços.

print("A área do tringulo é %03d" %area)

Há outro método de fazer a impressão, este método foi disponibilizado a partir da versão 3 do Python.



O {0: presente em todas impressões referencia que é o primeiro item dentro do .format().

# Manipulação de números

# Calculando área de um triangulo usando format

area = 0

largura = 10

altura = 10

area = (largura \* altura) / 2

# impressão decimal

print("A área do tringulo é {0:d}".format(500))

# impressão flutuante

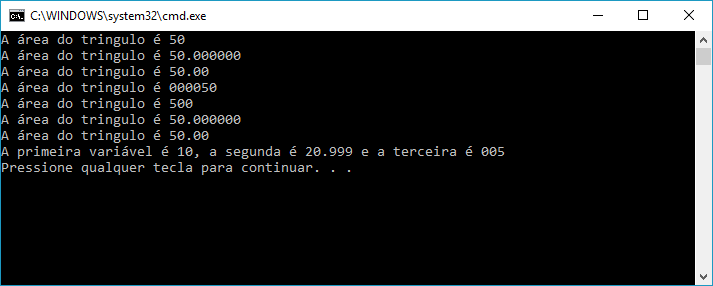
print("A área do tringulo é {0:f}".format(area))

# impressão flutuante com duas casas decimais

print("A área do tringulo é {0:.2f}".format(area))

# Agora iremos validar a impressão com multimas variáveis.

print("A primeira variável é {0:d}, a segunda é {1:.3f} e a terceira é {2:03d}".format(10,20.999,5))



Atenção: Talvez esteja escrevendo um comando muito extenso que esta ficando muito grande em uma linha, use \ barra invertida e continue na linha a seguir, Python irá considerar que se trata de uma comando único.

Total = 2 + 4 \* 40 / 30 + 50 \

40 + 10 – 20

Também é possível quebrar a sintaxe em duas. É muito importante que a linha seguinte possui tabulação com a superior, assim o código fará uma associação.

# Agora iremos validar a impressão com multiplas variáveis.

print("A primeira variável é {0:d}, a segunda é {1:.3f} e " +

"a terceira é {2:03d}".format(10,20.999,5))

Operações matemáticas

Quando formos realizar operações matemáticas é muito importante classificar as variáveis, pois senão serão tratadas como String.

# Agora iremos representar operações matemáticas.

# É muito importante classificar suas variáveis durante o processo de calculo.

salario = input("Informe seu salário " )

comissao = input("Informe sua comissão ")

# Valor não classificado

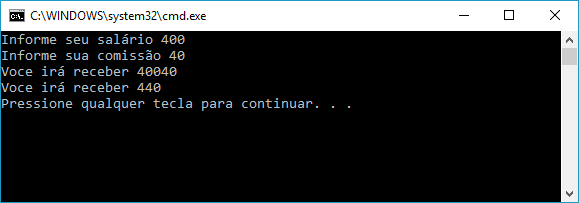
cheque = salario + comissao

print("Voce irá receber {0}".format(cheque))

# Valor classificado

chequeFinal = int(salario) + int(comissao)

print("Voce irá receber {0}".format(chequeFinal))



A questão anterior foi solucionada, mas não da maneira ideal. A condição ideal era que o dado fosse tratado na entrada, assim ficaria armazenado no formato correto.

# Reescrevendo o código usando boas práticas.

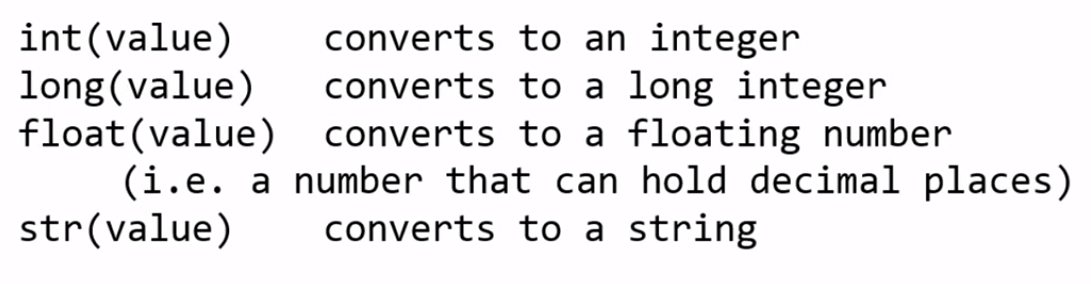
salarioInt = float(input("Informe seu salário " ))

