Senac
FACULDADE DE TECNOLOGIA E INOVAÇÃO

Linguagem técnica de Programação

Clenio Emidio

Esp.Gov.TI

@clenioemidio

www.cecursos.com.br

Introdução

- Carga horária disciplina: 60h
- Horários
- Celular
- Avaliações
- Manter comunicação
- Contato (clenio.Fonseca@df.senac.br)
- Metodologia: Atividades teóricas, Exercícios , trabalhos, prática em laboratório*, Projetos...

Alguns tópicos

IDE

- Revisar conceitos
- Laços de repetição
- Listas
- Conhecer algumas bibliotecas/framework
- Manipulação de dados – arquivos
- Funções
- Crud – Integração com banco de dados
- *Orientação à objetos - Estrutura modular de aplicações – classes*
- Pandas
- Interface gráfica(TKinter)
- Geração de gráficos - Matplotlib
- Sistema de supermercado* Relógio de ponto * Sistema de tarefas *Sistema de cadastro
- Implementação com banco de dados

O que é Python?

- Python é uma linguagem de programação amplamente usada, interpretada, orientada a objeto e de alto nível com semântica dinâmica, usada para programação de uso geral.
- E embora você possa conhecer a píton como uma cobra grande, o nome da linguagem de programação Python vem de uma antiga série de comédia da BBC chamada **Monty Python's Flying Circus**.
- No auge do sucesso, a equipe de Monty Python estava realizando seus esboços para viver audiências em todo o mundo, inclusive no Hollywood Bowl.
- Como Monty Python é considerado um dos dois principais elementos essenciais para um programador (o outro é pizza), o criador do Python nomeou a linguagem em homenagem ao programa de TV.

2 Quem criou Python?

- Um dos recursos surpreendentes do Python é o fato de que ele é realmente o trabalho de uma pessoa. Normalmente, novas linguagens de programação são desenvolvidas e publicadas por grandes empresas que empregam muitos profissionais e, devido às regras de direitos autorais, é muito difícil nomear qualquer uma das pessoas envolvidas no projeto. Python é uma exceção.
- Não há muitos idiomas cujos autores sejam conhecidos pelo nome. Python foi criada por **Guido van Rossum**, nascida em 1956 em Haarlem, na Holanda. Claro, Guido van Rossum não desenvolveu e evoluiu todos os componentes do Python...
- A velocidade com que o Python se espalhou pelo mundo é resultado do trabalho contínuo de milhares (muitas vezes anônimos) programadores, testadores, usuários (muitos deles não são especialistas em TI) e entusiastas, mas é preciso dizer que os a primeira ideia (a semente da qual Python brotou) veio à tona - o de Guido.

3- Objetivos do Python

- Em 1999, Guido van Rossum definiu seus objetivos para o Python:
- uma linguagem **fácil e intuitiva**, tão eficiente quanto a dos grandes concorrentes;
- **código aberto**, para que qualquer pessoa possa contribuir com seu desenvolvimento;
- código tão **compreensível** quanto o simples inglês;
- **adequado para tarefas diárias**, permitindo tempos de desenvolvimento curtos.

4- Por que Python?

É fácil de aprender - o tempo necessário para aprender Python é mais curto do que para muitas outras linguagens; isso significa que é possível iniciar a programação real mais rapidamente;
é fácil de ensinar - a carga de trabalho de ensino é menor do que a necessária em outros idiomas; isso significa que o professor pode colocar mais ênfase nas técnicas de programação geral (independentes do idioma), não desperdiçando energia em truques exóticos, estranhas exceções e regras incompreensíveis;
É fácil de usar para escrever novos softwares - muitas vezes é possível escrever código mais rápido ao usar Python;
é fácil de entender - também é mais fácil entender o código de outra pessoa com mais rapidez, se for escrito em Python;
É fácil de obter, instalar e implantar – Python é gratuito, aberto e multiplataforma; nem todas as linguagens podem se orgulhar disso.

5 Rivais do Python?

- O Python tem dois concorrentes diretos, com propriedades e predisposições comparáveis. São eles:

- **Perl** - uma linguagem de script criada originalmente por Larry Wall;
- **Ruby** - uma linguagem de script criada por Yukihiro Matsumoto.
- O primeiro é mais tradicional e mais conservador do que o Python, e se assemelha a algumas das linguagens antigas derivadas da linguagem de programação clássica C.

- Em contrapartida, o último é mais inovador e mais cheio de novas ideias do que o Python. O próprio Python está em algum lugar entre essas duas criações.

6- Onde podemos ver Python em ação?

- É amplamente utilizado para implementar **serviços de Internet** complexos, como mecanismos de pesquisa, armazenamento em nuvem e ferramentas, mídias sociais e assim por diante. Sempre que você usa qualquer um desses serviços, você está muito próximo do Python, embora não saiba.
- Muitas **ferramentas de desenvolvimento** são implementadas em Python. Cada vez mais **aplicativos de uso diário** estão sendo escritos em Python. Muitos **cientistas** abandonaram ferramentas proprietárias caras e mudaram para o Python. Muitos **testadores** de projeto de TI começaram a usar o Python para realizar procedimentos de teste repetíveis. A lista é longa.

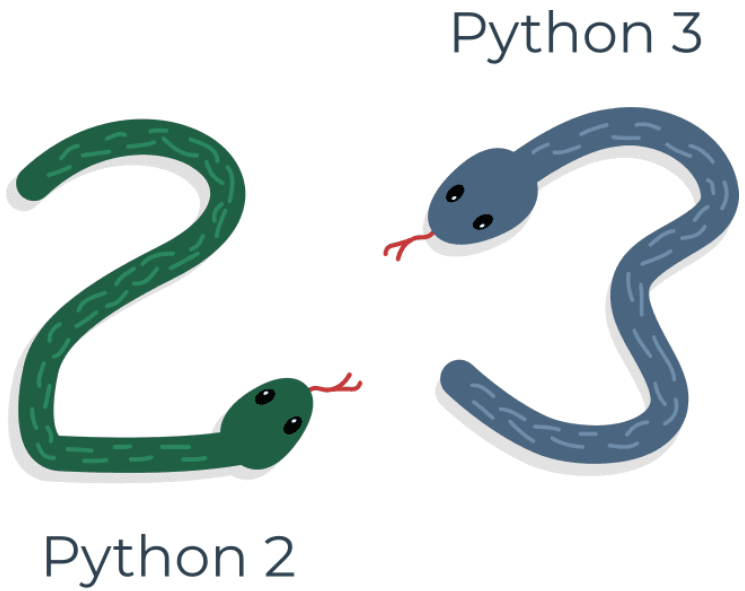
7- Por que não Python?

- Apesar da crescente popularidade do Python, ainda há alguns nichos nos quais o Python está **ausente** ou é raramente visto:

- **programação de baixo nível** (às vezes chamada de programação "próxima ao metal"): se você quiser implementar um driver ou mecanismo gráfico extremamente eficaz, não usará Python;
- **aplicativos para dispositivos móveis:** embora esse território ainda esteja esperando para ser conquistado pelo Python, provavelmente acontecerá algum dia.

