Aprenda a programar python em 12 páginas

Aprendizado garantido

Gabriel Botelho Malenowitch

Bem, como o objetivo é ensinar a programar em 10 páginas, vou ser o mais direto possível.

O que é Python?

Python é uma linguagem de programação interpretada, orientada a objetos, de alto nível e com semântica dinâmica. A simplicidade do Python reduz a manutenção de um programa. Python suporta módulos e pacotes, que encoraja a programação modularizada e reuso de códigos.

É uma das linguagens que mais tem crescido devido sua compatibilidade (roda na maioria dos sistemas operacionais) e capacidade de auxiliar outras linguagens. Programas como Dropbox, Reddit e Instagram são escritos em Python. Python também é a linguagem mais popular para análise de dados e conquistou a comunidade científica.

+

Se você está usando windows, saiba que o python não vem instalado nele por padrão, neste caso você deve baixar a versão mais recente em https://www.python.org sem as aspas.

+

Após ter baixado e instalado python chegou a hora de baixar um programa para escrever o código de forma mais confortável e fácil, chamado "IDLE". O python já vem com seu próprio IDLE porém não é boa para escrever longos códigos.

+

O "pycharm" é uma IDLE muito comum entre os programadores de python, porém é pesada para alguns computadores de baixo desempenho rodarem, se for o seu caso, você pode baixar e instalar o "VS Code".

Se você estiver usando "VS Code" é necessário instalar o python dentro da sua "lojinha" interna, caso esteja usando pycharm essa configuração não é necessária.

+

Tudo preparado é hora de começarmos a estudar:

Dentro do python usamos aspas únicas e aspas duplas " ', porém se iniciar uma parte de código com uma delas, deve terminar usando o mesmo tipo.

Ex:

print("Hello, world!")

A função "print()" irá printar o que você escrever entre as aspas. (Para executar o código basta apertar a tecla "F5")

Variáveis:

Na programação, uma variável é um objeto capaz de reter e representar um valor ou expressão. Enquanto as variáveis só "existem" em tempo de execução, elas são associadas a "nomes", chamados identificadores, durante o tempo de desenvolvimento.

Existem alguns tipos de variáveis, porém irei listar 3, que são as mais importantes:

str: define a variável como texto

int: define a variável como número inteiro

float: define a variável como número com uma casa após a virgula (Não é usado virgula para mexermos com números na programação. Usamos o ponto ex: 99.89)

Para definirmos uma variável devemos usar o sinal de igual “=”, pois no python ele tem a função de atribuição, veja o exemplo a baixo:

variavel = tipo_da_variavel("este é o valor da variável")

Não é obrigatório colocar o tipo da variável para a definir, pois por padrão elas são definidas pelo python como str.

ex:

a = str("hello, World!")

ou então

a = str("9.89")

no caso acima a variável está armazenada no sistema como uma string ou seja, ela exercerá um papel de uma palavra, portanto não será possível fazer nenhum tipo de operação matemática com ela.

Obs: input() é uma função utilizada para extrair um dado do usuário, como se fosse uma pergunta.

Vamos escrever um pequeno código aqui:

```
x = input("Qual é o seu nome?")

print(f'Seu nome é {x}.')
```

Nestas duas linhas o código te perguntou seu nome e após você ter apertado "enter" ele o escreveu com a frase "Seu nome é" e logo em seguida seu nome com um ponto no final.

Essas duas chaves "{}" utilizadas no código servem para marcar onde o python deve interpretar a variável, mas isso só acontece se houver um "f" minúsculo antes das aspas que as chaves estão inseridas(Será mais fácil programar se você tiver em mente que deve utilizar em 99% das vezes letras minúsculas.)