comissaoInt = float(input("Informe sua comissão "))

chequeInt = salarioInt + comissaoInt

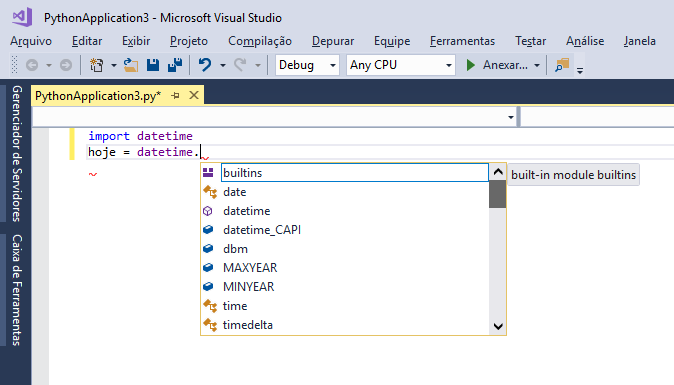
print("Voce irá receber {0:.2f}".format(chequeInt))

**Opções para conversão de variáveis.**



1. **Como trabalhar com datas e horas.**

Este bloco tem como objetivo abordar datas, sempre na programação haverá a necessidade de controlar prazos, históricos, agendamentos, etc... Então para esta finalidade serão aplicados informações ou valores de data e hora. Então para atender este objetivo iremos abordar recursos adicionais, mas precisamente a biblioteca datetime, que irão disponibilizar recursos ou funções que agregam recursos ao Python.



Eventualmente teremos a necessidade de representar data de forma escrita ao invés de representações por números. Para isso será necessário usar o recurso **strftime** assim será possível especificar o formato da representação.

# Aula 05 - Trabalhando com datas

import datetime

dataAtual = datetime.date.today()

print(dataAtual)

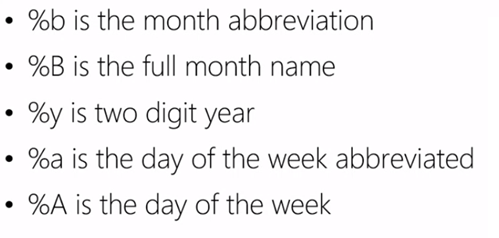
print(dataAtual.year)

print(dataAtual.month)

print(dataAtual.day)

print(dataAtual.strftime("%d %b, %Y"))

Para mais informações acesse [www.strftime.org](http://www.strftime.org)



Uma outra informação importante é que podemos informar na programação os parâmetros de localização, assim as configurações de idiomas e datas serão absorvidas no código de programação. Acesse <http://babel.pocoo.org/en/latest/>

**strptime()** pega um vetor de caractere e faz um objeto de data e hora.

Isso é entrada.

**strftime()** pega um objeto de data e hora e faz um vetor de caractere.

Essa é a saída e normalmente é chamada via format () ou print ().

Um recurso muito interessante é a possibilidade de comentar um texto grande, assim selecionando o texto e usando Ctrl+K em seguida Ctrl+C, todo o texto selecionado será comentado. Ctrl+K e Ctrl+U remove o comentário do texto.

Trabalhando com tempo.

