Princípios do Python

Florianópolis – SC

Professor: Moroni Fernandes

Instalação

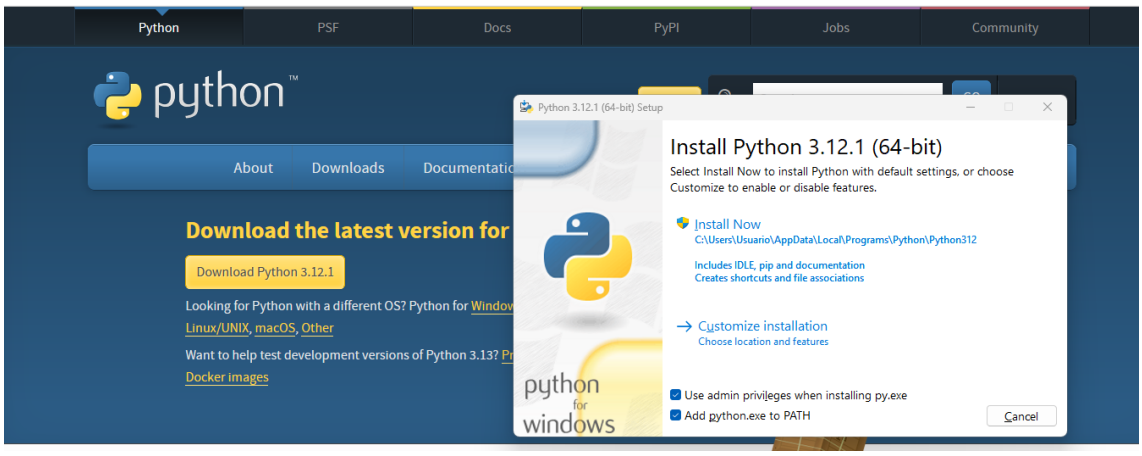
Antes de aprendermos os princípios básicos sobre Python, precisamos primeiro saber como instalá-lo, já que ele não vem instalado por padrão nos sistemas operacionais Windows. Caso sua máquina seja Linux, praticamente todas as distribuições já vêm com Python e IDLE instalados.

Em nossos estudos, estaremos utilizando uma máquina com o sistema operacional Windows!

Link de acesso: https://www.python.org/downloads/

Após fazer o download do arquivo, a instalação é bem simples. Você deve marcar a opção "**Add Python.exe to PATH**", como na imagem abaixo, e depois escolher a opção "**Install Now**", para que o Python seja instalado em sua máquina.

Figura 1 – Instalação Python



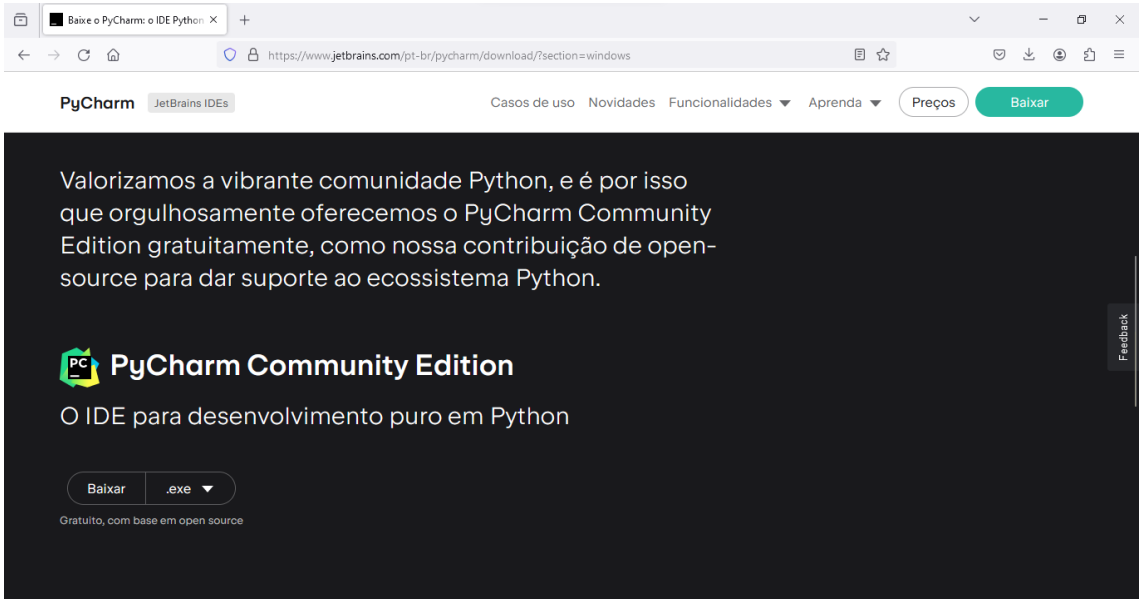
Acaba por aqui? Ainda não! Ainda temos que instalar o PyCharm, que basicamente é um ambiente de desenvolvimento integrado com interface gráfica (IDE). Qual a função dele? Basicamente, é através do PyCharm que vamos conseguir desenvolver nossos códigos em Python.

