FIAP ON

LÓGICA DE PROGRAMAÇÃO

ENTRADA E SAÍDA DE DADOS

2

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

ENTRADA E SAÍDA DE DADOS

1 INTRODUÇÃO

Para desenvolvermos um algoritmo, devemos aprender oito comandos bases de programação. Estes oito comandos existem em todas as linguagens de programação. Conhecendo as suas estruturas fica fácil migrarmos para outras linguagens de programação porque o que pode mudar é o verbo e a sintaxe.

Os oito comandos são:

- Entrada de dados: Leia
- Saída de dados: Escreva
- Processamento de dados
- Decisão:
    - Se então
    - Se então senão
    - Se encadeado
- Escolha
- Estruturas de repetição:
    - Pré-condicional: Enquanto-faça
    - Pós-condicional: Faça-enquanto
    - Contador: Para

Aprenderemos cada uma destas instruções durante o curso. Aplicaremos cada um deles em pseudocódigo, Python e Java.

Cada linguagem de programação trata estes comandos de uma forma diferente, com regras diferentes (em alguns casos até iguais) e sintaxes e verbos diferentes.

Há linguagens de programação que não tem um ou mais destes oito comandos. Trataremos todas estas exceções durante o curso. Neste capítulo falaremos sobre os comandos de entrada, saída e processamento de dados.

Antes de entrarmos nos conceitos técnicos, vamos apresentar uma terminologia que será adotada na parte escrita do curso para um melhor entendimento do conteúdo:

Utilizaremos este quadro toda vez que desejarmos dar alguma dica, informação relevante ou específica

Utilizaremos este quadro sempre que formos falar de uma representação de linguagem: “Momento Pseudocódigo”, “Momento Python” e “Momento Java”

O curso está estruturado em desenvolvimento de algoritmos representado por: **Pseudocódigo**, **Python** e **Java**. Sendo assim, em toda resolução de problema utilizaremos estas três representações.

2 EXIBINDO DADOS PARA O USUÁRIO

No computador há periféricos de entrada, saída e processamento de dados. Neste ponto, falaremos sobre os de saída de dados.

Saída de dados representa todo momento em que o Algoritmo precisa interagir com o usuário. Então, o programador usa esta instrução toda vez que desejar que o Algoritmo mostre algo para o usuário.

Os meios e periféricos de saída de dados podem ser representados por um papel de cupom de máquina fiscal, uma folha de extrato bancário, impressoras e arquivos, mas nós utilizaremos a tela do monitor como saída de dados.

- O termo “Sintaxe” significa: regras de utilização de uma instrução

- Código-fonte: é o arquivo texto onde escrevemos os programas

- Comentário: é algum texto inserido dentro do código-fonte que seja um auxílio para o entendimento das instruções. O compilador o ignora.

Momento Pseudocódigo

Sintaxe:

```
Escreva <expressão>
```

Exemplos:

```
Programa saida_de_dados
Início
     // exibe "Meu primeiro programa
     Escreva "Meu primeiro programa"
     // exibe o número 12
     Escreva 12
Fim
```

Código-fonte 1 – Saída de dados no pseudocódigo
Fonte: Elaborado pelo autor (2022)

Obs.: O termo "expressão" usado no código-fonte se refere a uma mensagem, variável, cálculo ou qualquer uma destas combinações.

Momento Python

Sintaxe:

```
print(<expressão>)
```

Exemplos:

```
# Exibe "Meu primeiro programa"
print("Meu primeiro Programa")
# Exibe o número 12
print(12)
```

Código-fonte 2 – Saída de dados em Python
Fonte: Elaborado pelo autor (2022)

O Python tem três formas diferentes de exibir dados com um print. Vamos a elas:

1. Separando os termos com vírgula (,)

Obs.: "Termos" se referem a uma combinação de variáveis, constantes ou expressões

2. Utilizando a função format()
3. Utilizando o print(f"")

Vamos a alguns exemplos. Considere a codificação abaixo:

```
nome = "Edson"
idade = 48
peso = 85.64

# Forma 1
print("1. O meu nome é", nome, "tenho", idade, "anos e ", peso,
"Quilos")

# Forma 2
```

```
print("2. O meu nome é {} tenho {} anos e {} Quilos".format(nome,
idade, peso))
print("2. O meu nome é {0} tenho {1} anos e {2:.1f}
Quilos".format(nome, idade, peso))
print("2. O meu nome é {n} tenho {i} anos e {p:.2f}
Quilos".format(n=nome,i=idade,p=peso))

# Forma 3
print(f"3. O meu nome é {nome} tenho {idade} anos e {peso:.2f}
Quilos")
```

Código-fonte 3 – Saída de dados - Formatações em Python
Fonte: Elaborado pelo autor (2022)