# Trabalhando com tempo

import datetime

currentTime = datetime.datetime.now()

print(currentTime)

print(currentTime.hour)

print(currentTime.minute)

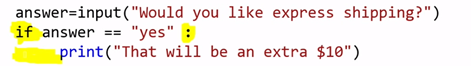
print(currentTime.second)

<https://dateutil.readthedocs.io/en/stable/>

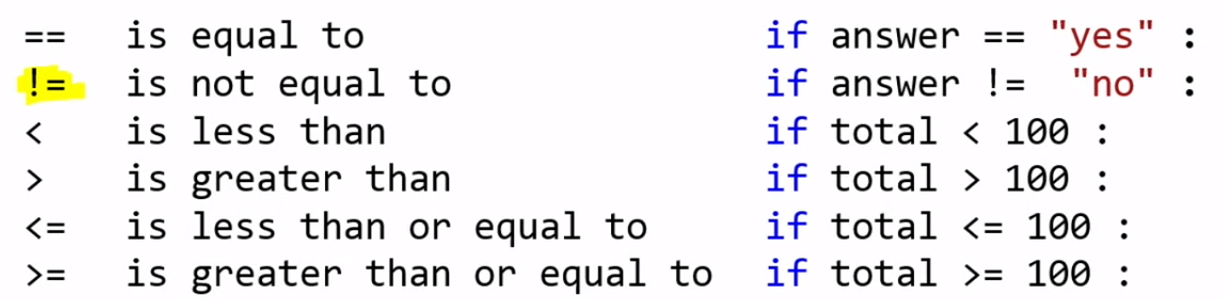
1. **Tomando decisões com o código.**

A programação representa necessidades do mundo real, sua finalidade é desenvolver soluções para estes problemas. Então o processo decisório também deve ser representado e aplicado na programação.

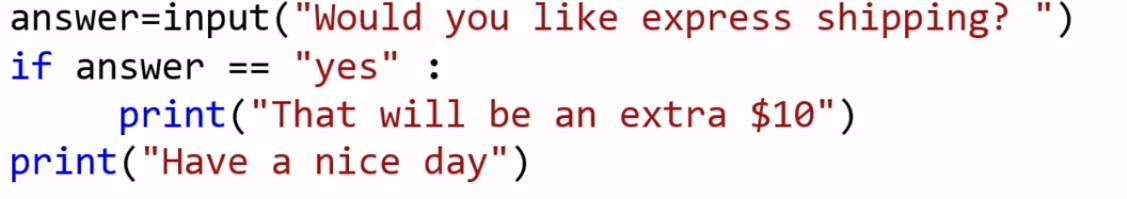
Vamos analisar uma sintaxe simples.



**Operadores condicionais**



Ao avançarmos na programação iremos identificar que se a clausula if for atendida ou não devemos dar alguma resposta, segue exemplo.



# Tomando decisões com codigo

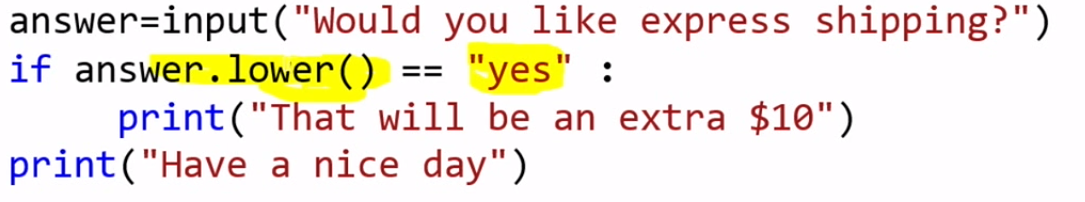
timeFavorito = input("Qual seu time? ")

if timeFavorito == "Flamengo" or timeFavorito =="flamengo":