Link de acesso:

https://www.jetbrains.com/pt-br/pycharm/download/?section=windows

Acessando o link acima, teremos duas opções: "**Professional**" e "**Community**". Você vai escolher a segunda opção e clicar em "Baixar".

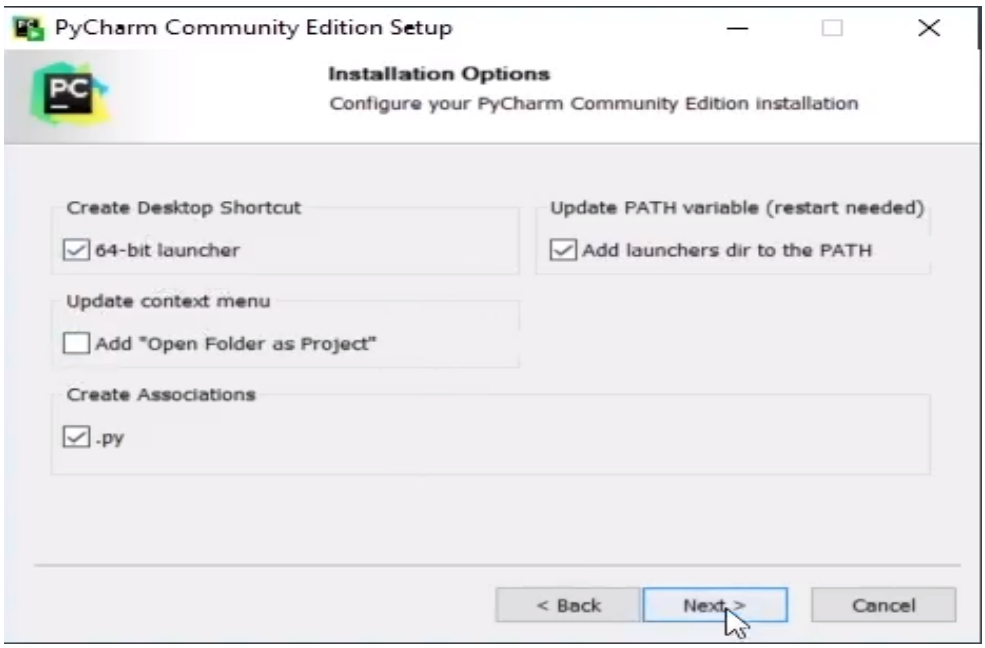
Figura 2 – Download Pycharm



O próximo passo é a instalação:

Comece selecionando a opção "**Next**" na primeira tela e, em seguida, "**Next**" novamente na segunda tela. “Na terceira tela, marque as opções: “**64-bit launcher**”, “**Add launchers dir to the PATH**” e “**.py**”. Na quarta tela, marque "**Next**" e, por último, clique em "**Install**".