A execução deste programa será:

```
1. O meu nome é Edson tenho 48 anos e  85.641 Quilos
2. O meu nome é Edson tenho 48 anos e 85.641 Quilos
2. O meu nome é Edson tenho 48 anos e 85.6 Quilos
2. O meu nome é Edson tenho 48 anos e 85.641 Quilos
3. O meu nome é Edson tenho 48 anos e 85.64 Quilos
```

Quadro 1 – Formatações em Python
Fonte: Elaborado pelo autor (2022)

Momento Java

Sintaxe:

```
System.out.println(<expressão>);
```

Exemplos:

```
// Exibe "Meu primeiro programa"
System.out.println("Meu primeiro programa");
// Exibe o número 12
System.out.println (12);
```

Código-fonte 4 – Saída de dados em Java
Fonte: Elaborado pelo autor (2022)

A execução deste programa será:

```
Meu primeiro programa
12
```

Quadro 2 – Saída de dados
Fonte: Elaborado pelo autor (2022)

Em Java utilizamos a classe System e o método out para exibir informações, objetos de cadeias de caracteres que contenham a implementação de formatação são instâncias de PrintWriter, uma classe de cadeias de caracteres, e PrintStream, uma classe de controle de bytes. Dois níveis de formatação estão disponíveis:

• print e println formatam valores individuais em um modo padrão;

• format formata qualquer valor numérico baseado numa String, quando são necessárias várias opções para uma formatação.

Os métodos print ou println enviam para a saída um simples valor após convertê-lo usando um método toString apropriado. Podemos ver o que ocorre no exemplo Equação:

```
public class Equacao {
    public static void main(String[] args) {
        int i = 2;
        double r = Math.sqrt(i);
        System.out.print("A raiz de ");
        System.out.print(i);
        System.out.print(" é ");
        System.out.print(r);
        System.out.println(".");
        i = 5;
        r = Math.sqrt(i);
        System.out.println("A raiz de " + i + " é " + r + ".");
    }
}
```

Código-fonte 5 – Métodos print, println e format em Java
Fonte: Elaborado pelo autor (2022)

A execução deste programa será:

```
A raiz de 2 é 1.4142135623730951.
A raiz de 5 é 2.23606797749979.
```

Quadro 3 – Métodos print, println e format em Java
Fonte: Elaborado pelo autor (2022)

3 RECEBENDO DADOS EM VARIÁVEIS

A instrução primitiva de Entrada de Dados é um complemento da instrução de Saída de dados. Entrada de dados é a instrução que faz a interação do usuário com o Algoritmo (ao contrário da Saída de dados).

Na saída de dados utilizaremos a tela do monitor como referência, na entrada de dados utilizaremos o teclado. Toda vez que o usuário tiver que inserir uma informação no algoritmo, o programador usará um comando de entrada de dados.

3.1 Variáveis de memória

Antes de prosseguirmos com o comando de entrada de dados, devemos conhecer o conceito de Variável de memória (RAM). Como diz o nome “variável” é algo que pode ter o seu conteúdo modificado no decorrer do algoritmo.

Uma variável é uma posição na memória do computador, com um nome e tipo, que armazena uma informação. Caso uma segunda informação seja colocada na mesma variável, o primeiro valor será sobreposto, armazenando assim o segundo valor.

3.2 Nomeando variáveis

Toda variável deve ter um nome próprio e único no algoritmo onde ela será utilizada. Somos nós, os programadores, quem definimos os nomes das variáveis. Para definirmos um nome de variável devemos seguir algumas regras gerais:

- O nome da variável não pode ser uma palavra reservada pela linguagem de programação, por exemplo: print
- O nome da variável deve começar com letra
- O nome da variável não pode conter caracteres especiais exceto o underline (_)

Como dito, as regras de nomeação de variáveis que vimos acima são regras gerais.

Há linguagens de programação com exceções como poder iniciar o seu nome com o caractere especial underline (_) caso da linguagem C e suas derivadas, outras que aceita cifrão ($) e deve iniciar com ele (linguagem PHP), entre outras.

A seguir, uma tabela com sugestões de nomes de variáveis válidos, inválidos e os motivos:

Tabela 1 – Nomeação de Variáveis de Memória

Descrição	Nome Válido	Nome Inválido	Motivo do Erro
Salário base	sal_base	salário_base	Acento no salário
Nome do pai	nome_pai	nome pai	Espaço entre nome e pai
Quantidade de filhos	qtd_filhos	qtd.filhos	Uso do ponto (.)
Escolher uma opção entre 1 e 2	Opção	1opcao	Não iniciou com letra
Observação	Observação	Observação	Uso do acento e cedilha
Maior de idade	Maioridade	Maior?	Interrogação no nome

Fonte: Elaborado pelo autor (2022)

Na Tabela acima, vimos exemplos de nomeação de variáveis de memória a partir de uma necessidade. Quanto ao nome da variável “ferir” o português, não se preocupe porque o nome delas não será visível ao usuário (também porque todas as linguagens de programação têm o inglês como língua nativa).