Operações aritméticas:

+ sinal de soma

- sinal de subtração

* sinal de multiplicação

/ sinal de divisão

> ou < Sinal para diferenciar maior de menor

** elevar ao quadrado

** 1/2 raiz quadrada (na verdade para tirarmos a raiz quadrada de um número, nós elevamos ele a 1 sobre 2, o que significa ** 1/2)

Ex:

```
n  = float(input("Digite um número para ser tirada a raiz quadrada: "))

n1 = n ** 1/2

print(f"A raiz quadrada de {n} é {n1}.")
```

Comentários:

Comentários são linhas do código que são desconsideradas pelo executor do código, geralmente servem para explicar uma parte do código.

Para comentar uma linha usamos “#”

Ex:

```
oi = "Olá, Boa noite"

#isso é um comentário

print(oi)
```

Para comentarmos parágrafos usamos aspas 3x, Ex:

```
"""este paragrafo está

Sendo comentado"""
```

Listas, tuplas e dicionários:

Essas 3 palavras escritas acima são parecidas com variáveis, porém armazenam mais de uma informação, calma, vou dar exemplos.

Listas, você pode defini-las de duas maneiras:

Lista = [] ou Lista = list()

Observe este pequeno código para entender o conceito de lista:

Lista = ["hamburguer ", "caldo de cana", "torrone"]

A contagem em python começa no 0 em diante;

Se você quiser se referir a palavra “hamburguer”, você deve escrever no código:

print(Lista[0])

desta maneira aparecerá apenas a palavra "hamburguer"

se quiser se referir a "caldo de cana", deve escrever:

print(Lista[1])

e assim por diante.

Se você se arrepender de não ter colocado uma palavra/número na lista, pode usar o comando append()

Lista.append("cachorro quente")

E aí fica:

Lista = ["hamburguer ", "caldo de cana", "torrone", "cachorro quente"]

O comando []. pop() vai remover a palavra que você digitar a posição entre parênteses, caso não digite nada será a última palavra/número.

Ex: Lista.pop(0)

Se quiser remover pela palavra e não pela posição utilize o comando " [].remove()"

[].sort() é utilizado para colocar a lista em ordem alfabética/ numérica, se quiser fazer a ordem reversa escreva "reverse = True" entre os parênteses.

Se quiser printar apenas algumas partes da lista utilize [:] Ex:

print(Lista[1:2]) #Se você não colocar nada no lugar de qualquer um dos números também funciona.

Será printado:

["caldo de cana", "torrone "]

Existem muitas funções dentro do python, porém não é possível descrever todas aqui, confesso que nem eu sei todas. Portando cabe a você ir na documentação do python em python.org e estudar as que lhe serão úteis no momento, é assim que se aprende a programar. Também é possível manipular texto, usando funções, como por exemplo len() ou split().

OBS: Se quiser pular uma linha no print utilize \n Ex:

print(' Gabriel \n Botelho \n Malenowitch')

resultado:

Gabriel

Botelho

Malenowitch

Tuplas, você pode defini-las de duas maneiras:

Tupla = tuple()

Ou

Tupla = ()

OBS:TUPLAS NÃO SÃO EDITÁVEIS (decore isso)

Ou seja, você pode exibi-las igual listas, usando "[:]", porém não pode edita-las após defini-la.

Dicionários, você pode defini-los de duas maneiras:

Dicionario = dict()

Ou

Dicionario = {}

Dicionários são mutáveis, e ao contrário das listas e tuplas, dicionários são referidos por nomes, e não por números, Ex de código:

Gabriel = {"idade":16, "altura": 1.70}

Print(Gabriel["idade"])

Será printado “16”

Se quiser adicionar um valor/palavra/ frase utilize:

N = input("Valor: ")

Gabriel["peso"] = N

Você também pode colocar dicionários dentro de listas Ex:

Gabriel = [{"idade":16, "altura": 1.70}, {"idade":72, "altura": 1.91}]

Para me referir a segunda idade, devo usar:

Print(Gabriel[1]["idade"])

Cores no terminal:

Entre as aspas do print digite \033[m e entre a letra "m" e a chave digite os números que determinam a cor da letra, fundo e estilo. Entre elas coloque ponto e virgula.