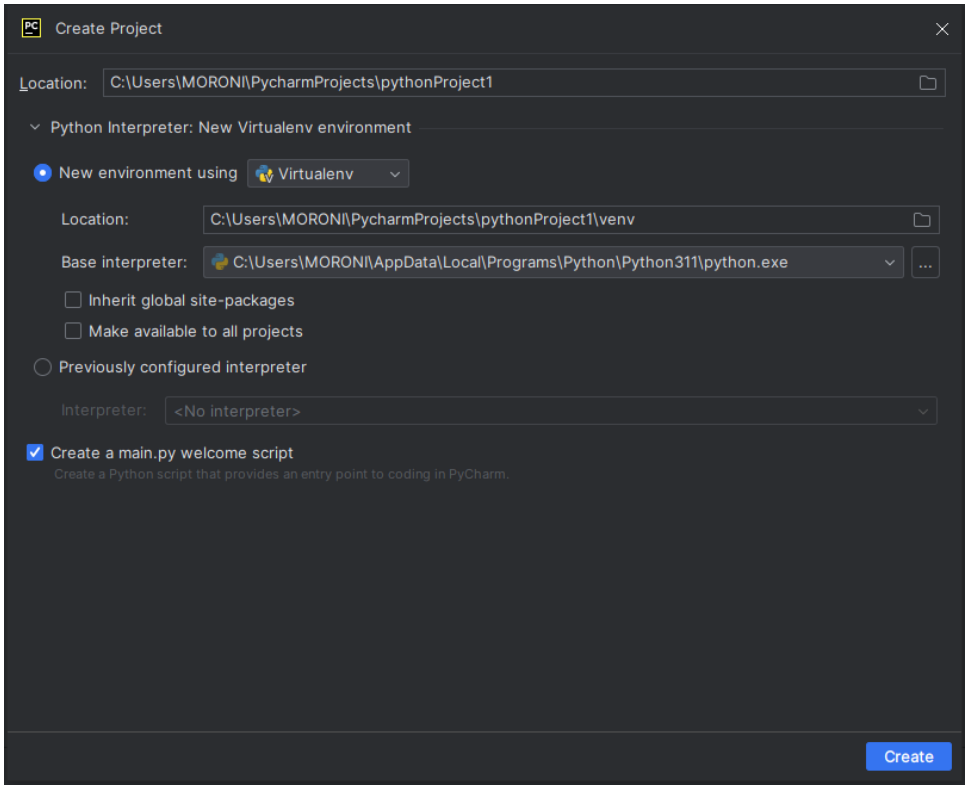
Figura 3 – Instalação Pycharm



Agora, com o PyCharm instalado, vamos abri-lo. No início, entre na opção "New Project" para podermos iniciar nosso projeto. Nessa tela, escolha o local onde os arquivos serão salvos. As demais opções podem ser deixadas como

estão na imagem abaixo. E, finalmente, podemos começar nosso aprendizado sobre Python!

Figura 4 – Criando projeto



Na sequência, teremos nossa tela inicial dentro do PyCharm, já na pasta que escolhemos:

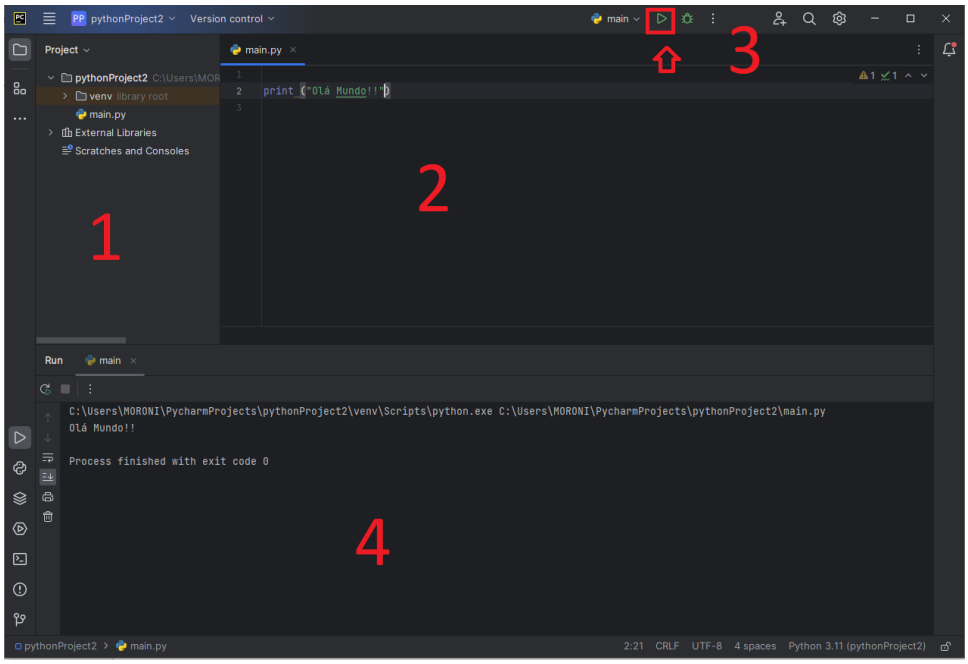
1. Do lado esquerdo da tela, temos as pastas e o arquivo "**main.py**", onde começaremos nosso projeto.