Apesar de não ser uma regra, é uma boa prática colocarmos nomes de variáveis com rótulos que lembrem a informação que ela armazenará.

Por exemplo, caso precisemos criar uma variável para armazenar um salário, será melhor o nome da variável como `salario` do que com o `nomes`. Apesar de ambas terem o nome válido.

3.3 Atribuindo um tipo às variáveis

Quando criamos variáveis de memória precisamos classificá-la com um tipo. Tipo é o formato em que o conteúdo será armazenado. Os tipos clássicos são:

- Caractere
- Texto
- Inteiro
- Real
- Lógico

Todos os demais tipos que existem em outras linguagens de programação, são derivados destes. Falaremos sobre cada um dos tipos clássicos:

- **Caractere:** utilizamos este tipo de dado quando precisamos armazenar apenas um caractere (entenda caractere como letra, dígito ou caractere especial) em uma variável. Por exemplo, `opcao = 's'`, sugerindo que a escolha foi [s]im a uma solicitação.

O conteúdo de um dado caractere pode estar entre apóstrofo (') ou aspas (").

- **Texto:** utilizamos este tipo de dado quando precisamos armazenar uma cadeia de caracteres como conteúdo de uma variável. Por exemplo, `endereco_fiap = "Av. Lins de Vasconcelos, 1222 – Aclimação"`. Como o tipo caractere, seu conteúdo pode estar entre aspas ou apóstrofo.
- **Inteiro**: este tipo de dado está na classificação dos numéricos. Os dados numéricos são aqueles que permitem que seja efetuado calculo com o seu conteúdo. Utilizamos este tipo de dado quando precisamos armazenar informações numéricas que não tenham casas decimais. Por exemplo, `qtd_cedulas_50 = 10`.

No conteúdo de uma variável numérica, seja inteiro ou real, colocamos apenas o número, sem a necessidade de caractere complementar.

- **Real:** este tipo de dado também está na classificação dos numéricos. Utilizamos este tipo de dado toda vez que precisamos armazenar valores que podem conter casas decimais (valores fracionários). Por exemplo, `preco_calca = 99.99`.

Pela linguagem de programação ser de língua inglesa, o separador de casas decimais é o ponto e não a vírgula, como usamos no português.

- **Lógico:** utilizamos este tipo de dado quando precisamos armazenar informações lógicas. Lógicas, são informações cujo conteúdo pode ser representado por Verdade (True) ou Falso (False). Por exemplo, `maior_de_idade = Verdade`.

Sendo assim, sempre que precisarmos de uma variável, devemos nomeá-la e atribuir um tipo antes de implementarmos as instruções do Algoritmo. Veja a aplicação:

Tabela 2 – Classificando as variáveis

Nome da variável	Tipo da variável
sal_base	Real
nome_pai	Texto
qtd_filhos	Inteiro
Opção	Caractere
Observação	Texto
Maioridade	Lógico

Fonte: Elaborado pelo autor (2022)

A teoria foi dada – sobre entrada de dados e variáveis de memória –, agora vamos implementar, em pseudocódigo, Python e Java, os conceitos vistos até então. Vejamos um exemplo aplicado em Pseudocódigo, Python e Java:

Momento Pseudocódigo

Sintaxe:

```
Leia <variável>
```

Exemplos:

```
Programa entrada_de_dados_e_variaveis
Var
    Salario: real
Inicio
    // Solicita o salário ao usuário
    Escreva "Digite o seu salário:"
    Leia salario
    // exibe o salário digitado
    Escreva "Salário = R$", salario
Fim
```

Código-fonte 6 – Entrada de dados no pseudocódigo
Fonte: Elaborado pelo autor (2022)

Momento Python

Sintaxe:

```
<variável> = input([<Mensagem>])
```

A mensagem dentro do input é opcional

Exemplos:

```
# Pede a digitação do salário ao usuário
salario = input("Digite o seu salário:")
salario = float(salario)
# Exibe o salário digitado
print("Salário = R$", salario)
```

Código-fonte 7 – Entrada de dados em Python
Fonte: Elaborado pelo autor (2022)

Em Python não declaramos variáveis com um tipo. A sua Tipagem é dinâmica, ou seja, no decorrer do programa ela pode assumir conteúdos de tipos diferentes.