Estilo: 0, 1, 4, 7 -- Cor da letra: 30 ao 37 -- Cor do fundo: 40 ao 47 ex: print("\033[0;31;47m oi")

Obs: Se quiser juntar dois prints em um só, utilize “end” ex:

print("Gabriel", end = " ")

print("Botelho Malenowitch")

Aparecerá na tela "Gabriel Botelho Malenowitch" sem as aspas.

IF, ELIF, ELSE:

Vamos começar pelas traduções:

If = se

else = se não

elif é uma mistura de if com else (else if = se não se), o que resulta em elif.

*Para cada uma dessas funções devemos dar um espaço de baixo da linha para o que desejamos colocar dentro delas, esse espaço recebe o nome de tabulação(para fazer isso basta pressionar a tecla TAB de seu teclado). Também é necessário colocar dois pontos no final da linha de função. Veja alguns exemplos mais a baixo.

If: serve para atribuir opção ao programa, veja como exemplo esse trecho de código de uma calculadora:

```
n1 = float(input('Digite um valor: ')

x = input(' [1]somar \n [2]multiplicar \n [3]dividir \n [4]subtrair \n Digitar:')

n2 = float(input('Digite outro valor : '))

if x == '1':

    r = n1 + n2

    print(f'{n1}+{n2}={r}')
```

Como você já deve ter percebido, existem dois sinais de igual “=” após o “if”, por que neste caso não estamos atribuindo um valor para if, estamos perguntando ao programa se a variável “x” recebe o valor “1”, ou seja:

Para atribuição usamos apenas “=”, e para ações com funções utilizamos “==”.

else: Serve para falar “se não for aquilo, vai ser isso”, é usado após o “if”. Ex:

```
if x == '1':

    r = n1 + n2

    print(f'{n1}+{n2}={r}')

else:

    r = n1 - n2

    print(f'{n1}-{n2}={r}')
```

elif: e se você quiser colocar mais de uma condição, por exemplo “se digitar + faz isso, ou então se digitar – faz isso, se não faz isso:”

OBS: elif deve estar SEMPRE a baixo de if.

Se não entendeu as linhas acima, basta entender que elif vai ter o mesmo papel da frase “ou então, se”. Observe o exemplo a baixo:

```
if x == '1':
    r = n1 + n2
    print(f'{n1}+{n2}={r}')
elif x == '3':
    r = n1 / n2
    print(f'{n1}/{n2}={r}')
else:
    r = n1 - n2
    print(f'{n1}-{n2}={r}')
```

Agora estamos nos aprofundando um pouco na programação, já entendemos como manipular variáveis, texto, condições(if elif else), variáveis e funções aritiméticas. Chegou a hora de usarmos um pouco deste conhecimento com coisas que não por padrão dentro do seu código python, os chamados “módulos”.

Alguns módulos já vem no python, porém outros você deve instalar.

Para instalar um módulo você deve abrir o cmd, que pode ser aberto apertando a tecla do Windows + R e digitando “cmd”, logo após isso aperte enter.

Você deve estar olhando para uma ela preta que da para digitar algo, não é?

Se a resposta for “sim”, basta digitar:

pip install MODULO_QUE_VOCÊ_QUER_INSTALAR

por exemplo:

pip install pygame

basta esperar instalar e depois já pode fechar essa tela preta, cujo o nome é “promp de comando”.

Para utilizar este módulo em seu código, você deve digitar nas primeiras linhas do código:

Import pygame #Neste caso ele importará o módulo inteiro.