8- Python 2 vs. Python 3

- Existem dois tipos principais de Python, chamados Python 2 e Python 3.
- Python 2 é uma versão mais antiga do Python original. Seu desenvolvimento foi intencionalmente interrompido, embora isso não signifique que não há atualizações para ele. Pelo contrário, as atualizações são emitidas regularmente, mas não pretendem modificar a linguagem de forma significativa. Eles preferem corrigir bugs e falhas de segurança recém-descobertos. O caminho de desenvolvimento do Python 2 já chegou a um impasse, mas o próprio Python 2 ainda está vivo.
- **Python 3 é a versão mais recente (ou para ser mais preciso, a atual) da linguagem. Ela está seguindo seu próprio caminho evolutivo, criando seus próprios padrões e hábitos.**
- Se você vai começar um novo projeto Python, **deve usar o Python 3, e esta é a versão do Python que será usada durante este curso**



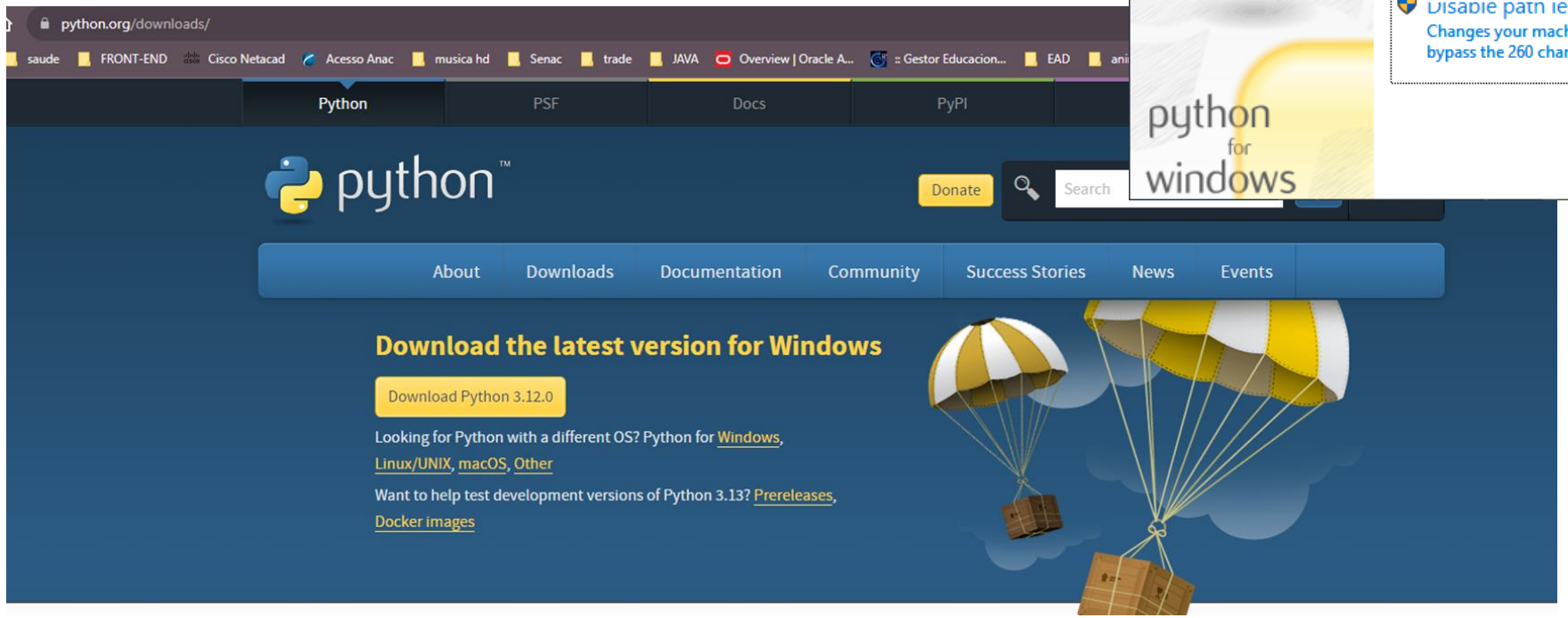
O Python 3

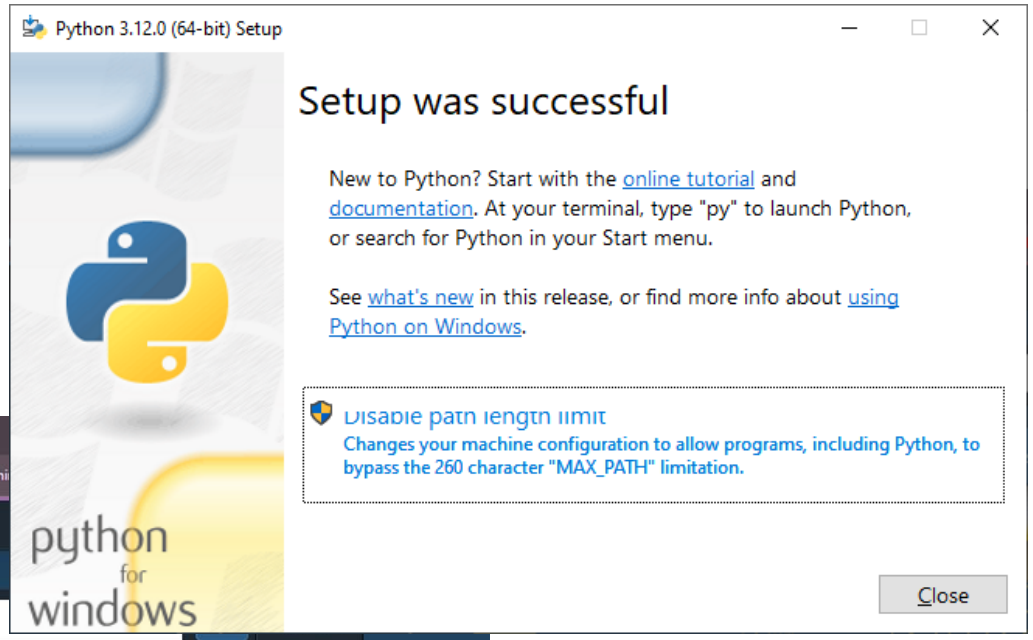
- O Python 3 não é apenas uma versão melhor do Python 2 - é uma linguagem completamente diferente, embora muito semelhante ao seu antecessor. Quando você os observa à distância, eles parecem ser os mesmos, mas quando você olha de perto, você percebe muitas diferenças.
- Se você estiver modificando uma solução Python antiga, é muito provável que ela tenha sido codificada em Python 2. Essa é a razão pela qual o Python 2 ainda está em uso. Há muitos aplicativos Python 2 para descartá-lo completamente.

Implementações de Python

- *Uma implementação* do Python se refere a "um programa ou ambiente, que oferece suporte para a execução de programas escritos na linguagem Python, conforme representado pela implementação de referência do CPython".
- A implementação *tradicional* do Python, chamada **CPython**, é a versão de referência da linguagem de computação Python de Guido van Rossum, e na maioria das vezes é chamada apenas de "Python". Quando você ouve o nome *CPython*, ele provavelmente é usado para diferenciá-lo de outras implementações alternativas não tradicionais.