2. No centro da tela, temos a aba onde iremos digitar nossos códigos.

3. Para executar o código, aperte o símbolo no topo da tela, mais à direita, que se parece com um triângulo de lado. Você também pode clicar com o botão direito na aba do código e selecionar "**Run (Nome do projeto)**".

4. O resultado da execução do código sempre aparecerá na aba na parte inferior da tela, como mostrado na imagem abaixo.

Figura 5 – Conhecendo Pycharm



Para criar um novo projeto, vá até a área de pastas, clique com o botão direito e selecione "**New** -> **Python File**", depois nomeie seu novo projeto.

Algoritmos

Vamos começar entendendo o que são algoritmos. Basicamente, algoritmos são uma sequência de ações ou passos para completar algum objetivo.

Exemplo:

- Pegue o pote de achocolatado
- Com uma colher, coloque duas porções de achocolatado no copo
- Adicione o leite até 2/3 do copo
- Misture até não ver mais o pó do achocolatado. Pronto, seu achocolatado está feito!

Existem algumas formas de representar algoritmos, como Descrição Narrativa, Pseudocódigo e Fluxograma. No entanto, vamos focar em apenas uma delas: o Pseudocódigo.

Pseudocódigo: Possui regras definidas e uma linguagem genérica, mas com um português estruturado. Ele é o mais próximo de um programa computacional sem ser, de fato, uma linguagem de programação.

Exemplo de um Pseudocódigo:

```
1.  Var
2.     z,p: inteiro
3.   Início
4.        Ler: (z,p)
5.        Se (z=p) então
6.             Mostrar (“Valores iguais!”)
7.        Senão
8.             Mostrar (“Valores diferentes!”)
9.        Fimse
10. Fim
```

Podemos perceber que temos um conjunto de regras bem definidas, certo? É fundamental que um algoritmo em pseudocódigo comece sempre com a palavra "**Algoritmo**” seguido pela atribuição de um nome ao código. Da mesma forma, é essencial que termine com a palavra "**Fim**". Qualquer outra designação quebra a formalidade do pseudocódigo, causando erro. Termos como: ler, mostrar, se, então, entre outros, são usados intencionalmente e possuem significados específicos.

Princípios Python

Para começar vou te ensinar alguns passos que são essenciais quando se está programando. São eles:

- Sempre verifique (duas, três, quatro vezes!) se você digitou corretamente seu código.
- Letras maiúsculas e minúsculas são diferentes dentro da programação; fique atento a isso!
- Sempre que abrir aspas, lembre-se de fechá-las. O mesmo vale para parênteses. Exemplo: “Teste”, (Projeto).
- Preste atenção aos espaços entre os códigos. No decorrer deste projeto, você irá entender as diversas operações que utilizam isso.