Se quiser importar apenas uma parte do módulo utilize a seguinte linha, em que me refiro ao módulo “random”:

from random import randint

Se você tiver importado um módulo inteiro, deve se referir primeiro a ele e depois as funções que o módulo possui. Ex:

pygame.init()

Essa função inicia o pygame(não são todos os módulos que necessitam um comando de inicialização, porém o pygame precisa)

Se tiver importado apenas uma parte do módulo pode se referir apenas a ela. Exemplo do módulo random:

N = randint(0,100) #Essa função escolhe um número de 0 a 100 de forma aleatória

Lasso de repetição:

Imagine que você deseja fazer um contador regressivo de 10 segundos, não precisa escrever praticamente o mesmo código 10 vezes, você pode usar laços de repetição. Não se preocupe, darei exemplos:

```
from time import sleep

for i in range(-10,0):
    i = i * -1
    print(i)
    sleep(1)
```

Rode esse código na sua maquina para ver como funciona.

O python possui dois tipos de laços de repetição, o “for” e “while”:

Vamos começar pelo for

Você pode passar por todas as palavras de uma lista usando:

```
for i in (NOMEDALISTA):
    print(i)
```

A letra que você colocar depois do for será uma variável. Pode usar também:

```
for n in range(0,19):
    print(n)
```

Outro exemplo muito útil é:

```
for c, c2 in enumerate(NOMEDALISTA):
    print(f'A palavra {c2} está na posição {c} da lista {NOMEDALISTA}')
```

Ou seja, o laço de repetição for é para se basear numa sequência já estabelecida, como por exemplo a contagem de um número.

While: O laço de repetição while(traduzido fica “enquanto”) serve para determinar o que acontecerá enquanto algo.

Por exemplo:

```
x = 0

while x < 51:

    print(x, end = ' ' )

    x += 1
```

Tudo que estiver tabulado dentro do while será repetido, até que x seja “51”. Por isso tem o nome de enquanto, ou seja “este código vai se repetir enquanto x for menor que 51”.

Se você digitar “while True” no lugar de “while x < 51:” o código vai rodar infinitamente ou até você o parar.

OBS: “x += 1” é a mesma coisa que dizer “x = x+ 1”, é possível fazer isso com outros sinais também.

Criando funções: Neste momento sua mente pode estar explodindo te ideias, e vou dizer, aconteceu a mesma coisa comigo quando eu estava aprendendo a programar.

Se até agora você não compreendeu o que é uma função, eu irei te explicar em poucas linhas:

Função na programação é como um verbo no português. Se eu quero esperar um segundo por exemplo, eu utilizo o módutlo “time” e escrevo “time.sleep()”.

OBS: Funções sempre tem parênteses.

Para definirmos uma função usamos o comando “def”, logo depois escrevemos o nome da função e então colocamos parênteses:

```
def EscrevaNaTela(frase):

    print(frase)

EscrevaNaTela('Hello, World!')
```

A palavra entre parênteses (frase) refere-se a uma variável que foi definida após você escrever dentro dos parênteses, e ela só vale dentro da função, portando se você escrever “print(frase)” o programa vai dar erro, pois em seu código não existe uma variável com nome de “frase”.

Também existem variáveis globais, que você deve definir que ela é global, como por exemplo:

```
frase = 'Hello, World!'

def EscrevaNaTela():

    global frase

    print(frase)

EscrevaNaTela()
```

Se quiser fazer a função retornar alguma coisa, escreva return, e o que deseja retornar na frente. Ex:

```
from math import sin

def JuntarNumeros():

    Lista = list()

    for i in range(0,50):

        x = 2*sin(i)

        Lista.append(x)

    return Lista

Numeros_aleatórios = JuntarNumeros()

print(Numeros_aleatórios)
```

Ótimo, agora que você aprendeu a definir funções está pronto para o próximo passo.

Try, except: "Try" é uma palavra em inglês que significa "tentar", e except no nosso caso significa "se não funcionou faz isso".

Você já deve ter reparado que quando seu código está errado ele simplesmente da erro, porém existe uma maneira de solucionar esse problema, fazendo o código tentar executar algo. Exemplo de código em que a variável x não foi definida:

```
try:

    print(x)

except Exception:

    print("Variável x não definida!")
```

Ao invés do código dar erro, ele vai simplesmente dizer que a variável x não foi definida.