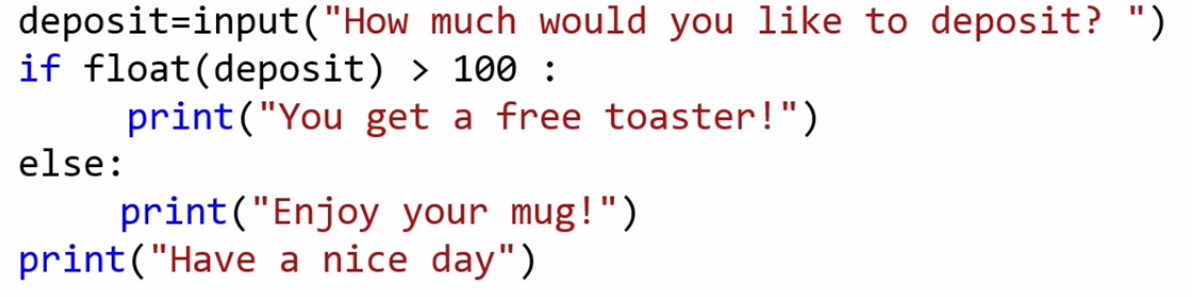
print("Parabêns seu time é top!")

print("Boa tarde!")

No exemplo a seguir irei representar um erro recorrente, pois podemos colocar o valor da variável de entrada como caixa alta ou baixa. Mas se o valor de comparação não acompanhar o valor de origem o código não vai funcionar.



Chegamos ao ponto em que se o if não for atendido devemos tomar outra ação, para isso use else. Segue sintaxe.



Também podemos usar variáveis boleanas no processo de decisão, assim quando fizermos referência da variável em vários pontos do código será mais fácil de gerenciar.

# Usando variaveis boleanas para tomar decisões

deposito = 100

if deposito > 500:

brinde = True

else:

brinde = False

if brinde :

print("Parabens, você ganhou um brinde! ")

print("Tenha um bom dia! ")

**Opção 2**

# Usando variaveis boleanas para tomar decisões

deposito = 600

brinde = False

if deposito > 500:

brinde = True

if brinde :

print("Parabens, você ganhou um brinde! ")

print("Tenha um bom dia! ")

1. **Decisões complexas.**

Este bloco tem como objetivo abordar o avanço do bloco anterior, assim apresentando na programação como as decisões mais complexas devem ser aplicadas. Para isso iremos usar **if** e **elif**, quando qualquer uma das possibilidades forem atendidas a execução irá saltar para um item posterior a esta chamada **if, elif.**

# Decisões complexas.

# Vamos abordar um exemplo de possibilidades de respostas dependendo do idioma de entrada.

idioma = input("Qual idioma o Sr.(a) fala? ").upper()

if idioma == "PORTUGUES":

print("Bom dia ")

elif idioma == "INGLES":

print("Good Morning")

elif idioma == "ESPANHOL":

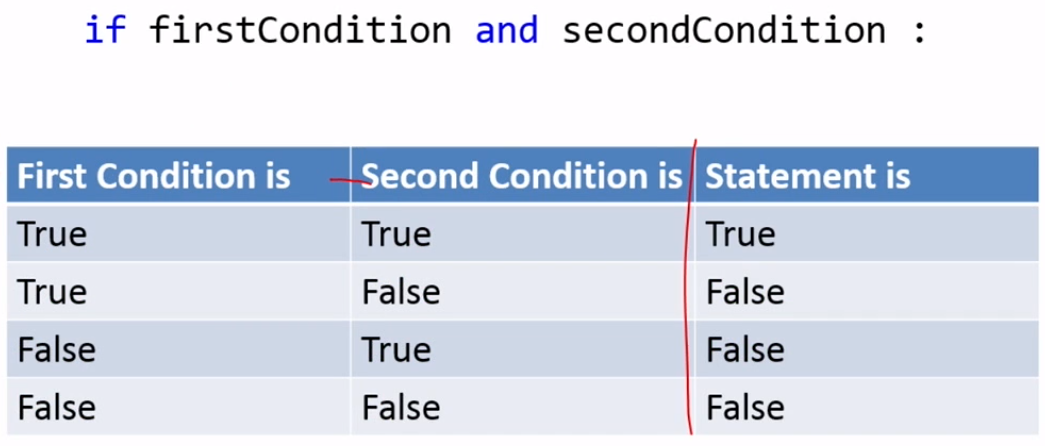
print("Buenos dias")

O código representado acima é funcional, desde que possua uma entrada controlada, como um painel de seleção que o usuário irá somente selecionar um dos três idiomas informados. Caso seja informado alguma resposta fora das opções a programação não irá funcionar. Para solucionar esta questão use **else**.