Print

Esse código tem a principal ação de mostrar na tela algo que foi solicitado. Como eu uso? Vamos ver sua sintaxe, que nada mais é do que a forma como ele é escrito:

```
print ( ' Olá Mundo! Vamos estudar juntos! ' )
```

Temos a função, abrindo os parênteses, abrindo as aspas, a mensagem, fechando as aspas e fechando os parênteses. **Podemos usar aspas simples ou aspas duplas.**

É assim que fazemos o código Print. Dessa forma, a mensagem que colocarmos dentro desse código será retornada para o usuário. No nosso exemplo, ela aparecerá assim:

```
Olá Mundo! Vamos estudar juntos!
```

Tudo que colocarmos dentro das aspas será retornado exatamente igual. Por isso, vamos testar algumas contas de matemática. Primeiro, faremos como foi ensinado, depois faremos sem as aspas.

```
print ('4+8')
print (4+8)


4+8
12
```

O primeiro resultado nos deu exatamente o que escrevemos dentro das aspas: '4+8'. Já o segundo resultado nos deu a soma matemática: 12.

Com isso, podemos entender que, sem as aspas, o programa entende que estamos fazendo uma operação matemática. Já com as aspas, o programa entende que estamos digitando um texto. Assim, para usarmos o print com operações matemáticas, fazemos da seguinte maneira:

```
print ( 4 + 8 )
```

Temos a função, abrindo os parênteses, a mensagem e fechando os parênteses. **Podemos usar aspas simples ou aspas duplas.**

Através do print, também podemos juntar duas mensagens. Usando nosso último exemplo, colocando somente os números entre as aspas e deixando o operador de adição fora.

Exemplo: **print ('4' + '8')**

Esse resultado nos daria a seguinte situação: 48. Pois o programa entendeu que os números 4 e 8 entre as aspas são textos e não números. E o sinal do operador de adição "junta" esses textos. Isso também serve para textos, veja na sequência.

Faça o teste em seu computador junto comigo:

print (“Código” + “Em Construção”)

Lembre-se de que o programa irá juntar as duas mensagens, logo é interessante colocar um espaço entre as aspas e a segunda mensagem.

```
Código Em Construção
```

Operadores Matemáticos

Vimos que podemos usar o Python para realizar operações matemáticas, certo? Então vamos aprender como utilizamos cada operação. Abaixo, temos uma tabelinha que vai nos ajudar com isso:

Figura 6 – Tabela de operações matemáticas

Python	Operação
+	Adição
-	Subtração
*	Multiplicação
/	Divisão (com casa decimais)
//	Divisão (somente a parte inteira)
%	Módulo/resto da divisão
**	Exponenciação ou potenciação

Os cálculos dentro do programa funcionam igual à matemática tradicional, ou seja, a ordem de precedência dos operadores deve ser respeitada, assim como fazemos nossas contas no caderno. Devemos prestar muita atenção aos parênteses, pois eles vão ditar a ordem da operação.

Exemplo: $6 \times \frac{(6+2)}{4}$

Como podemos realizar essa operação dentro do Python?

```
print (6 * (6 + 2) /4)
print (6 * (6 + 2) //4)


12.8
12
```

Na primeira linha do código, utilizamos ***/*** para a divisão, e o resultado veio com casas decimais (**números depois da vírgula**). Já na segunda linha, utilizamos ***//***, e o resultado veio somente com os números inteiros (**sem números após a vírgula**).

Variáveis / Dados

Agora iremos estudar sobre variáveis, vamos entender o que elas são, como se dividem e o que podemos utilizar delas:

Uma variável é um nome dado para uma região da memória do programa, onde, toda vez que ela for utilizada, o programa saberá exatamente como manipulá-la. Existem 3 tipos de variáveis que usamos no Python:

- Numérico: Representam qualquer número, sejam inteiros ou com casas decimais. Utilizamos essa variável quando precisamos realizar operações aritméticas (como as contas que fizemos antes).
- Caractere: Serve para representar letras, caracteres especiais, acentuações e também números (quando não forem usados em operações aritméticas).
- Literal/Booleano: Serve para identificar somente dois estados: verdadeiro ou falso (1 ou 0).

