Florianópolis – SC

Professor: Moroni Fernandes

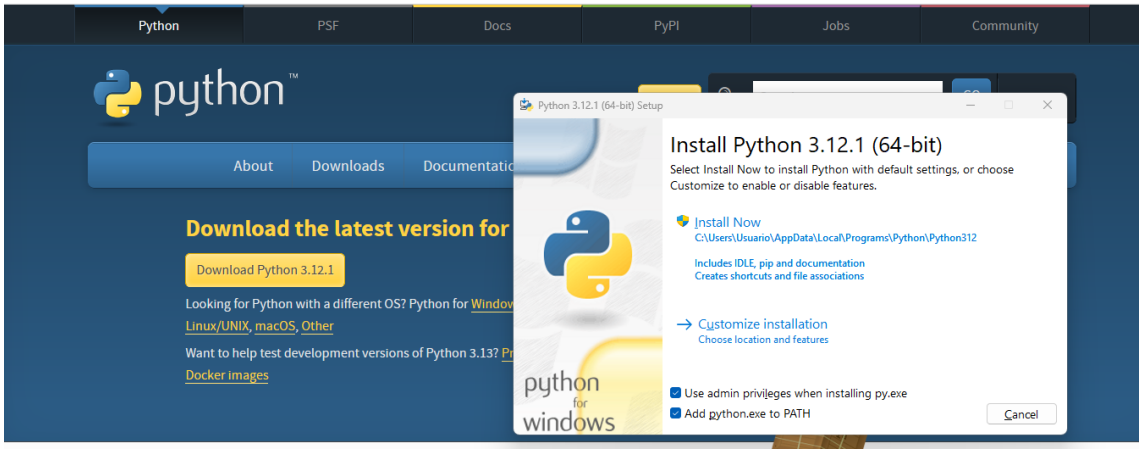
Instalação

Antes de aprendermos os princípios básicos sobre Python, devemos primeiro saber como instalar ele, já que este 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.

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

Após fazer o Download do arquivo a instalação acaba por sendo bem simples. Você deve marcar a opção **"AddPython.exe to PATH"** como na imagem abaixo e depois escolher a opção **"Install Now"**, de modo que o Python será instalado em sua máquina.

Figura 1 – Instalação Python



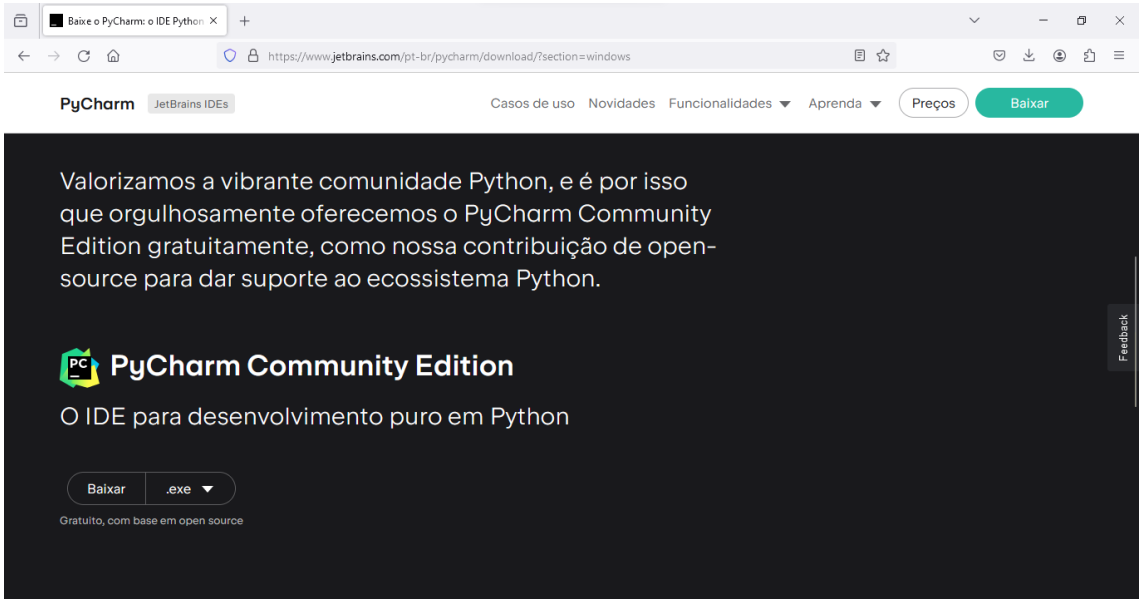
Acaba por aqui? Ainda não, temos ainda que instalar o **Pycharm**, que basicamente é um ambiente de desenvolvimento integrado com interface gráfica (IDE), qual sua função? Basicamente através dele que iremos conseguir desenvolver códigos em linguagem Python.