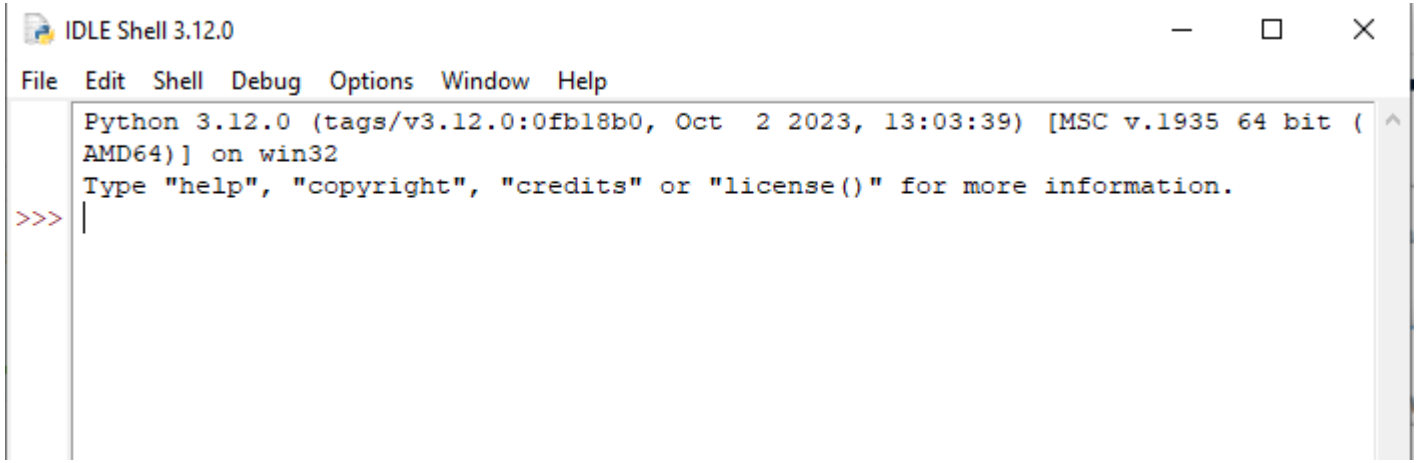
Download


python.org/downloads/
saude
FRONT-END
Cisco Netacad
Acesso Anac
musica hd
Senac
trade
JAVA
Overview | Oracle A...
:: Gestor Educacion...
EAD
Python
PSF
Docs
PyPI
python
Donate
Search
About
Downloads
Documentation
Community
Success Stories
News
Events
Download the latest version for Windows
Download Python 3.12.0
Looking for Python with a different OS? Python for Windows,
Linux/UNIX, macOS, Other
Want to help test development versions of Python 3.13? Prereleases,
Docker images

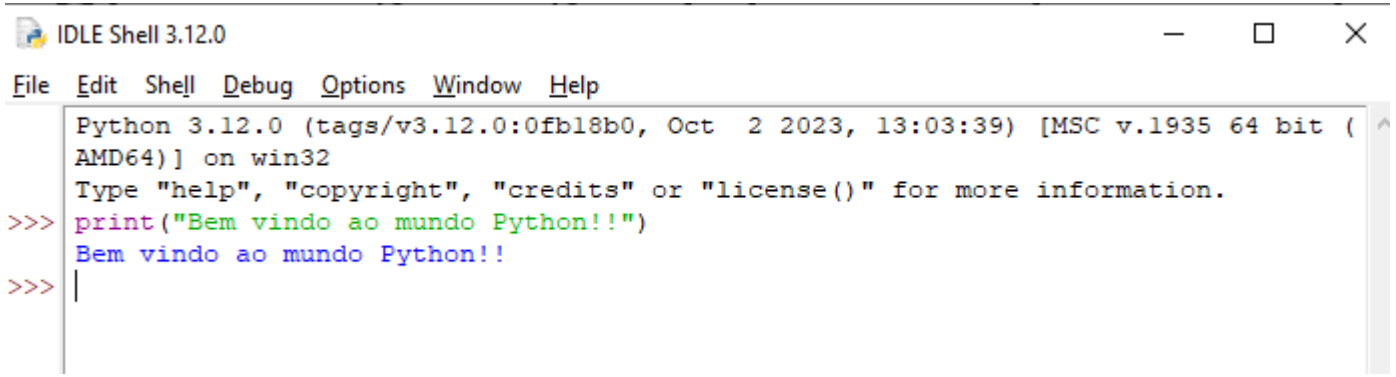

Python 3.12.0 (64-bit) Setup
Setup was successful
New to Python? Start with the online tutorial and
documentation. At your terminal, type "py" to launch Python,
or search for Python in your Start menu.
See what's new in this release, or find more info about using
Python on Windows.
Disable path length limit
Changes your machine configuration to allow programs, including Python, to
bypass the 260 character "MAX_PATH" limitation.
python
for
windows
Close

- Agora que você tem o Python 3 instalado, é hora de verificar se ele funciona e fazer o primeiro uso dele.
- Esse será um procedimento muito simples, mas deve ser suficiente para convencê-lo de que o ambiente Python é completo e funcional.
- Há muitas maneiras de utilizar o Python, especialmente se você for um desenvolvedor de Python.
- Para começar o trabalho, você precisa das seguintes ferramentas:
  - um **editor** que ajudará você a escrever o código (ele deve ter alguns recursos especiais, não disponíveis em ferramentas simples); este editor dedicado oferecerá mais do que o equipamento padrão do SO;
  - um **console** no qual é possível iniciar o código recém-escrito e interrompê-lo à força quando ele ficar fora de controle;
  - uma ferramenta chamada **debugger**, capaz de iniciar o código passo a passo, o que permitirá que você inspecione o código a cada momento de execução.
- Além de seus muitos componentes úteis, a instalação padrão do Python 3 contém um aplicativo muito simples, mas extremamente útil chamado IDLE.
- **IDLE** significa Integrated Development and Learning Environment (Desenvolvimento e ambiente de aprendizado integrados).

- Agora é a hora de escrever e executar o seu primeiro programa Python 3. Vai ser muito simples, por enquanto.
- A primeira etapa é criar um novo arquivo de origem e preenchê-lo com o código. Clique em *Arquivo* no menu IDLE e selecione *New file*.
- Como você pode ver, o IDLE abre uma nova janela para você. Você pode usá-lo para escrever e alterar seu código.
- Esta é a **janela do editor**. Seu único objetivo é ser um local de trabalho no qual o código-fonte é tratado. Não confunda a janela do editor com a janela do shell. Eles desempenham funções diferentes.



- Observação: não defina nenhuma extensão para o nome do arquivo que você usará. O Python precisa que seus arquivos tenham a extensão *.py*, então você deve confiar nos padrões da janela de diálogo. O uso da extensão padrão .py permite que o sistema operacional abra esses arquivos corretamente.
- Agora, coloque apenas uma linha na janela do editor recém-aberta e nomeada.