Para criar uma variável, é bem simples. Primeiro, escolhemos um nome para ela e, em seguida, colocamos um sinal de igual para armazenar a informação naquele nome. Por exemplo:

a) **projeto = ‘Código em construção’**
b) **professor = ‘Moroni Fernandes’**
c) **materiais = ‘2’**

Agora, poderíamos adicionar um print para as variáveis que escolhemos:

print (projeto)

print (professor)

print (materiais)

Dessa forma, teríamos as variáveis e seus conteúdos armazenados, e o resultado seria o seguinte:

```
projeto = "Código em construção"
professor = "Moroni Fernandes"
materiais = "2"
print(projeto)
print(professor)
print(materiais)

Código em construção
Moroni Fernandes
2
```

Porém, não podemos criar uma variável de qualquer jeito. Existem algumas regras que limitam essa ação. Temos uma tabelinha onde podemos encontrar o que fazer e o que não fazer:

Figura 7 – Tabela variável válida e inválida

Nome	Permitido?	Explicação
idade	Sim	Nome formado somente por letras/caracteres não especiais.
v3	Sim	Números são permitidos desde que não no início da palavra.
3v	Não	Não podemos iniciar uma variável com um número.
Maior_nota	Sim	Podemos usar caracteres maiúsculas e o sublinha sem problemas.
Maior nota	Não	O uso de espaços não é permitido em nomes de variáveis.
maior	Sim	Podemos usar a sublinha em qualquer parte da variável, inclusive no início.
#maior	Não	Caracteres especiais não são permitidos em nenhuma parte do nome da variável.
adicao	Sim	Nome formado somente por letras/caracteres não especiais.
Adição	Parcialmente	Somente o Python 3 permite caracteres com acentuação. Recomenda-se fortemente que evite esta prática, pois quase nenhuma linguagem admite isso.

Variáveis Numéricas

Existem dois tipos de variáveis numéricas: inteiros e ponto flutuante! A diferença entre as duas é bem simples.

Os **inteiros** (**int**) fazem parte dos números naturais e inteiros, aqueles que vimos lá na aula de matemática, lembra? (... -5, -4, -3, -2, -1, 0, 1, 2, 3...) Caso não se lembre, são os números que não possuem vírgula.

Já o **ponto flutuante (float)** se refere aos números que possuem casas decimais, ou seja, que têm vírgula (na língua inglesa e na programação, usamos **ponto ao invés de vírgula**). Exemplos: (15.25, 1.99, 100.00, -129.99).

Cuidado! Mesmo que um ponto flutuante tenha zero como seus decimais, ele continua sendo um valor de ponto flutuante!

Variáveis Lógicas

Essa variável é responsável por armazenar apenas dois estados: verdadeiro (true) ou falso (false). Através dessa variável, podemos realizar operações lógicas. Veja na tabela abaixo os operadores lógicos, tanto em Python, pseudocódigo e a operação em si:

Figura 8 – Tabela operadores lógicos

Python	Pseudocódigo	Operação
==	=	Igualdade
>	>	Maior que
<	<	Menor que
>=	>=	Maior ou igual que
<=	<=	Menor ou igual que
!=	<>	Diferente

Temos que tomar cuidado para não errar ao utilizar o sinal de igual! (=)

Quando utilizamos ele uma vez, tem a função de atribuição, já duas vezes possui o significado de igualdade. Por exemplo:

```
z = 5
m = 10
resposta = z == m
print(resposta)

False
```

A leitura aqui seria: "**z é igual a m**?" Como sabemos que 5 não é igual a 10, a resposta seria **False** (**falso**).

Caso a pergunta fosse da seguinte maneira, o resultado seria diferente:

```
z = 5
m = 10
resposta = z != m
print(resposta)

True
```

Aqui estamos afirmando que "**z é diferente de m**", o que sabemos que é algo verdadeiro. Logo, a resposta é **True** (**verdadeiro**).

Cadeia de Caracteres (Strings)

O último tipo de variável que temos são os caracteres, que são strings. Essas strings são onde iremos armazenar conjuntos de símbolos, frases inteiras, acentuações, pontuações, números, tabulação, tudo isso em uma única variável! Vejamos o exemplo:

```
frase = "Essa frase é um teste!"
print(frase)


Essa frase é um teste!
```

O Python e outras linguagens de programação nos dão a opção de acessar partes específicas do texto da string, já que, para o programa, cada caractere é tratado como um dado individual na memória.