Link de acesso:

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

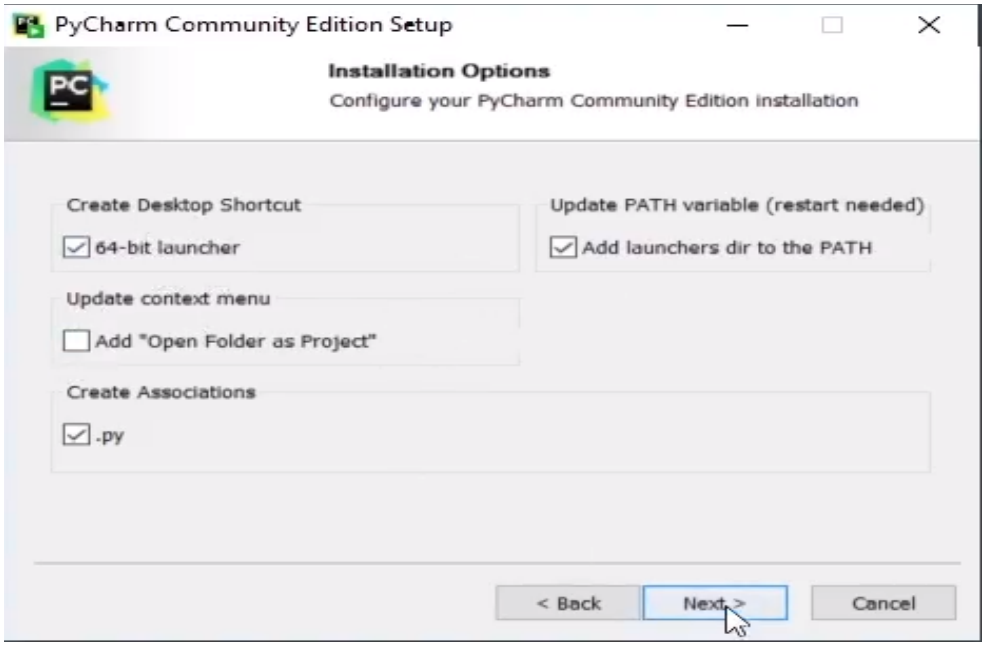
Acessando o link acima teremos duas opções, “Professional” e “Community”, você irá escolher essa 2° opção e ir em “Baixar”.