Testando

segundo.py - C:/Users/Clenio Emidio/Desktop/segundo.py (3.12.0)

File Edit Format Run Options Window Help

```
print("A aranha pequenininha\nsubiu a tromba d'água.")
print()
print("abaixo veio a chuva\ne lavou a aranha.")
```

- Pressione f5 ou
- Menu Run e Run Module

segundo.py - C:/Users/Clenio Emidio/Desktop/segundo.py (3.12.0)

File Edit Format Run Options Window Help

```
print("A aranha pequenininha\nsubiu a tromba d'água.")
print()
print("abaixo veio a chuva\ne lavou a aranha.")
print("_________")
print("A aranha pequenininha" , "subiu" , "a tromba d'água.")
print("Meu nome é", "Python.")
print("Monty Python.")
print("Meu", "nome", "é", "Monty", "Python.", sep="-")
print("2")
print(2)
print(10*2)
print ("ou")
print (10**2)
```

Alternativa de uso

- https://www.onlinegdb.com/online_python_compiler

Literais - os dados em si

- **Um literal são dados cujos valores são determinados pelo próprio literal.**
- print("2") é texto
- print(2) é número inteiro

Variáveis

É necessário declarar variáveis em Python?

Python é dinamicamente tipada, o que significa que não é necessário declarar o tipo da variável e nem fazer casting (alterar o tipo da variável), pois tudo isso é encarregado pelo interpretador.

Para criar(declarar) uma variável em Python , basta atribuir um valor a ela utilizando o operador de atribuição = .

```
print("Hello World\n FAC Senac")
valor = 1000
desconto = 100
valorFinal = valor-desconto
print(valorFinal)
```

```
print("Hello World\n FAC Senac")
valor = 1000
desconto = 100
valorFinal = valor-desconto
print(valorFinal)
print ("O valor com desconto é de:",valorFinal)
```

Tipos de variáveis em Python

- Inteiro (int)
- Ponto Flutuante ou Decimal (float)
- Tipo Complexo (complex)
- String (str)
- Boolean (bool)
- List (list)
- Tuple
- Dictionary (dic)

```
1  idade = 18
2  ano = 2002
3
4  print(type(idade))
5  print(type(ano))
```

Resultado

```
<class 'int'>
<class 'int'>
```

```
1  a = 5+2j
2  b = 20+6j
3
4  print(type(a))
5  print(type(b))
6  print(complex(2, 5))
```

Resultado

```
<class 'complex'>
<class 'complex'>
(2+5j)
```

Conversões

Em determinados cenários pode ser necessário mudar o tipo de uma variável e no Python isso é muito fácil, uma das vantagens de uma linguagem dinamicamente tipada.

Entre os erros mais comuns que acontecem principalmente com quem está iniciando sua jornada com a linguagem Python estão os famosos TypeError!

Sempre que o tipo TypeError aparecer em seu programa, preste atenção aos tipos dos dados e as operações que estão sendo executadas.

```
  File "c:\Users\Clenio Emidio\Desktop\python\ex1.py", line 6, in <module>
    print("A sub é ", n1-n2)
                      ~~^~~
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Resultado

```
<class 'int'>
<class 'float'>
18.0
```

```
1   # Antes da conversão
2   idade = 18
3   print(type(idade))
4
5   # Conversão do tipo
6   idade = float(idade)
7
8   # Depois da conversão
9   print(type(idade))
10  print(idade)
```

Exemplo conversões

ex1.py > ...

```python
a= 10
b= 20

n = input("Digte seu nome")
n1 = int(input("Digite o primeiro numero"))
n2 = int(input("Digite o segundo numero"))
n3= float(input("Digite um número com casas decimais"))
n4= 10
n5= str(n4)
print (n5)
print (n5+n1) #

```

A indentação nas estruturas condicionais no Python if-else

- Um aspecto interessante das estruturas condicionais no Python é o fato de que elas não envolvem chaves ou demarcadores de início do bloco de comandos.
- Isso é feito de forma natural, com a chamada indentação. Essa forma de codificar segue a filosofia de limpeza e minimalismo da linguagem.
- Por isso, depois de escrever o IF e a condição, o programador deve usar o tab para começar o comando com um recuo na próxima linha. Isso indica que aquele comando faz parte do IF.
- Caso o código não esteja indentado (um erro muito comum), o compilador vai entender que aquele código está fora do IF e deve ser executado somente quando o IF terminar.
- **Elif em Python**
  - O elif é uma estrutura intermediária dentro da seção if-else no python e deve vir como um complemento a ambos. Quando você já tem um IF e um ELSE, mas precisa de uma condição para especificar outra regra, pode usar o elif.

Condicional IF

ELSE

```
valor1 = 10
valor2 = 20

if valor1 > valor2:
    print('O valor1 é maior do que o valor2')

else:
    print('O valor2 é maior do que o valor1')
```

```
O valor2 é maior do que o valor1
```

ELIF

```
cor = "alguma cor"

if cor == 'verde':
    print('Acelerar')

elif cor =='amarelo':
    print('Atenção')

else:
    print('Parar')
```

```
Parar
```

```
print("Hello World\n FAC Senac")
valor = 1000
desconto = 100
valorFinal = valor-desconto
print(valorFinal)
print ("O valor com desconto é de:",valorFinal)
if valorFinal >800:
    print("Parabens! Você ganhou um bônus!")
else:
        print("Não ganhou o Bônus")
```

Operadores Lógicos

```
variavel =

if nome:
    print('A variável nome não é vazia')

if nome == #Igual

if nome != #Diferente

if nome >= #Maior ou Igual

if nome <= #Menor ou Igual

if nome > #Maior

if nome < #Menor

if a in nome #Está dentro

if a in lista

if is True #É verdadeiro

if True
```

IF COM FOR

```
lista = ['Heitor','Daniel','Lira']

for item in lista:
    if item == 'Daniel':
        print( 'É Daniel')
    else:
        print( 'Não é Daniel')
```

```
Não é Daniel
É Daniel
Não é Daniel
```

Criando uma Decisão baseado em ordens de Manutenção

```
ordens = ['20010102', '3030304050', '40005004305','20067802', '3030356750', '40005005675', '3030304050',
          '24505004305', '30067802', '4030356750']

for ordem in ordens:
    if ordem[0]=='2':
        print(f'Ordem {ordem} - Manutenção Preventiva')
    elif ordem[0]=='3':
        print(f'Ordem {ordem} - Manutenção Corretiva')
    else:
        print(f'Ordem {ordem} - Manutenção Preditiva')
```

```
Ordem 20010102 - Manutenção Preventiva
Ordem 3030304050 - Manutenção Corretiva
Ordem 40005004305 - Manutenção Preditiva
Ordem 20067802 - Manutenção Preventiva
Ordem 3030356750 - Manutenção Corretiva
Ordem 40005005675 - Manutenção Preditiva
Ordem 3030304050 - Manutenção Corretiva
Ordem 24505004305 - Manutenção Preventiva
Ordem 30067802 - Manutenção Corretiva
Ordem 4030356750 - Manutenção Preditiva
```

Listas

- Uma Lista (list) em Python, nada mais é que uma coleção ordenada de valores, separados por vírgula e dentro de colchetes [].
- Elas são utilizadas para armazenar diversos itens em uma única variável. Entender este conteúdo é de extrema importância para dominar a linguagem por completo!

```
listap= ['iphone 12','notebook dell ips','apple watch',2023]
for item in listap:
    print(item)
```

```
iphone 12
notebook dell ips
apple watch
2023


...Program finished with exit code 0
Press ENTER to exit console.
```

```
for i in range(10):
    print("EStude!")
```

Entrada de dados com a Função input

- Esse é um artifício muito comum em programação, quando precisamos que o usuário passe ao programa algum tipo de dado.
- Em Python, fazemos isso utilizando a função input(), que é literalmente 'entrada' em inglês.
- A função input() recebe como parâmetro uma string que será mostrada como auxílio ao usuário, geralmente o informando que tipo de dado o programa está aguardando receber.