Índice da String: É um número que indica a posição do caractere na cadeia. O primeiro índice é sempre zero. Logo, para acessarmos o índice zero, chamamos o nome da variável e colocamos colchetes. Dentro dos colchetes, inserimos o índice, que é um valor inteiro.

Observe que o índice é sempre um número a menos do que a posição desejada, uma vez que nossos índices iniciam a contagem em zero:

```
print(frase [0])
print(frase [5])
print(frase [8])

E

f

s
```

Concatenação

Lembra quando vimos sobre o comando print e juntamos alguns números em uma frase ao invés de somá-los? Então, iremos fazer isso com as strings, de forma que montaremos frases que, inicialmente sozinhas, não possuem sentido, mas juntas trazem entendimento.

```
mf = "Código em construção: "
mf = mf + "Python e HTML nas escolas"
print(mf)

Código em construção Python e HTML nas escolas
```

Utilizamos a concatenação através da adição. Em nosso exemplo, criamos a variável **mf**, onde damos a ela uma parte de um nome. Em seguida, concatenamos o que já tínhamos com outra parte de um nome e imprimimos na tela.

Composição

Agora iremos ver uma função muito importante: a composição, que é capaz de misturar texto e dados numéricos em um print. Preste bastante atenção, pois isso será algo que utilizaremos muito em Python!

Podemos colocar o valor de uma variável dentro de outra variável que seja do tipo string. Para isso, iremos utilizar as chaves **{}**, que vamos chamar de marcadores de posição. Esse símbolo será colocado dentro do nosso texto, no local exato onde o valor de uma variável deve aparecer. Assim, ele irá marcar a posição da variável, que será substituída pelas chaves durante a execução do programa pelo valor da variável.

```
presenca = 75.5
mf = "Você alcançou {} por cento de presença no curso!" .format (presenca)
print(mf)

Você alcançou 75.5 por cento de presença no curso!
```

Perceba que temos nossa variável **string mf**, que imprime na tela uma **presenca**, portanto um dado numérico. Em um ponto específico da frase, existe o **símbolo de chaves**. Este símbolo será substituído pelo valor da variável **presenca**. A variável está posicionada após o término da frase, onde existe o comando **.format** e o respectivo nome. Isso significa que a variável **presenca** terá seu valor inserido no lugar de **{}** dentro do texto.

Vamos agora, como último exemplo desse tema, juntar o que aprendemos? Vamos utilizar uma variável de presença e uma de curso e printá-las no PyCharm:

```
presenca = 75.5
curso = "Código em construção: Python e HTML nas escolas"
mf = "Você alcançou {} por cento de presença no projeto {}!"
.format(presenca,curso)
print(mf)

Você alcançou 75.50 por cento de presença no projeto Código em construção:
Python e HTML nas escolas!
```

Input

Já aprendemos a utilizar o print e também tudo sobre criar uma variável. Agora vamos conhecer a função **input**, que torna o computador capaz de "**conversar**" conosco. Mas como assim? Bom, ele irá perguntar algo, e nós iremos responder.

```
input ( ' Insira seu nome: ' )
```

A mensagem que colocamos dentro do input será mostrada para o usuário. Em seguida, será aberta uma caixa de texto para o usuário inserir sua resposta. Essa resposta será armazenada em uma variável.

Para entender melhor, veja o exemplo a seguir, onde criamos uma variável chamada nome. Uma mensagem irá aparecer na tela solicitando ao usuário que coloque seu nome. Na sequência, o código print irá exibir na tela o nome que foi digitado:

```
nome = input ("Digite seu nome:")
print (nome)

Digite seu nome: Moroni
Moroni
```

Note que, enquanto não for digitada a resposta (em verde), o programa ficará "travado" no input. Somente após a inserção da resposta é que o programa seguirá para executar o print.