A seguir, veja os exemplos das classificações das variáveis e utilização de casting (método de conversão de tipo):

```
nome = "Edson"       # Tipo Texto (string)
salario = 1234.56    # tipo real (float)
qtd_filhos = 2       # tipo inteiro (int)
opcao = 's'          # tipo caractere
maioridade = True    # tipo lógico

# Utilizando casting para modificar o tipo da variável
x = "55"             # x é do tipo string e vale '55'
x = float(x)         # x passou a ser float e vale 55.0
x = str(x)           # x voltou a ser string
x = int(x)           # x passou a ser int e vale 55
```

Código-fonte 8 – Tipos de dados e casting em Python
Fonte: Elaborado pelo autor (2022)

Podemos fazer o casting na variável: `salario = float(salario)`

ou

No próprio input: salario = float(input(“Digite o seu salário:”))

Como vimos, os tipos de dados em Python não são explícitos, sim dinâmicos. Isso quer dizer que a variável pode assumir diversos tipos no decorrer do programa. Então vem a pergunta: Como fazemos para saber o tipo atual dela? Utilizamos a função type(). Veja o seu funcionamento utilizando a codificação acima:

```
# UTILIZANDO VARIÁVEIS DIFERENTES E DESCOBRINDO OS TIPOS
nome = "Edson"       # Tipo Texto (string)
print(f"A variável nome = {nome} é do tipo {type(nome)}")
salario = 1234.56    # tipo real (float)
print(f"A variável salario = {salario} é do tipo {type(salario)}")
qtd_filhos = 2       # tipo inteiro (int)
print(f"A variável qtd_filhos = {qtd_filhos} é do tipo
{type(qtd_filhos)}")
opcao = 's'          # tipo caractere
```

```
print(f"A variável opcao = {opcao} é do tipo {type(opcao)}")
maioridade = True    # tipo lógico
print(f"A variável maioridade = {maioridade} é do tipo
{type(maioridade)}\n")

# UTILIZANDO A MESMA VARIÁVEL E DESCOBRINDO O SEU TIPO DEPOIS DE UM
CASTING
x = "55"             # x é do tipo string e vale '55'
print(f"x = {x} e é do tipo {type(x)}")
x = int(x)           # x passou a ser int e vale 55
print(f"x = {x} e é do tipo {type(x)}")
x = str(x)           # x voltou a ser string
print(f"x = {x} e é do tipo {type(x)}")
x = float(x)         # x passou a ser float e vale 55.0
print(f"x = {x} e é do tipo {type(x)}")
```

Código-fonte 9 – Descobrindo os tipos dos dados em Python
Fonte: Elaborado pelo autor (2022)

A execução deste programa será:

```
A variável nome = Edson é do tipo <class 'str'>
A variável salario = 1234.56 é do tipo <class 'float'>
A variável qtd_filhos = 2 é do tipo <class 'int'>
A variável opcao = s é do tipo <class 'str'>
A variável maioridade = True é do tipo <class 'bool'>

x = 55 e é do tipo <class 'str'>
x = 55 e é do tipo <class 'int'>
x = 55 e é do tipo <class 'str'>
x = 55.0 e é do tipo <class 'float'>
```

Quadro 4 – Descobrindo os tipos dos dados em Python
Fonte: Elaborado pelo autor (2022)

Momento Java

Sintaxe:

```
Scanner entrada = new Scanner(System.in);
...
<varInt> = <objeto>.nextInt(); // Lê variável inteira
<varFloat> = <objeto>.nextFloat(); // Lê variável inteira
<varString> = <objeto>.nextLine(); // Lê variável inteira
```

Exemplos:

```
Scanner entrada = new Scanner(System.in);
// Pede a digitação do salário ao usuário
salario = input("Digite o seu salário:")
System.out.println("Digite o seu salário:" );
salario = entrada.nextFloat();
// Exibe o salário digitado
System.out.println("Salário = R$" + salario);
```

Código-fonte 10 – Entrada de dados em Java

Fonte: Elaborado pelo autor (2022)

Em Java utilizamos a classe Scanner para fazer a entrada de dados, essa classe permite a leitura do buffer de teclado e guardar o conteúdo lido em uma variável do mesmo tipo definido.

Por meio do método next ele receberá a digitação de uma String até o pressionar do ENTER. Ao executar esse código, haverá uma interrupção no momento que o método next for executado, para que no prompt do terminal seja digitado algo. Essa digitação será capturada e mostrada na saída padrão como String. Abaixo outras formas de utilizar o método next para outros tipos de dados.

```
float numF = sc.nextFloat();

int num1 = sc.nextInt();

byte byte1 = sc.nextByte();

long lg1 = sc.nextLong();

boolean b1 = sc.nextBoolean();

double num2 = sc.nextDouble();

String nome = sc.nextLine();
```

4 OPERAÇÕES COM VARIÁVEIS NUMÉRICAS

Vimos as instruções de saída de dados e a de entrada de dados. Também há uma instrução primitiva de processamento de dados. Esta instrução não tem intervenção do usuário, ela simplesmente processa os dados colocados em expressões. Estas expressões no geral são atribuições e cálculos. Assim, quando você precisar de algum cálculo, utilize esta instrução.

Para aprendermos operações de cálculos, devemos prender os operadores que processam números.