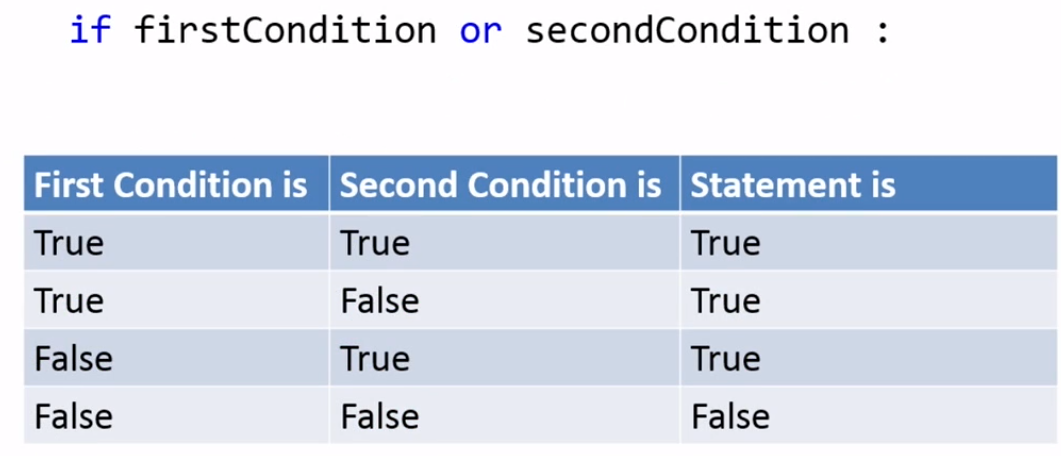
else:

print("Idioma desconhecido")

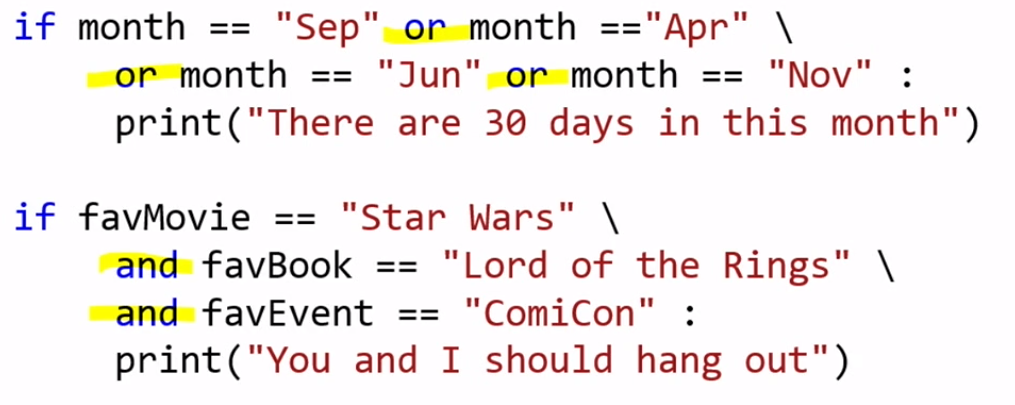
Também iremos nos deparar com processos decisórios onde duas ou mais condições devem ser atendidas, então para isso use **and**. Salvo que se uma das condições não forem atendidas este bloco de código não será executado.



