Lógica de Programação

Introdução ao Python

Checando o Aprendizado

- O que são variáveis?
- Quais são os tipos de dados que o Python possui?

- É possível fazer operações matemáticas com variáveis?
- Os dados booleanos só representam 2 valores quais são eles ?

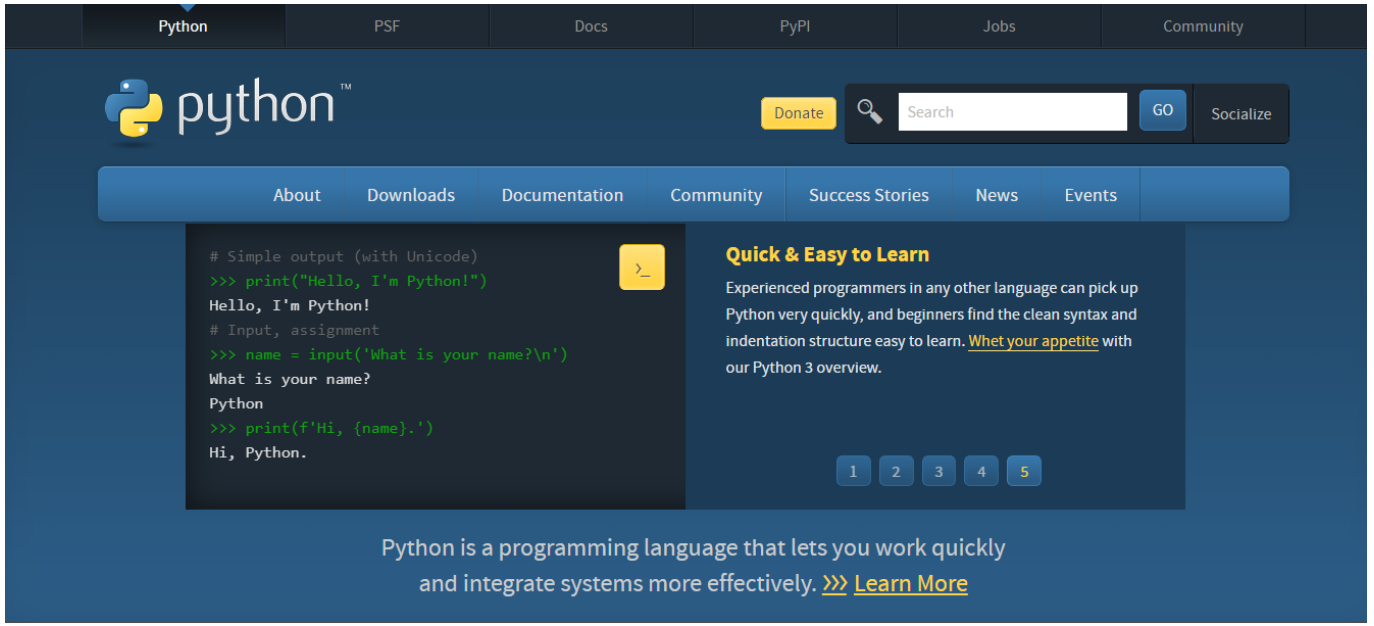
Setup

Agora que já vimos alguns conceitos vamos preparar nosso setup para conseguirmos aproveitar melhor a nossa jornada juntos.

Baixando o Python...

Para iniciarmos precisaremos ter o python instalado, par baixar é bem simples, siga o passo a passo:

Acesse: python.org



Baixando o Python...

Clique na aba Downloads

Baixando o Python...