Tabela 3 – Operadores aritméticos

Descrição	Pseudocódigo e Python	Java	Ordem de precedência
Parênteses	( )	( )	1
Exponenciação	**	Não existe	2
Multiplicação	*	*	3
Divisão	/	/	3
Módulo (Resto da divisão)	%	%	3
Divisão inteira	//	/	3
Soma	+	+	4
Subtração	-	-	4

Fonte: Elaborado pelo autor (2022)

Divisão inteira (//) e resto da divisão (%) em todas as linguagens de programação só funcionam com dados inteiros. Python é uma exceção, ou seja, podemos executar estas operações com dados reais também.

A seguir alguns exemplos de cálculos envolvendo operadores aritméticos com variáveis:

```
valor1 = 5
valor2 = 3
valor3 = 2
resposta = valor1 + valor2 * valor2     // Resulta 11
resposta = (valor1 + valor2) * valor2   // Resulta 16
valor1 = 17
valor2 = 7
resposta = valor1 % valor2          // Resulta 3
resposta = valor1 // valor2         // Resulta 2
resposta = valor1 / valor2          // Resulta 2.428
resposta = 2 ** 4                   // Resulta 16
```

Código-fonte 11 – Cálculos com variáveis e operadores aritméticos
Fonte: Elaborado pelo autor (2022)

5 DE-PARA: PSEUDOCÓDIGO PARA JAVA E PYTHON

Neste capítulo aprendemos o comando de entrada, saída e processamento de dados; também aprendemos o conceito de variáveis, operadores e operações aritméticas.

A partir destes conceitos, vamos sugerir alguns algoritmos e resolvê-los em Pseudocódigo, Python e Java.

Exemplo 1:

Dados quatro (4) números pelo usuário, fazer um algoritmo que calcule a média simples destes números.

Momento Pseudocódigo

```
Programa exemplo1
Var
    n1, n2, n3, n4, media: real
Início
    // Solicita quatro números ao usuário
    Escreva "Digite 4 números:"
    Leia n1
    Leia n2
    Leia n3
    Leia n4
    // Calcula a média dos 4 números
    Media = (n1 + n2 + n3 + n4) / 4
    // Calcula a média dos 4 números
    Escreva "Média = ", media
Fim
```

Código-fonte 12 – Pseudocódigo do algoritmo média de 4 notas
Fonte: Elaborado pelo autor (2022)

No pseudocódigo não importa se as letras dos nomes das variáveis estão em maiúsculo ou minúsculo, funciona.

No pseudocódigo, se quisermos separar uma frase de uma variável (ou vice-versa) no comando, Escreva: utilizamos a vírgula.

Momento Python

```
# Solicita quatro números ao usuário
print("Digite 4 números:")
n1 = input()
n1 = float(n1)
n2 = float(input())
n3 = float(input())
n4 = float(input())
# Calcula a média dos 4 números
media = (n1 + n2 + n3 + n4) / 4
print("Média = ", media)
```

Código-fonte 13 – Python do algoritmo média de 4 notas
Fonte: Elaborado pelo autor (2022)

Repare na leitura do n1. Ele está recebendo o input(), na linha de baixo o n1 é convertido para float e jogado nele mesmo. Nas próximas variáveis estes dois processos são feitos na mesma linha.

O input() pega o dado fornecido pelo usuário e sempre o retorna no tipo string. Caso queira o dado em outro tipo, faça o Casting

Momento Java

```java
import java.util.Scanner;
public class Media {
	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);
		//Declaração das variáveis
		float n1, n2, n3, n4, media;
		// Solicita quatro números ao usuário
		System.out.println("Digite 4 números:");

		n1 = entrada.nextFloat();
		n2 = entrada.nextFloat();
		n3 = entrada.nextFloat();
		n4 = entrada.nextFloat();
		// Calcula a média dos 4 números
		media = (n1 + n2 + n3 + n4) / 4;
		System.out.println("Média = " +  media);
	}
}
```

Código-fonte 14 – Java do algoritmo média de 4 notas
Fonte: Elaborado pelo autor (2022)

Exemplo 2:

Dado o **preço do maço** de cigarros, a **quantidade** de maços consumidos por dia e o tempo em **anos** que a pessoa fuma, calcular quanto esta pessoa já gastou fumando.