Figura 2 – Download Pycharm



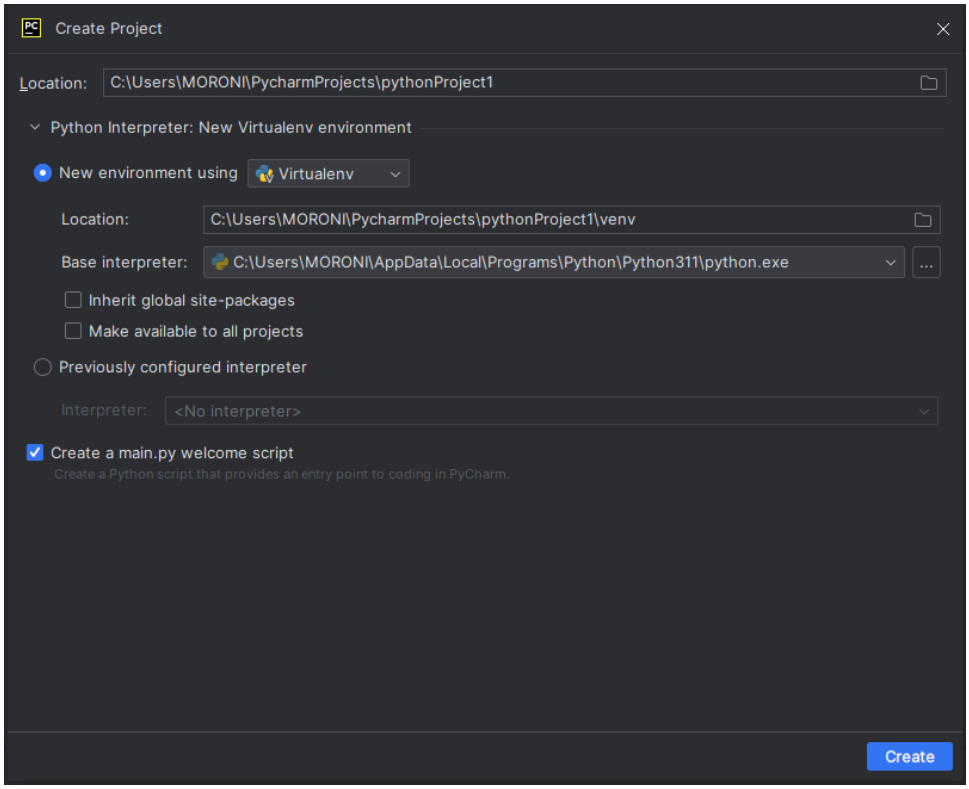
O Próximo passo é sua instalação: Começando por selecionar a opção Next na primeira tela, novamente Next na segunda tela, já na terceira tela marcar as opções: “64-bit launcher”, “Add launchers dir to the PATH” e “**.py”** Na quarta tela marcar **Next** e depois por último “**Install”**.

Figura 3 – Instalação Pycharm



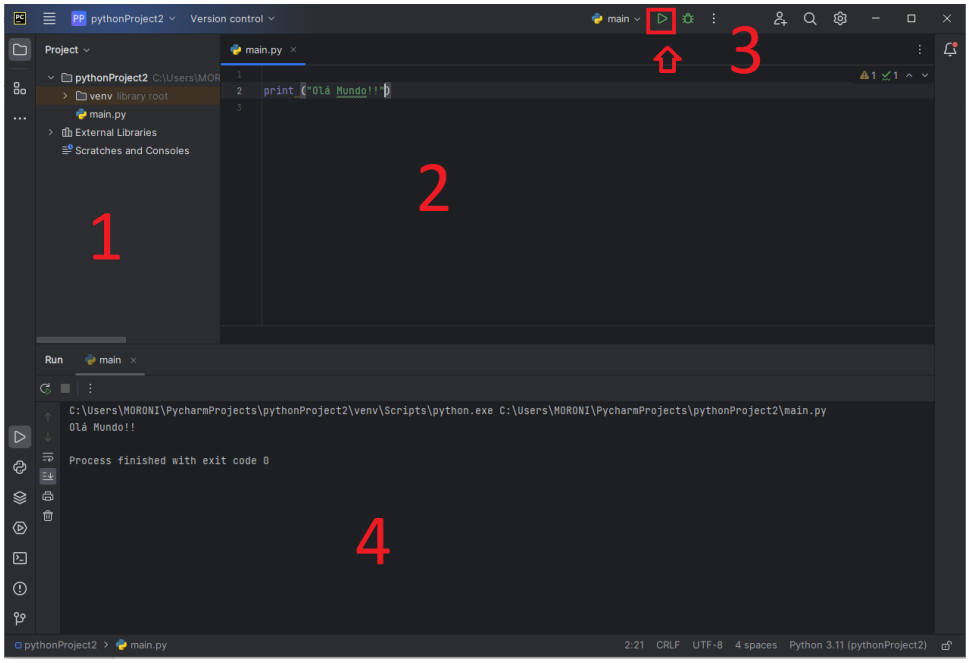
Agora com o Pycharm instalado, iremos abrir ele e de início iremos entrar na opção **New Project** para podermos iniciar nosso projeto. Nessa tela iremos escolher o local onde nossos arquivos irão ficar salvos. As demais opções podem deixar como estão na imagem abaixo. E enfim poderemos dar início ao nosso aprendizado sobre Python!