Agora iremos aprender como podemos colocar esse resultado do input dentro de uma frase. Para explicar, criarei um exemplo:

```
nome = input ('Qual é o seu nome:')
idade = int(input ('Qual é a sua idade?'))



print('Olá me chamo {} e tenho {} anos de idade!' .format(nome, idade))
```

Percebeu que utilizamos **int** e **.format** aqui também? Assim, podemos perceber que cada peça que vamos aprendendo complementa a anterior. Por isso, é muito importante ficarmos revisando o básico!

Lembrando que, dentro dos parênteses, a ordem em que colocamos as variáveis influencia como será mostrado no print.

```
Qual é o seu nome:Moroni
Qual é a sua idade?22
Olá me chamo Moroni e tenho 22 anos de idade!
```

Dessa forma, aprendemos a utilizar o comando input para inserir informações armazenadas dentro de variáveis e integrá-las em frases.

Resumo

Pseudocódigo:

- Possui regras definidas;
- É escrito em português estruturado;
- Utiliza uma linguagem genérica.

Print:

- A função serve para exibir informações na tela;
- Pode receber múltiplos argumentos, que serão impressos separados por vírgula;
- É a maneira mais essencial de visualizar resultados e mensagens durante a execução de um programa.

Operações aritméticas:

- O Python suporta as operações básicas da matemática: soma '**+**', subtração '**-**', divisão '***/***' e multiplicação '*****';
- A ordem das operações pode ser modificada usando parênteses;
- Para as divisões terem resultados com casas decimais, usamos "***/***", para termos resultamos inteiros usamos "***//***".

Variáveis:

- Usamos as variáveis para armazenar dados na memória;
- Não comece o nome de uma variável com um número, nem utilize espaços ou acentuações;
- Use nomes que estejam relacionados ao que queremos armazenar.

Variáveis Numéricas:

- Int: números naturais e inteiros sem casas decimais;
- Float: números com casas decimais; lembre-se de usar o ponto ao invés de vírgula!

Variáveis Lógica:

- Lembrar que um sinal de igual "**=**" é **atribuição**;
- Enquanto dois sinais de igual **"=="** indicam **igualdade**.

Strings:

- Nos índices de strings, o primeiro número é sempre zero;
- Utilizamos a concatenação através da adição;
- Na composição, usamos chaves **{}** como marcadores de posição dentro do texto e **.format** (**variável**) fora do texto.

Input:

- Permite que o usuário interaja com o programa.
- Através dele, podemos dar um nome ou resposta para uma variável;
- Quando utilizamos o comando **.format**, lembre-se que as variáveis devem ser colocadas na sequência desejada dentro dos parênteses.

Finalizando

Chegamos ao final da nossa aula! Espero que as informações que compartilhei possam colaborar para o seu aprendizado e influenciar positivamente o surgimento de novos estudantes na nossa tão querida programação.

Agradeço o esforço de tentar aprender algo novo e posso garantir que todo aprendizado, especialmente na área de tecnologia, será útil e trará ótimos benefícios para você!

Continue estudando; o segredo para o sucesso é disciplina e determinação! Bons estudos!

Referências

Download a última versão do Python. Python, S.d.(a) Disponível em: https://www.python.org/downloads/ Acesso em: 19 de dezembro 2023.

O zen do Python. Python, 19 ago. 2004. Disponível em: https://peps.python.org/pep-0020/ Acesso em: 22 de dezembro de 2023.

PERKOVIC, L. Introdução à computação usando Python – Um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.

PUGA, S.; RISSETI, G. Lógica de programação e estrutura de dados. 3. ed. São Paulo: Pearson, 2016.

MATTHES, E. Curso Intensivo de Python: uma introdução prática baseada em projetos à programação. São Paulo: Novatec, 2015.

MENEZES, N. N. C. Introdução à programação Python: algoritmos e lógica de programação para iniciantes. 3. ed. São Paulo: Novatec, 2019.

LÓGICA. In: Michaelis. Disponível em: https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/l%C3%B3gica/ Acesso em: 21 de dezembro de 2023.