```
nome = input("Qual seu nome?")
print ("O nome digitado é:",nome)
```

```
nome = input("Qual seu nome? ")
ano = input("qual ano de nascimento? ")
idade = 2023 - int(ano)
print (nome, " tem ",idade," anos")
print (f'{nome} tem {idade} anos.')
```

Lista simples

- Permite armazenar uma coleção de itens em uma única variável.
  - permite membros duplicados
  - possui índice (0,1,2,3...)

```
5   lista = ['Clenio', 'Andre', "Andre",'Ana Paula','Marilza']
6   print (lista)
7
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emidio\Desktop\python\exemplolistas.py"
['Clenio', 'Andre', 'Andre', 'Ana Paula', 'Marilza']
PS C:\Users\Clenio Emidio\Desktop\python>
```

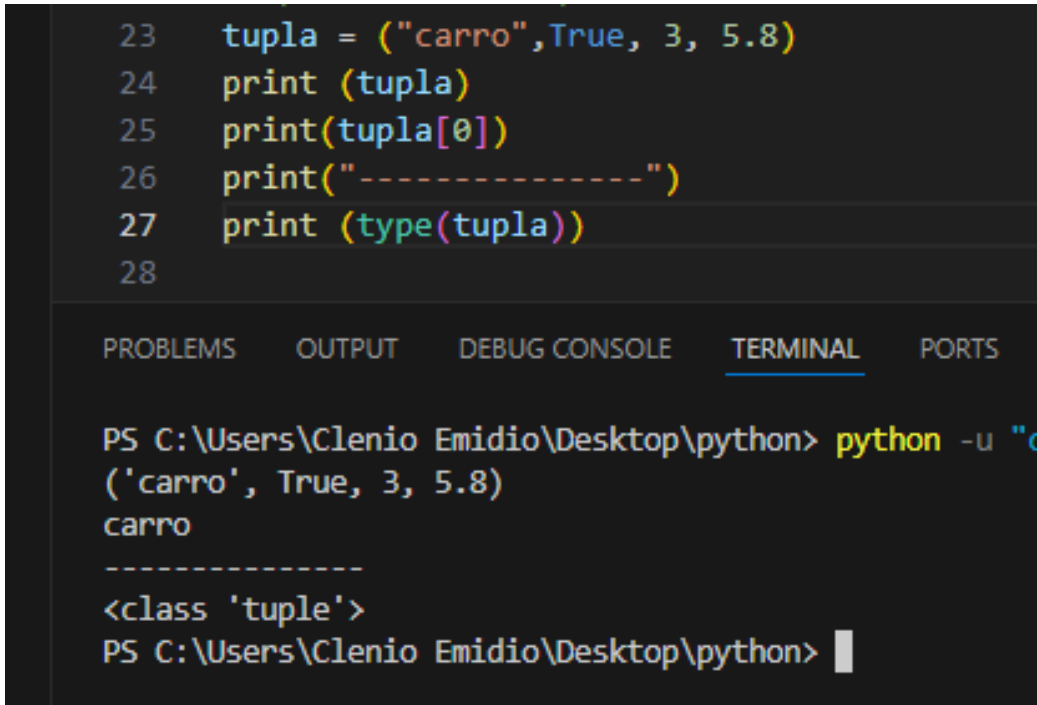
  - Para imprimir uma posição bastaríamos colocar o número da posição:

```
5   lista = ['Clenio', 'Andre', "Andre",'Ana Paula','Marilza']
6   print (lista)
7   print(lista[3])
8
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emi
['Clenio', 'Andre', 'Andre', 'Ana Paula', 'Marilza']
Ana Paula
PS C:\Users\Clenio Emidio\Desktop\python>
```

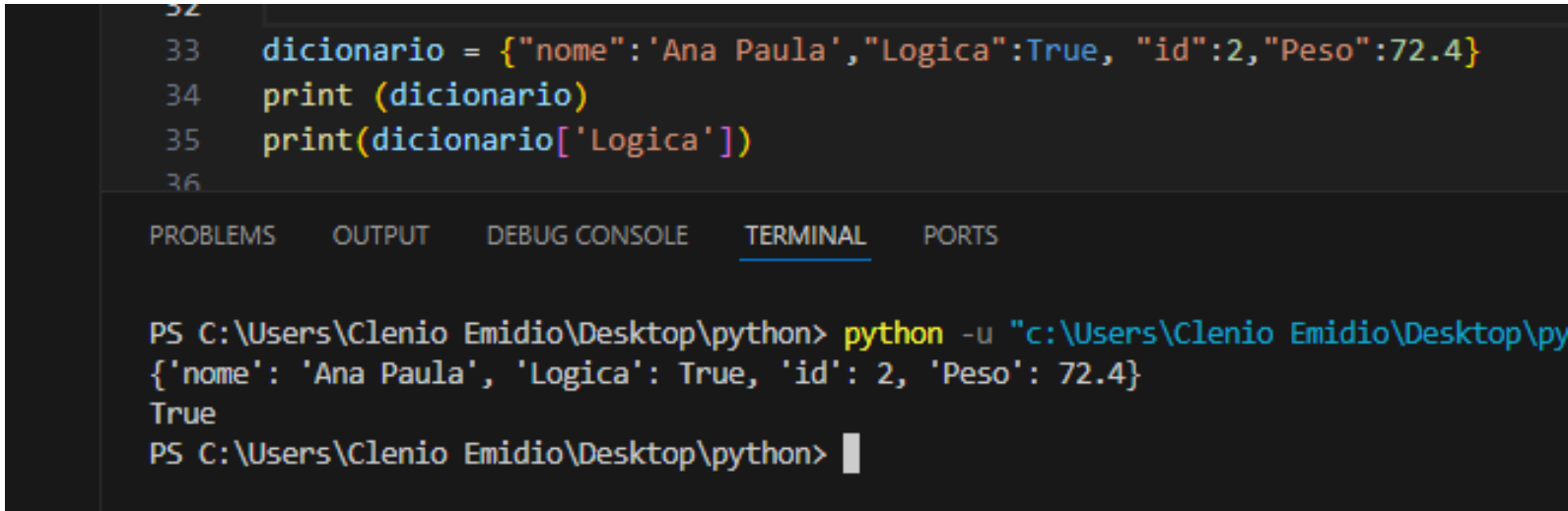
Listas

- **Tupla** - é uma sequência imutável de valores de qualquer tipo. Permite membros duplicados
- Possui índices (0,1,2,3...)
- Caso quiséssemos imprimir a posição “0” basta digitar print(tupla[0])conforme linha 25 na imagem ao lado
- Na linha 27 na imagem permite imprimir o tipo
- De lista <class ‘tuple’>



Listas

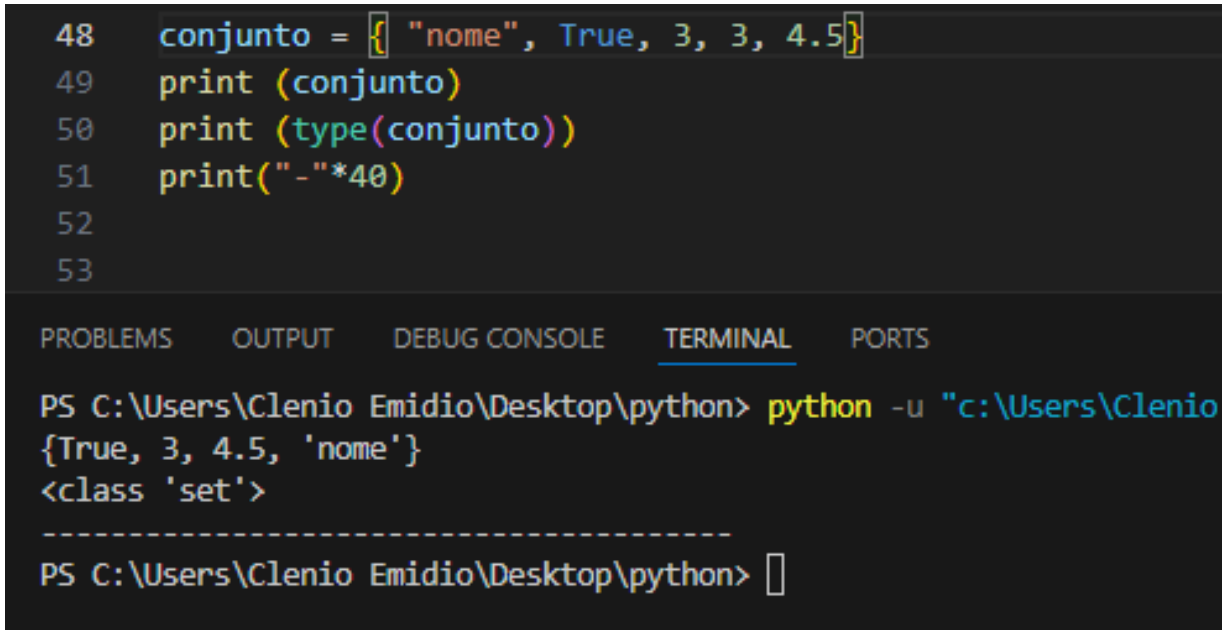
- **Dicionário**: - é uma coleção ordenada com elementos "**chave-valor**" que permite representar melhor o mundo real. Ambos são separados por “:”
- nenhum membro duplicado, possui índice!
- Porém para imprimir a posição de um índice, identificamos a chave. No exemplo abaixo (linha 35) queremos imprimir o valor da chave ‘Lógica’ e foi impresso na tela ‘True’.



Listas

- **Conjunto** - O mesmo que dicionário, porém "sem" a definição de chave e valor
- É uma coleção não ordenada e 'não indexada'. Nenhum membro duplicado. Ou seja, caso tenha algum valor que se repita, ele ignora, conforme mostra na imagem.
- Ele também altera a ordem de exibição

Dos valores.



Exemplos para adicionar e remover itens lista