Mas se você quiser ser mais profissional e fazer uma exceção para um erro específico, pode escrever da seguinte maneira:

```
try:

    print(x)

except NameError:

    print("Variável x não definida!")
```

classes:

Para iniciarmos uma classe devemos digitar:

```
Class NOME_DA_CLASSE(object):   #Você pode definir o nome da classe como quiser.
    def __init__(self):
        """Aqui você define o principal da classe, você irá entender melhor nos exemplos a seguir, mas
        você deve lembrar que toda vez que for invocado apenas o nome da classe, será executado a
        ação definida aqui. Você compreenderá melhor nos exemplos adiante."""
    Def FUNÇÃO_ALEATÓRIA(SELF):
        #Aqui você define o que quiser
n = NOME_DA_CLASSE() #Aqui você invocou sua classe, portanto ela irá executar a função "__init__()".
NOME_DA_CLASSE.FUNÇÃO_ALEATÓRIA() #Aqui ele irá executar apenas a "FUNÇÃO_ALEATÓRIA"
```

Ex:

```
class Televisão(object):
    def __init__(self):
        self.estado = 'Desligada'
        self.tamanho = '43 polegadas' #O parâmetro "self" é usado para se referir classe "Televisão()"
        print('Método construtor chamado com sucesso')
    def imprimir(self):
        print(f'A televisão tem {self.tamanho} e ela está {self.estado}.')
televisão = Televisão()
televisão.imprimir()
#Aqui, o código executará as duas funções.
```

Criação de módulos:

Para criar um módulo, você deve criar uma pasta no mesmo local que você está executando o seu código principal. Essa pasta deve ter o nome que você deseja dar ao módulo, após isso, você deverá criar um arquivo chamado "__init__.py" dentro desta pasta, e fazer seu módulo dentro dele.

Portanto se você importar o módulo completo, você estará importando apenas esse arquivo (__init__.py).

Mas se você quiser colocar mais arquivos dentro da pasta para serem importados, crie arquivos com o nome desejado e extensão ".py", e para importalos basta usar

"from NOMEDAPASTA import NOMEDOARQUIVO"

Se quiser importar o módulo completo utilize:

import NOMEDAPASTA

```
#Calculadora
n = 'c'
while n in 'c g':
   n1 = float(input('Digite um valor: '))
   x = input(' [1]somar \n [2]multiplicar \n [3]dividir \n [4]subtrair \n [5]raiz quadrada
\n [6]potênciação \n [7]maior \n [8]novos números \n [9]sair do programa \n Digitar:')
   if x != '9':
      n2 = float(input('Digite outro valor : '))
   n = 'c'
   if x == '1':
      r = n1 + n2
      print(f'{n1:.2f}+{n2:.2f}={r:.2f}')
   elif x == '2':
      r = n1 * n2
      print('{:.2f}x{:.2f}={:.2f}'.format(n1,n2,r))
   elif x == '3':
      r = n1 / n2
      print('{:.2f}/{:.2f}={:.2f}'.format(n1,n2,r))
   elif x == '4':
      r = n1 - n2
      print('{:.2f}-{:.2f}={:.2f}'.format(n1,n2,r))
   elif x == '5':
      n2 = 1 / 2
      r = n1 ** n2
      print('A raiz quadrada de {:.2f} é {:.2f}'.format(n1, r))
   elif x == '6':
      r = n1 ** n2
      print('{:.2f} elevado a {:.2f} é igual a {:.2f}'.format(n1, n2, r))
   elif x == '7':
      if n1 > n2:
         print('{:.2f} é maior que {:.2f}'.format(n1, n2))
      else:
         print('{:.2f} é maior que {:.2f}'.format(n2, n1))
   elif x == '8':
      n = 'c'
   elif x =='9':
      break
   else:
      print('O comando digitado não está listado no sistema!')
      n = 'c'
```