Figura 4 – Criando projeto



Na sequência teremos nossa tela inicial dentro do Pycharm, já dentro da pasta que escolhemos. **1.** Do lado esquerdo da tela temos as pastas e o arquivo **main.py** que é nele que começaremos nosso projeto. **2.** No meio da tela temos nossa aba onde iremos digitar nossos códigos. **3.** Para executar nosso código devemos apertar no símbolo no topo da tela mais à direita, que se parece com um triangulo de lado. Ou também temos a opção de apertar o botão direito na aba do código e selecionar "**Run (Nome do projeto)" 4.** O resultado da execução de nosso código sempre irá aparecer na aba mais embaixo da tela, tudo conforme a imagem abaixo.

Figura 5 – Conhecendo Pycharm



Para criar um projeto novo basta ir à área de pastas e apertar o botão direito do mouse e selecionar a opção **New -> Python File** e nomear seu projeto novo.

Algoritmos

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

Exemplo:

- Pegue o pote de achocolato
- Pegue com uma colher esse achocolatado e coloque duas vezes 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 representações dos Algoritmos como Descrição Narrativa, Pseudocódigo e Fluxograma, porém vamos focar em somente um destes: O Pseudocódigo.

Pseudocódigo: Possui regras definidas, sua linguagem é genérica, porém possui um português estruturado. É o mais próximo de um programa computacional sem ser 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" seguida pela atribuição de um nome ao seu código (linha 1). Da mesma forma, é essencial que termine com a palavra "Fim" (linha 11). Qualquer outra designação utilizada quebra a formalidade do pseudocódigo, acarretando erro. Termos como: ler; mostrar; se; então; entre outros, são empregados 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, se atente a isso!
- Sempre que abrir aspas, lembre-se de fechá-las. O mesmo vale para parênteses. Exemplo: “Teste”, (Projeto).
- Se atente aos espaços dados entre os códigos, no decorrer desse projeto você irá entender as diversas operações que utilizam disso.

Print

Esse código possui a principal ação de mostrar na tela algo que foi solicitado. Como o utilizo? Iremos ver sua sintaxe, que nada mais é do que a forma como ele é escrita:

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

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

É assim que fazemos o código Print, dessa forma a mensagem que colocarmos dentro desse código será retornado para o usuário. Em nosso exemplo ela voltaria assim:

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

Tudo que colocarmos dentro das aspas será retornado exatamente igual, devido a isso vamos testar algumas contas de matemática. Primeiro iremos fazer como foi ensinado, depois iremos fazer 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 utilizarmos o print com operações matemáticas fazemos da seguinte maneira:

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

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

Através do print podemos também juntar duas mensagens. Utilizando nosso último exemplo colocando somente os números entre as aspas, 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 um texto e não números. E o sinal do operador de adição “junta” esses textos. Isso serve para textos também, 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 utilizar o Python para realizarmos operações matemáticas, certo? Então vamos aprender como fazemos para utilizar cada operação. Abaixo temos uma tabelinha que irá 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 iguais na matemática tradicional, ou seja, a ordem de precedência dos operadores deve ser respeitada igual quando fazemos nossas contas no caderno. Devemos nos atentar muito aos parênteses, pois eles irão 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 / na parte da divisão e o resultado veio com casas decimais (números depois da vírgula). Já na segunda linha do código 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 manipular ela. Existem 3 tipos de variáveis que utilizamos 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 operações aritméticas).
- Literal/Booleano: Server para identificar somente dois estágios, são eles verdadeiro ou falso. (1 ou 0)