Momento Pseudocódigo

```
Programa exemplo2
Var
    preco_maco, qtd_maco, anos, dias_fumante, custo: real
Início
    // Solicitar os dados ao usuário
    Escreva "Digite o preço do maço:"
    Leia preco_maco
    Escreva "Digite a quantidade de maços:"
    Leia qtd_maco
    Escreva "Digite a qtd. de anos que fuma:"
    Leia anos
    // calcula a qtd. de dias como fumante
```

```
        dias_fumante = anos * 365
        // calcula a qtd. de dias como fumante
        dias_fumante = anos * 365
        // calcula o gasto do tempo que fuma
        custo = dias_fumante * preco_maco
        // Exibe o custo
        Escreva "Você já gastou R$ ", custo, "Fumando"
Fim
```

Código-fonte 15 – Pseudocódigo do algoritmo custo x cigarro
Fonte: Elaborado pelo autor (2022)

Momento Python

```python
# Solicitando os dados ao usuário
preco_maco = float(input("Digite o preço do maço: "))
qtd_maco = float(input("Digite a quantidade de maços:"))
anos = float(input("Digite a qtd. de anos que fuma:"))
# calcula a qtd. de dias como fumante
dias_fumante = anos * 365
# calcula o gasto do tempo que fuma
custo = dias_fumante * preco_maco
# Exibe o custo
print("Você já gastou R$ ", custo, "Fumando")
```

Código-fonte 16 – Python do algoritmo custo x cigarro
Fonte: Elaborado pelo autor (2022)

Repare na leitura do n1. Ele está recebendo o input(), na linha de baixo o n1 é convertido para float e jogado nele mesmo. Nas próximas variáveis estes dois processos são feitos na mesma linha.

O input() pega o dado fornecido pelo usuário e sempre o retorna no tipo string. Caso queira o dado em outro tipo, faça o Casting

Momento Java

```java
import java.util.Scanner;

public class Exemplo2 {

  public static void main(String[] args) {
      Scanner entrada = new Scanner(System.in);

      //Declaração de variáveis
      float preco_maco, qtd_maco, anos, dias_fumante, custo;
      // Solicitar os dados ao usuário
      System.out.println("Digite o preço do maço:");
      preco_maco = entrada.nextFloat();
      System.out.println("Digite a quantidade de maços:");
      qtd_maco = entrada.nextFloat();
      System.out.println("Digite a qtd. de anos que fuma:");
      anos = entrada.nextFloat();
```

```
        // calcula a qtd. de dias como fumante
        dias_fumante = anos * 365;
        // calcula a qtd. de dias como fumante
        dias_fumante = anos * 365;
        // calcula o gasto do tempo que fuma
        custo = dias_fumante * preco_maco;
        // Exibe o custo
        System.out.println("Você já gastou R$ " + custo + "Fumando");
    }
}
```

Código-fonte 17 – Java do algoritmo custo x cigarro
Fonte: Elaborado pelo autor (2022)

Exemplo 3:

Dada uma quantia (considere que já seja múltiplo de 10), fazer um algoritmo que calcule quantas cédulas de 10, 20 e 50 são necessárias para compor esta quantia.

Momento Pseudocódigo

```
Programa exemplo3
Var
    ced10, ced20, ced50: inteiro
Início
    /* Solicita a quantia */
    Escreva "Digite a quantia:"
    Leia quantia
    /* Efetua os cálculos das quantias de cédulas */

    ced50 = quantia // 50
    quantia = quantia % 50
    ced20 = quantia // 20
    quantia = quantia % 20
    ced10 = quantia // 10
    quantia = quantia % 10
    /* Exibe as quantidades de cédulas */
    Escreva "Quantidade das cédulas: "
    Escreva "Cédulas de 50 = ", ced50
    Escreva "Cédulas de 50 = ", ced20
    Escreva "Cédulas de 50 = ", ced10
Fim
```

Código-fonte 18 – Pseudocódigo do algoritmo cédulas
Fonte: Elaborado pelo autor (2022)

Momento Python

```python
# Solicita a quantia
quantia = int(input("Digite a quantia: "))
# Efetua o cálculo das quantias de cédulas
ced50 = quantia // 50
quantia = quantia % 50
ced20 = quantia // 20
quantia = quantia % 20
ced10 = quantia // 10
quantia = quantia % 10
# Exibe as quantidades de cédulas
print("Quantidade das cédulas 50: ", ced50)
print("Quantidade das cédulas 20: ", ced20)
print("Quantidade das cédulas 10: ", ced20)
```

Código-fonte 19 – Python do algoritmo cédulas
Fonte: Elaborado pelo autor (2022)

Repare na leitura do n1. Ele está recebendo o input(), na linha de baixo o n1 é convertido para float e jogado nele mesmo. Nas próximas variáveis estes dois processos são feitos na mesma linha.