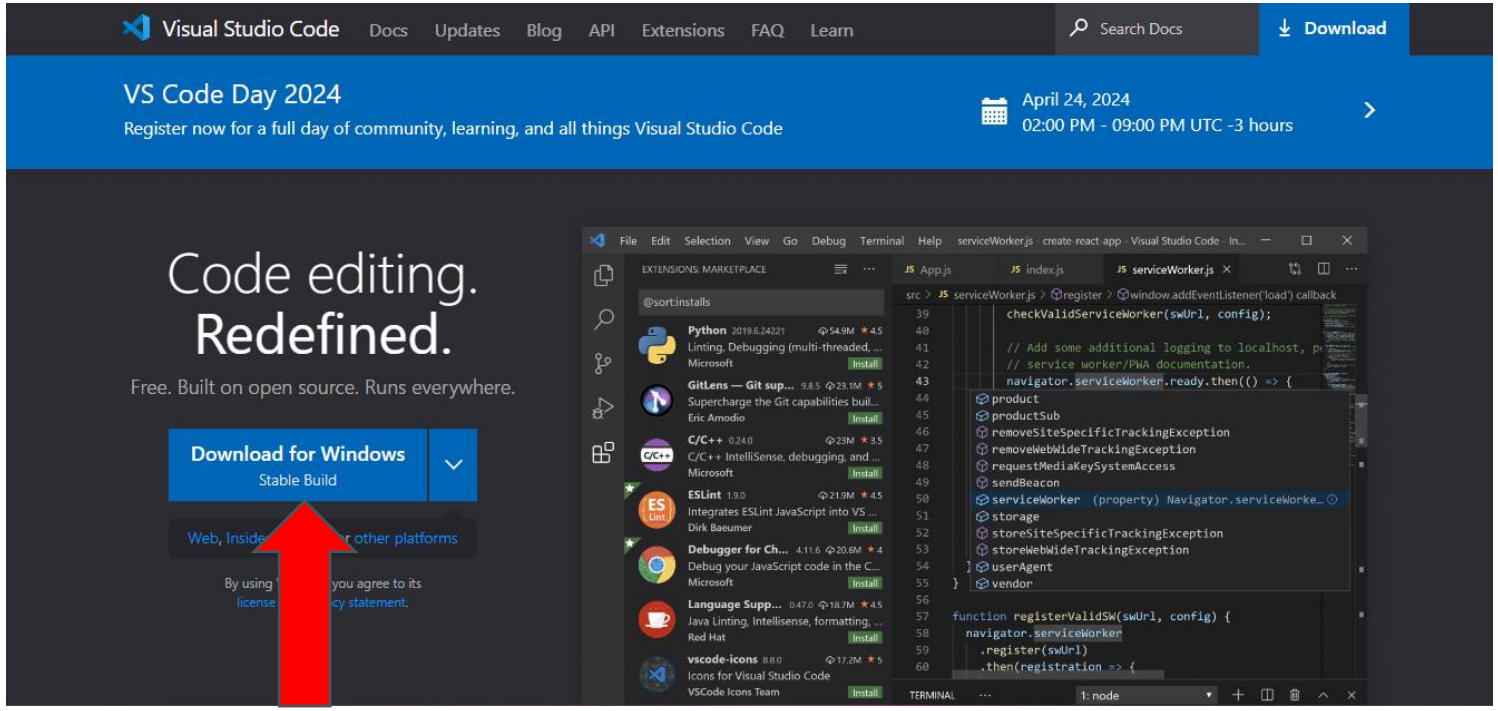
Clique no botão “Download Python 3.12.3”

Baixando o VS Code

Iremos utilizar o Visual Studio Code como editor de código

Para baixar acesse o link: code.visualstudio.com

E Clique em:
Download for Windows



Setup Concluído

Agora que nosso setup está configurado, daqui pra frente utilizaremos apenas ele para executar nossos códigos.

Trabalhando com operadores matemáticos

Para testarmos suas habilidades e prosseguirmos da melhor maneira possível vamos executar alguns códigos.

1) Faça um programa para **somar** 2 números e mostre o resultado no terminal.

2) Faça um programa para **multiplicar** 2 números e mostre o resultado no terminal.

3) Faça um programa que eleve um número a uma **potência** mostre o resultado no terminal.

Trabalhando com operadores matemáticos

4) Faça um programa que divida dois números e mostre o resultado no terminal.

5) Faça um programa que mostre o resultado do resto de uma divisão entre dois número e mostre o resultado no terminal.

Trabalhando com operadores lógicos

1) Faça um programa que verifica se um número é **maior** que o outro e mostre o resultado no terminal.

2) Faça um programa que verifica se um número é **menor** que outro e mostre o resultado no terminal.

3) Faça um programa que verifica se um número é **menor ou igual** ao outro e mostre o resultado no terminal.

4) Faça um programa que verifica se um número é **maior ou igual** ao outro e mostre o resultado no terminal

Trabalhando com operadores lógicos

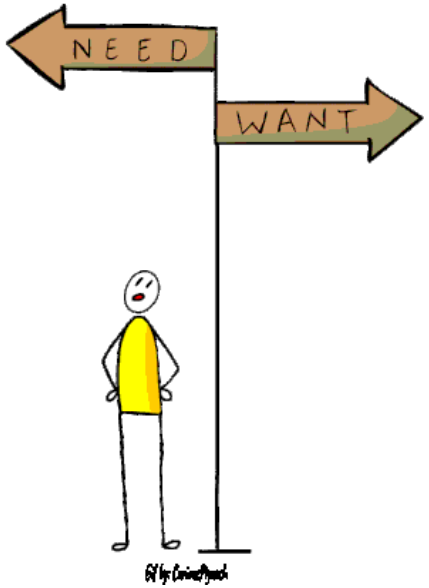
6) Faça um programa que verifica se um número é **exatamente igual** ao outro e mostre o resultado no terminal.