Para criarmos uma variável é algo bem simples, primeiro iremos escolher um nome para ela e na sequência iremos colocar um sinal de igual para armazenar a informação naquele nome. Por exemplo:

a) **projeto = 'Código em construção'**
b) **professor = 'Moroni Fernandes'**

c) **materias = ‘2’**

Agora poderíamos adicionar um print para as variáveis que escolhemos:

print (projeto) ; print (professor) ; print (materias)

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

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

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

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

Variáveis Lógicas

É a variável 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! (=)

Pois quando utilizamos ele uma vez, possui função de atribuição, já duas vezes possui significado de igualdade, por exemplo:

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

False
```

A leitura aqui seria então: z é igual a m? Como sabemos que 5 não é igual a 10 a reposta 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 é True (verdadeiro).

Cadeia de Caracteres (Strings)

O último tipo de variável que temos são os caracteres, essas strings serão 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 o zero, logo para acessarmos o índice zero chamamos o nome da variável e colocando 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 iremos montar frases que inicialmente sozinhas não possuem sentido, porém juntas trazer 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, na sequência

concatenamos o que já tínhamos com outra parte de um nome e imprimimos na tela.

Composição

Agora iremos ver uma função muito importante que é a composição, esta 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 uma outra variável que seja do tipo string. Para isso, iremos utilizar as chaves **{}** que vamos chamar de marcador de posição. Esse símbolo será colocado dentro de nosso texto, no local exato onde o valor de uma variável deve aparecer. Assim, ele irá marcar a posição da variável, que irá substituir as 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**, a qual imprime na tela uma presença, 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 printar elas 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”, função essa que torna o computador conversar conosco, mas como assim? Bom ele irá perguntar alguma coisa 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 referente ao solicitado e assim armazenar sua reposta em uma variável.

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

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

Digite seu nome: Moroni
Moroni
```

Note que na área onde o código é executado enquanto não for digitada a resposta o programa ficará “travado” no Input, somente após ser inserida a resposta é que o programa irá seguir para executar o print.

Agora iremos aprender como podemos colocar esse resultado do input inserido dentro de uma frase. Para explicar irei criar 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 o **int** e o **.format** aqui também? Logo podemos perceber que cada pecinha que vamos aprendendo, complementa a anterior. Dessa forma é muito importante ficarmos revisando o básico!

Lembrando que dentro dos parênteses a ordem que colocarmos irá influenciar 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 nosso comando input para podermos inserir informações armazenadas dentro de variáveis e inserir estas dentro de frases.

Resumo

Pseudocódigo:

- Regras definidas
- Português estruturado
- 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 começar o nome de uma variável com um número, nem utilizar espaços ou acentuações;
- Usar nomes relacionados com o que queremos pedir.

Variáveis Numéricas:

- Int: números naturais e inteiros sem casas decimais

- Float: números com casas decimais; lembrar de usar o ponto ao invés de vírgula!

Variáveis Lógica:

- Lembrar que um sinal de igual é atribuição,
- dois sinais de igual é igualdade

Strings:

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

Input:

- Permite o usuário interagir com o programa;
- Através dele que iremos dar um nome/resposta para uma variável;
- Quando for utilizado o comando **.format** lembrar que dentro dos parênteses as variáveis são na sequência que se deseja inseri-la.

Finalizando

Chegamos ao final dessa nossa aula, espero que as informações que passei a vocês possam colaborar para o seu aprendizado, que possa influenciar de forma positiva a termos novos estudantes para a nossa tão querida programação. Agradeço o esforço de tentar aprender algo novo e posso-lhe garantir que todo aprendizado novo e principalmente dentro dessa área que é a tecnologia será útil e poderá trazer 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.