```python
# listas

# Lista de frutas
frutas = ['maçã', 'banana', 'cereja']

# Adicionar um item ao final da lista
frutas.append('uva')

print(frutas)  # ['maçã', 'laranja', 'cereja', 'uva']

# Remover um item específico
frutas.remove('laranja')

print(frutas)  # ['maçã', 'cereja', 'uva']
```

For e While – Estruturas condicionais

Laço de Repetição ou loop

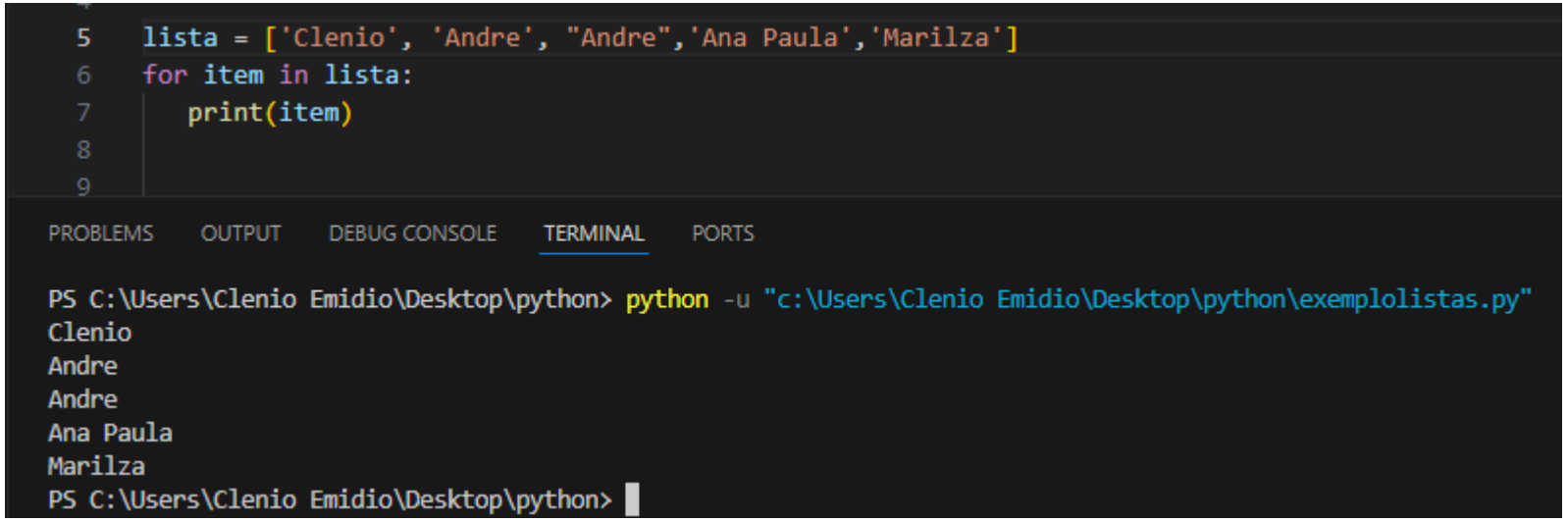
- Em Python, os loops são codificados por meio dos comandos for e while.
- O '**For**' permite percorrer os itens de uma coleção e, para cada um deles, executar um bloco de código ou um conjunto de instruções em cada item.., é utilizado para percorrer ou iterar sobre uma sequência de dados (seja esse uma lista, uma tupla, uma string)
- Já o **while**, executa um conjunto de instruções várias vezes enquanto uma condição é atendida.