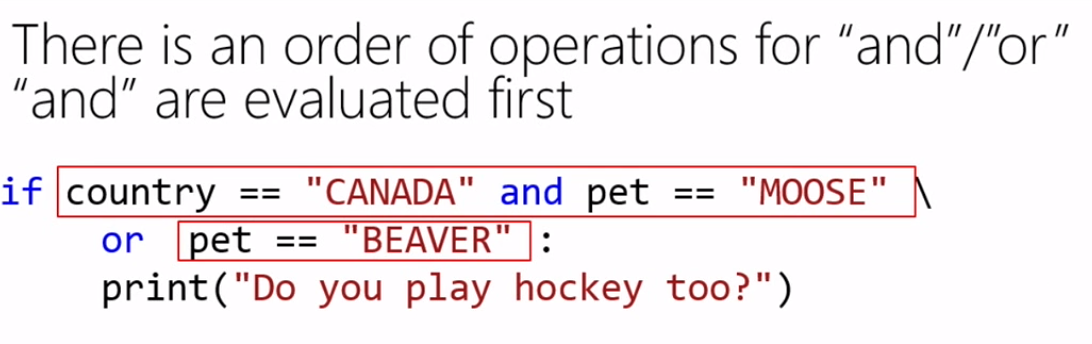
Assim como **and** foi aplicado também podemos aplicar **or**, mas o funcionamento é diferente. Caso uma das condições seja atendida o código será executado. Só não ira funcionar se as duas condições forem falsas.



Aplicação



Também é possível aplicar em conjunto **and** e **or**.



Mas a forma representada acima irá apresentar erro, pois irá retornar verdadeiro caso um dos pets sejam verdadeiros, independente da primeira validação country. Como solucionar isto:

# Validando and e or na mesma instrução, para isso iremos representar um processo de brindes de cartão de credito

# Se o cartão for outro ou prata e o consumo superior a valor X, o titular do cartão ganhará um brinde.

tipoCartao = input("Informe o tipo do cartão: ").upper()

fatura = 1300.00

if fatura >= 1100.00 and (tipoCartao == "OURO" or tipoCartao == "PRATA"):

print("Parabêns! Você ganhou um brinde")

print("Tenha um bom dia!")

1. **Repetição de eventos.**

Para facilitar o aprendizado iremos utilizar a biblioteca turtle, que serve para efetuarmos desenhos baseados em direções e cores definidas no código.

# Usando a biblioteca turtle podemos representar um desenho como processo

import turtle

turtle.forward(100)

turtle.right(90)

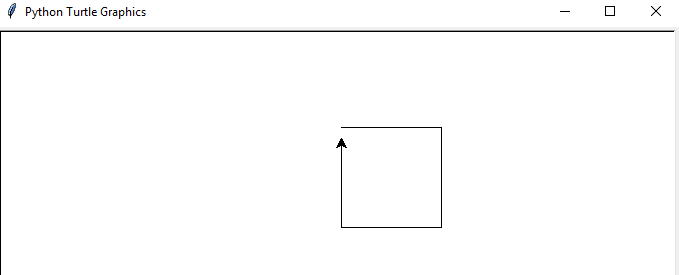
turtle.forward(100)

turtle.right(90)

turtle.forward(100)

turtle.right(90)

turtle.forward(100)



# Usando a biblioteca turtle com loop para desenha o mesmo quadrado

import turtle

for steps in range(4):

turtle.forward(100)

turtle.right(90)

# Também podemos colocar um loop dentro de outro loop

import turtle

for steps in range(4):

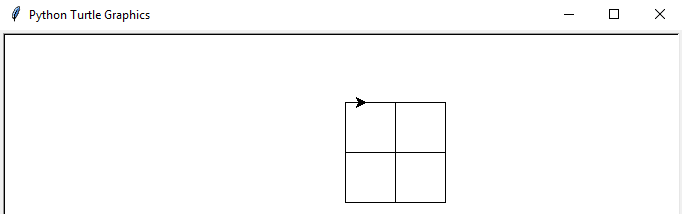
turtle.forward(100)

turtle.right(90)

for steps in range(4):

turtle.forward(50)

turtle.right(90)



# Tambem é possivel usar a biblioteca para realizar desenho mais elaborados

import turtle

for steps in range(5):