7) Faça uma lista e mostre ela no terminal

Estruturas de Controle em Python

As estruturas de **controle** são os tijolos fundamentais para construir a **lógica** dos seus programas em Python. Elas permitem **tomar decisões**, **executar tarefas repetidamente** e **estruturar o código de forma organizada e eficiente.** Nesta aula, vamos mergulhar no universo das estruturas de controle em Python, explorando seus diversos recursos e funcionalidades com exemplos práticos e exercícios desafiadores.

Dominando as Decisões com if e else



- A estrutura if é a sua chave para tomar decisões inteligentes no programa. Ela funciona como um juiz imparcial, avaliando uma condição e determinando qual caminho o código deve seguir. Se a condição for verdadeira, o bloco de código após o if será executado, revelando um resultado ou realizando uma ação específica. Caso contrário, o bloco de código após o else entra em cena, lidando com a situação alternativa.

Dominando as Decisões com if e else

- **Exemplo:** Imagine um programa que verifica se uma pessoa é maior de idade. A estrutura if seria perfeita para essa tarefa:

```
idade = 18

if idade ≥ 18:
    print("Parabéns! Você já é maior de idade!")
else:
    print("Calma, sua hora ainda vai chegar! Faltam", 18 - idade, "anos para você ser maior.")
```

Operadores Lógicos: Refinando suas Decisões

- Os operadores lógicos (and, or e not) são seus aliados na construção de condições mais complexas. Imagine que você precisa verificar se a pessoa é maior de idade e possui carteira de motorista para dirigir. Com os operadores lógicos, você consegue expressar essa condição com clareza:

Operadores Lógicos: Refinando suas Decisões

```python
idade = 17
tem_carteira = True

if idade ⩾ 18 and tem_carteira:
    print("Tudo certo! Você pode dirigir!")
else:
    if idade ⩾ 18:
        print("Você tem idade suficiente para dirigir, mas precisa tirar sua carteira antes.")
    else:
        print("Você ainda não tem idade para dirigir e nem carteira. Calma, tudo tem seu tempo!")

```

elif: Mais Opções para suas Decisões

- E se você precisar avaliar mais de duas possibilidades? O elif surge como um herói para expandir suas opções de decisão. Imagine um programa que classifica alunos de acordo com suas notas:

```
nota = 8.5

if nota ≥ 9:
    print("Parabéns! Você tirou nota excelente!")
elif nota ≥ 7.5:
    print("Muito bom! Você está no caminho certo!")
elif nota ≥ 6:
    print("Continue se esforçando! Você está quase lá!")
else:
    print("Não desanime! Estude mais e você conseguirá melhores notas.")
```

Repetições com for: Automatizando Tarefas

Cansado de digitar o mesmo código várias vezes? O for é a solução para automatizar tarefas repetitivas e economizar seu tempo. Imagine que você precisa imprimir todos os números de 1 a 100. Com o for, você faz isso em um piscar de olhos:

```
for numero in range(1, 101):
    print(numero)
# O range() gera a sequência de 1 a 100

```

range(): Gerando Sequências Personalizadas

- O poder do range() não se limita a sequências numéricas simples. Você também pode especificar o passo da contagem, definir um limite final diferente ou até mesmo gerar sequências em sentido inverso!

range(): Gerando Sequências Personalizadas

- **Exemplo:** Imprimir números pares de 2 a 20:

```
for numero in range(2, 21, 2):
    # Começa em 2, vai até 20 (não inclusive) e pula de 2 em 2
    print(numero)
```

break e continue: Controlando o Fluxo da Repetição

- Nem sempre você precisa executar o loop for até o fim. O break permite interromper o loop a qualquer momento, enquanto o continue faz com que o loop ignore a iteração atual e siga para a próxima.

break e continue: Controlando o Fluxo da Repetição

- **Exemplo:** Imprimir números ímpares de 1 a 10, pulando o número 5:

```python
for numero in range(1, 11):
    if numero == 5:
        continue  # Pula para a próxima iteração se o número for 5
    if numero % 2 != 0:  # Verifica se o número é ímpar
        # Adicione aqui o que você deseja que aconteça quando o número for ímpar
        print(f"O número {numero} é ímpar.")
```

Por hoje é só...

- Hoje vimos muita coisa, vamos descansar que amanhã tem mais!