```
1 | nomes = ['Pedro', 'João', 'Leticia']
2 | for n in nomes:
3 |     print(n)
```

```
1 | contador = 0
2 | while contador < 5:
3 |     print(contador)
4 |     contador = contador + 1
```

Listas com laço de Repetição ‘For’

- Permite percorrer a lista e imprime cada valor abaixo um do outro.
  - A indentação é primordial, observe que o print está um pouco à direita
  - Tudo o que estiver indentado faz parte do laço de repetição.



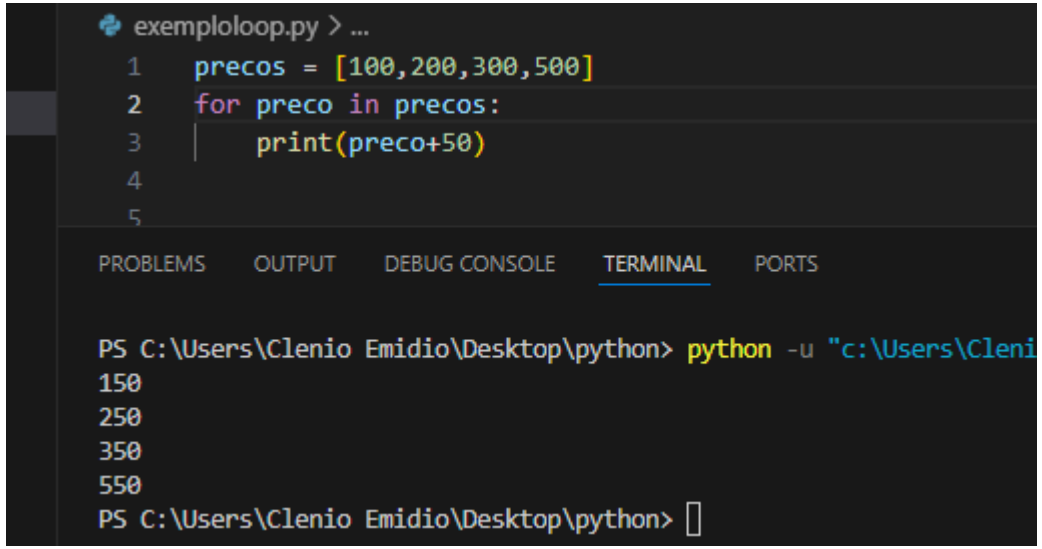
Laço de repetição For com outros argumentos

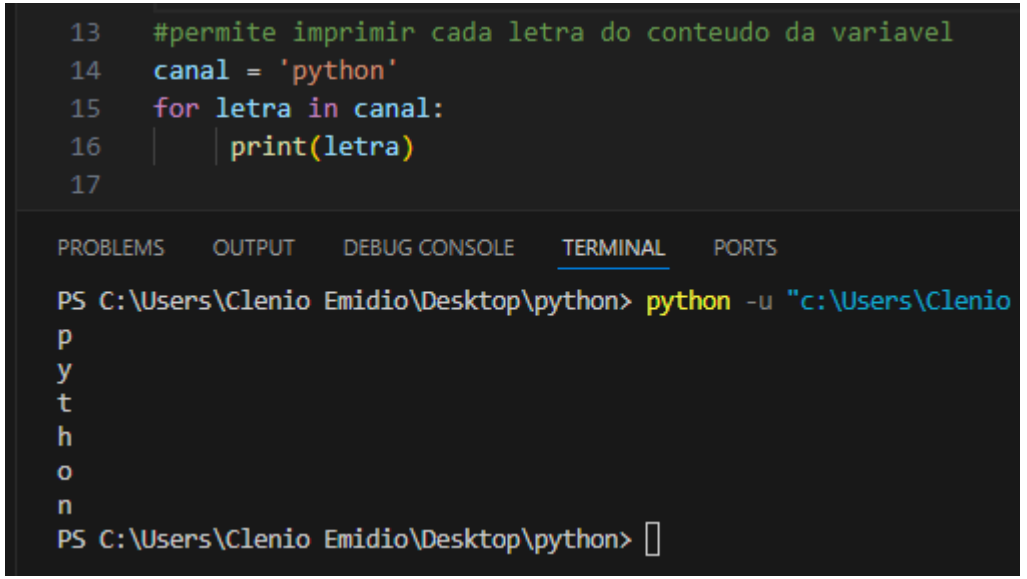
```
5   lista = ['Clenio', 'Andre', "Andre",'Ana Paula','Marilza']
6   for item in lista:
7       print(item)
8   print (lista[1])
9   print("*"*40)
10  print(type(lista))
```

```
ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

S C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emidio\Desktop\python\exemplolistas.py"
lenio
ndre
ndre
na Paula
arilza
ndre
****************************************
class 'list'>
S C:\Users\Clenio Emidio\Desktop\python>
```

For com listas

- No exemplo abaixo será somado o valor 50 a cada preço.
  - Pois o print está dentro do laço de repetição.
  - No exemplo a direita podemos imprimir caracteres, embora não seja uma lista





While

- O while é uma estrutura de repetição utilizada quando queremos que determinado bloco de código seja executado ENQUANTO (do inglês while) determinada condição for satisfeita.

- Em outras palavras: só saia da estrutura de repetição quando a condição não for mais satisfeita

```
while <condição>:
    # Bloco aser executado
```

```
16  #loop while
17
18  i = 1
19  while i < 10:
20      print(i)
21      i=i+1    # ou i+= 1
22      print(i) # para otimir o numero 10 basta colocar print(i-1)
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emidio\Desktop\python\exemploloop.py"
1
2
2
3
3
4
4
5
5
6
6
7
7
8
8
9
9
10
PS C:\Users\Clenio Emidio\Desktop\python>
```

While com intervalo

- Para isso utilizamos o ‘range’

  E passamos o intervalo onde inicia e termina entre parênteses.

  No exemplo abaixo imprimimos os números pares

```
36  #testando numeros pares
37  for i in range(1, 10) :
38      if i % 2 == 0 :
39          print(i)
40
41  print("*"*20)
42
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PO

PS C:\Users\Clenio Emidio\Desktop\python> python
2
4
6
8
********************
PS C:\Users\Clenio Emidio\Desktop\python>
```

```
28  #Itervalo
29  for T in range(1, 10) :
30      print(T)
31
32  print("-"*20)
33
34
35  '''
36
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\
1
2
3
4
5
6
7
8
9
--------------------
PS C:\Users\Clenio Emidio\Desktop\python>
```

Exemplo com While

```python
#while

senha_correta = 'python123'
tentativa = ''

print('Bem-vindo! Por favor, digite a sua senha.')

while tentativa != senha_correta:
    tentativa = input('Digite a senha: ')
    if tentativa != senha_correta:
        print('Senha incorreta. Tente novamente.')