O input() pega o dado fornecido pelo usuário e sempre o retorna no tipo string. Caso queira o dado em outro tipo, faça o Casting

Momento Java

```java
import java.util.Scanner;

public class Saque {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        //Declaração de variáveis
        int quantia, ced50, ced20, ced10;
        // Solicita a quantia
        System.out.println("Digite a quantia: ");
        quantia = entrada.nextInt();
        // Efetua o cálculo das quantias de cédulas
        ced50 = quantia / 50;
        quantia = quantia % 50;
        ced20 = quantia / 20;
        quantia = quantia % 20;
        ced10 = quantia / 10;
        quantia = quantia % 10;
        // Exibe as quantidades de cédulas
        System.out.println("Quantidade das cédulas 50: " + ced50);
        System.out.println("Quantidade das cédulas 20: " + ced20);
        System.out.println("Quantidade das cédulas 10: " + ced20);
    }
}
```

Código-fonte 20 – Java do algoritmo custo x cigarro
Fonte: Elaborado pelo autor (2022)

ANEXO I – INSTALAÇÃO E UTILIZAÇÃO DO PYTHON

Neste anexo, mostraremos um tutorial de como instalar o Python, PyCharm (IDE) e criar uma aplicação simples.

Instalando o Python:

No Google digite: *"*download python*"* e escolha a primeira opção

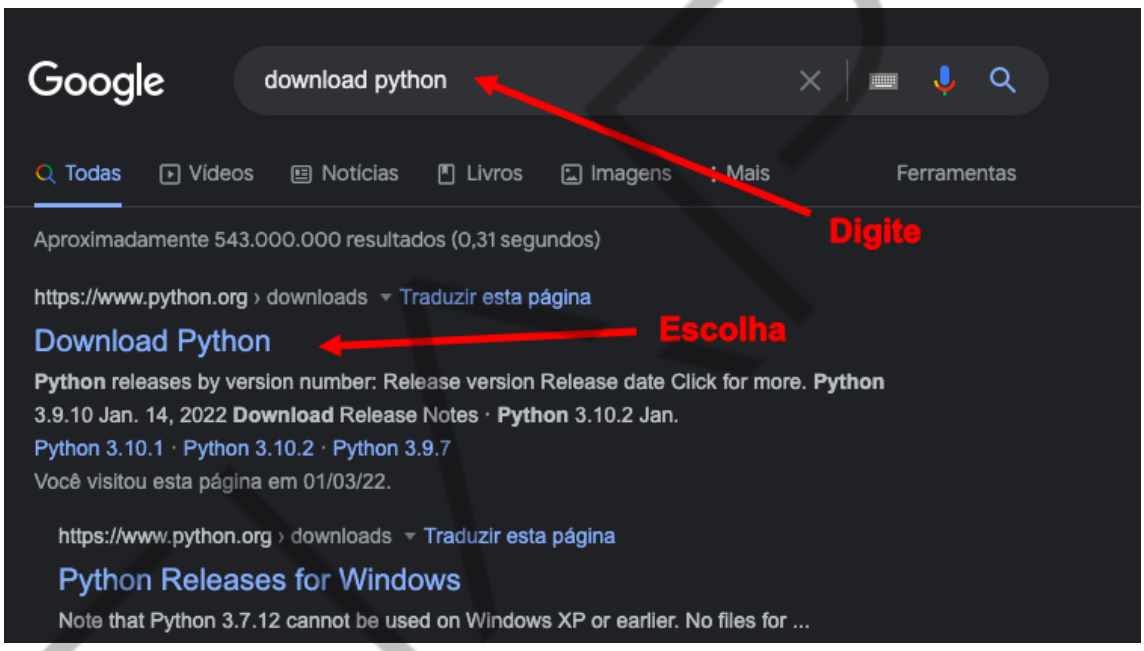

Figura 1 – Como fazer o download de Python (1)
Fonte: Elaborada pelo autor (2022)

- ✓ Clique em "Downloads"
- ✓ Escolha o seu Sistema Operacional
- ✓ Selecione a última versão estável

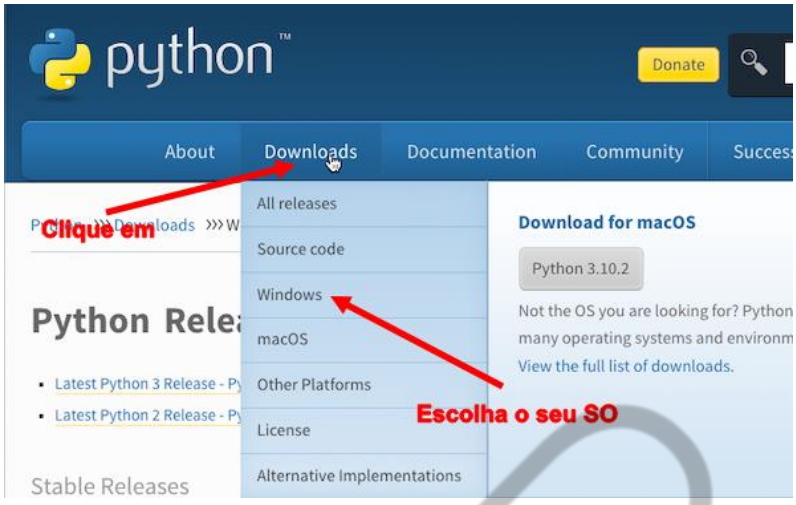

Figura 2 – Como fazer o download de Python (2)
Fonte: Elaborada pelo autor (2022)