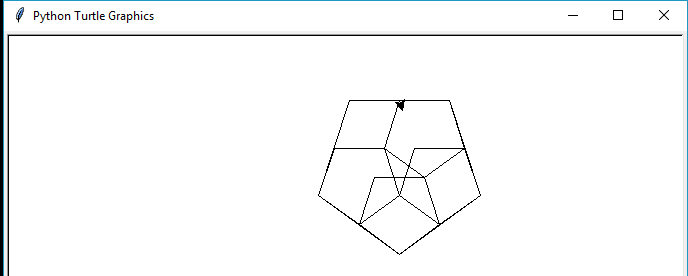
turtle.forward(100)

turtle.right(360/5)

for steps in range(5):

turtle.forward(50)

turtle.right(360/5)



# Uma boa prática de programação é passar os parâmetros em variáveis, assim reduzirá a mão de obra numa manutenção futura

import turtle

nr\_lados = 4

for steps in range(nr\_lados):

turtle.forward(100)

turtle.right(90)

for steps2 in range(nr\_lados):

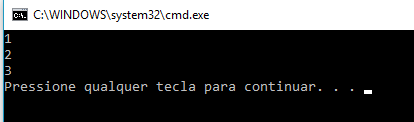
turtle.forward(50)

turtle.right(90)

# Podemos tambem definir um início e fim para o range de execução.

for steps in range(1,4):

print(steps)

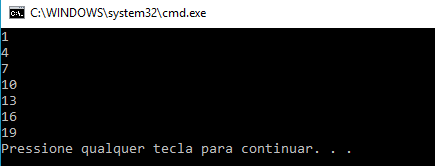


A contagem será iniciada em 1 mas não alcançará o 4, pois usa o conceito de até 4 no caso é a variável limite.

# Podemos tambem definir um salto dentro do início e fim do range. O salto é 3

for steps in range(1,20,3):

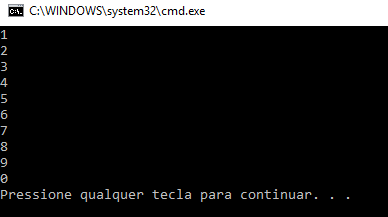
print(steps)



# Também há a necessidade de trabalharmos com arrays de variáveis. Salvo que serão sempre strings - Não carrega a informação range

for steps in [1,2,3,4,5,6,7,8,9,0]:

print(steps)



# Vamos representar uma execução de loop com usando a informação do array

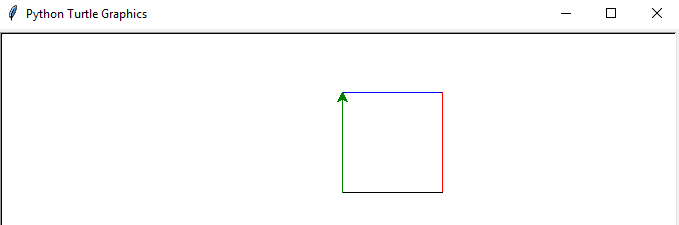
import turtle

for cores in ["blue","red","black","green"]:

turtle.color(cores)

turtle.forward(100)

turtle.right(90)



1. **Repetição de eventos ate a conclusão.**

No modulo anterior efetuamos a programação usando for, mas o for funciona usando um range ou um array de execução. Agora iremos abordar a execução do programa até que uma demanda seja atendida. Ou ate mesmo uma execução constante, no aguardo de uma entrada.

Exemplos: Lavar louça até o fim, servir um café ao cliente que adentrar o estabelecimento, coleta contínua de dados de sensores...

Então o programa irá funcionar da seguinte maneira:

# while = Enquanto, então até que a resposta não seja atendida o sistema irá executar...

resposta = 0

while resposta != "12":

resposta = input("Quanto é 28 - 16 : ")

print("Você errou")

print("Parabéns, resposta correta ")

# Vamos misturar o while com turtle, para analisarmos o funcionamento

import turtle

contagem = 0

while contagem < 4:

turtle.forward(100)

turtle.left(90)

contagem = contagem + 1

# O incremento também pode ser executado assim

contagem += 1