print('Senha correta! Acesso permitido.')
```

For com contador

```
58    c= 0  #contador
59    for i2 in range(1, 10) :
60        c= c+1
61        print (c)
62
```

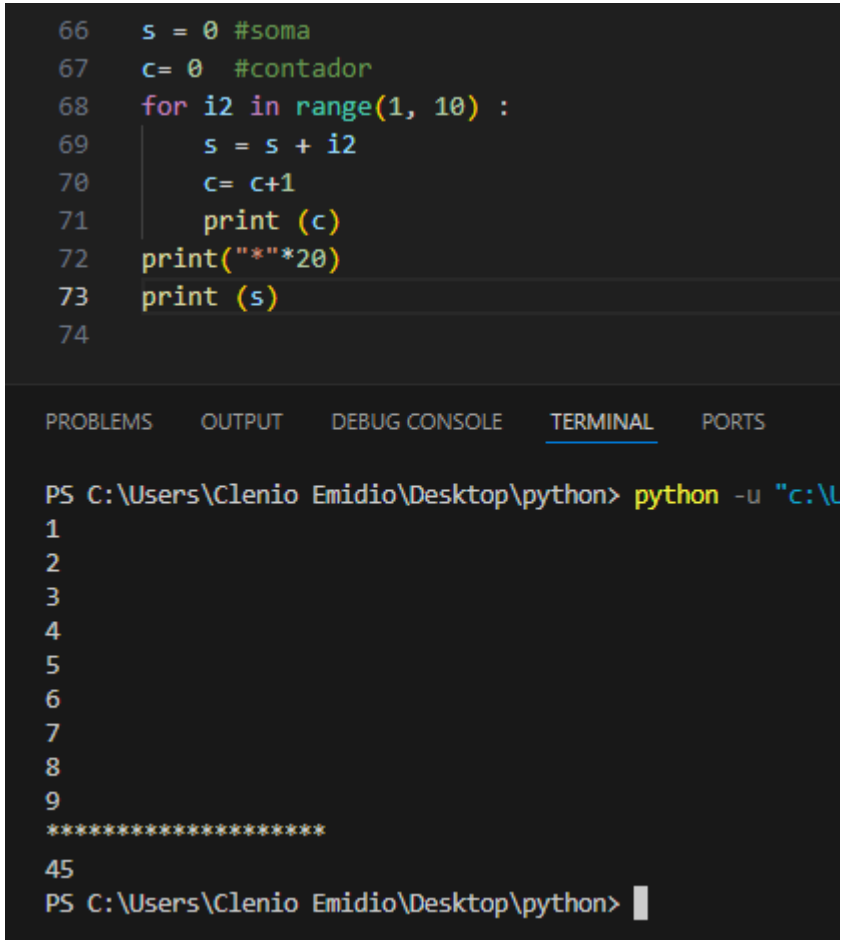
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c
1
2
3
4
5
6
7
8
9
PS C:\Users\Clenio Emidio\Desktop\python>
```

- Observe o mesmo código mas sem indentação do Print no laço de repetição.
- Muda tudo!
- Mostra apenas o último número na última passagem no laço*(conforme imagem a direita).*

```
58    c= 0  #contador
59    for i2 in range(1, 10) :
60        c= c+1
61    print (c)
62
63
```

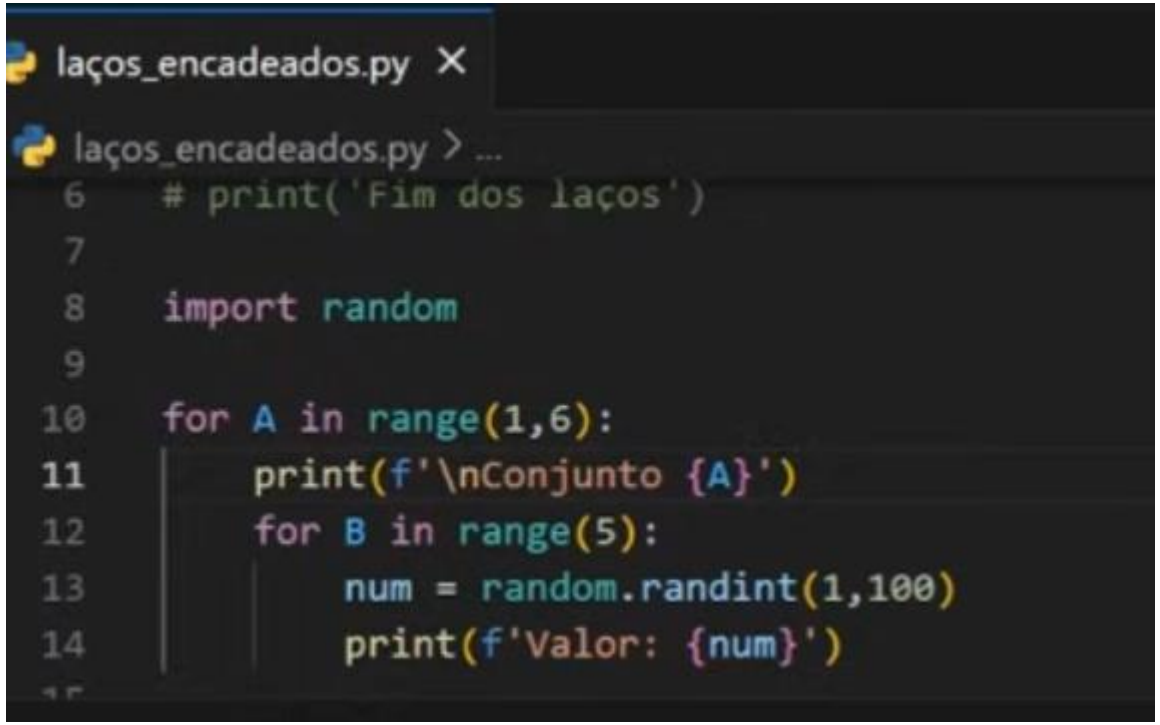
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  POR

PS C:\Users\Clenio Emidio\Desktop\python> python
9
PS C:\Users\Clenio Emidio\Desktop\python>
```

Contador e acumulador(soma)



- Variável de soma (s)
  - A cada passagem vai somando cada número, 1+2+3...+9 = 45
- Variável contador(c)
  - A cada passagem é somado +1,
  - Resultado na lista de 1 a 9.
- print("*"*20) serve apenas para repetir o “*” 20 vezes e separar os valores.

Laços Encadeados



F-Strings em Python

- É basicamente uma ferramenta que temos para formatação de textos em Python.
- As f-strings vão servir para que você consiga colocar uma variável dentro de um texto, e isso é feito utilizando a letra “f” antes do texto e colocando a sua variável dentro de {} chaves.
- Isso é muito útil para que você não tenha que ficar concatenando o seu texto com as variáveis e tenha que fatiar seu texto várias vezes por conta disso.
- https://www.hashtagtreinamentos.com/f-strings-em-python?gad_source=1&gclid=Cj0KCQjwwuG1BhCnARIsAFWBUC11A82cZmad6MVwoOAiSz1g5g1Vq0PdEIlfEW5ILCuQP5_knjk_kN0aAk9LEALw_wcB

```
20    nome = input("Qual seu nome?")
21
22    print (f"Olá, {nome} seja bem vindo(a)!!")
23    print ("Olá,",nome, "seja bem vindo(a)!")
```

Formatando dados com casa decimal/moeda

fatura.py > ...

```python
#formatando com casa decimal
faturamento = 2000
print (f"Faturamento: {faturamento:.2f}")
#formato moeda
print (f"Faturamento: R$ {faturamento:,.2f}")

```

```
p\python\fatura.py"
Faturamento: 2000.00
Faturamento: R$ 2,000.00
```

Bibliografia

- https://skillsforall.com/
- https://www.hashtagtreinamentos.com/estruturas-condicionais-no-python?gad=1&gclid=CjwKCAjw-eKpBhAbEiwAqFL0mmxCOLCP7LH5Bw0bbeF1OTHwiDnXQjDhkMtTfJMJQ-tR_r2Fv18ZmBoCOl8QAvD_BwE
- https://pythonacademy.com.br/blog/input-e-print-entrada-e-saida-de-dados-no-python
- https://www.becas-santander.com/pt_br/blog/python-para-que-serve.html
- https://didatica.tech/primeiros-passos-no-python/

- https://community.revelo.com.br/lacos-de-repeticao-em-python/
- https://pythonacademy.com.br/blog/estruturas-de-repeticao
- https://gabriel-schade-cardoso.gitbook.io/python-aprendendo-a-programar/chapter6