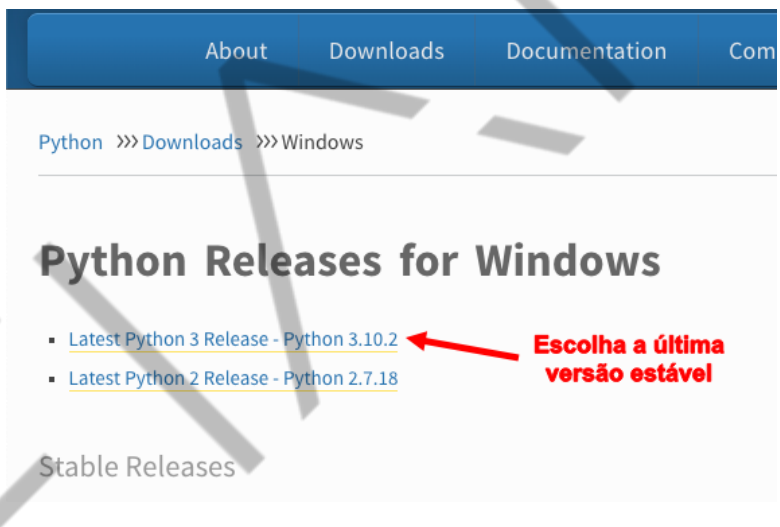

Figura 3 – Como fazer o download de Python (3)
Fonte: Elaborada pelo autor (2022)

- ✓ Marque “Add Python to PATH”
- ✓ Prossiga com a instalação

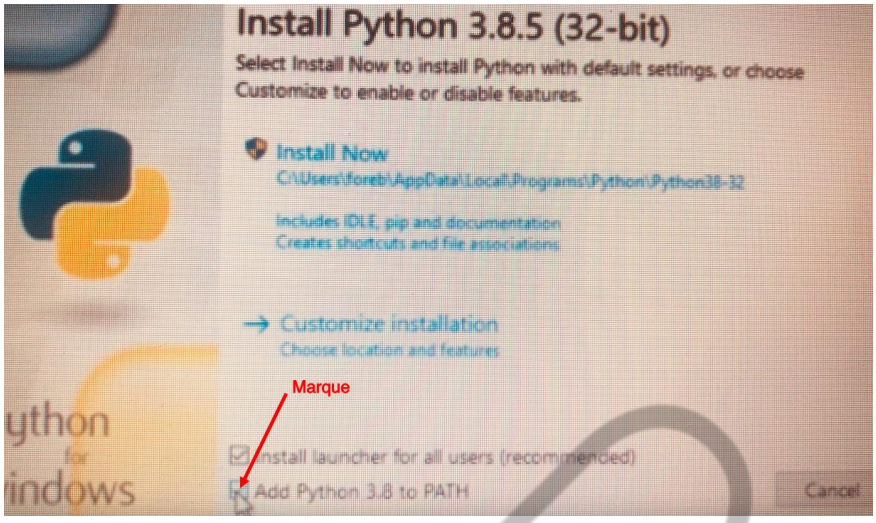

Figura 4 – Como fazer o download de Python (4)
Fonte: Elaborada pelo autor (2022)

Instalando o Pycharm

- ✓ No Google digite: “download pycharm community” (essa é a versão gratuita)
- ✓ Escolha a primeira opção

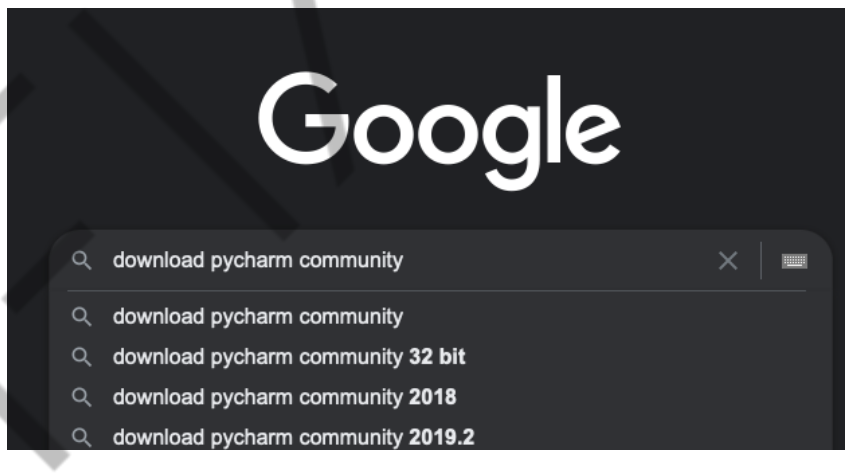

Figura 5 – Como fazer o download de Pycharm (1)
Fonte: Elaborada pelo autor (2022)

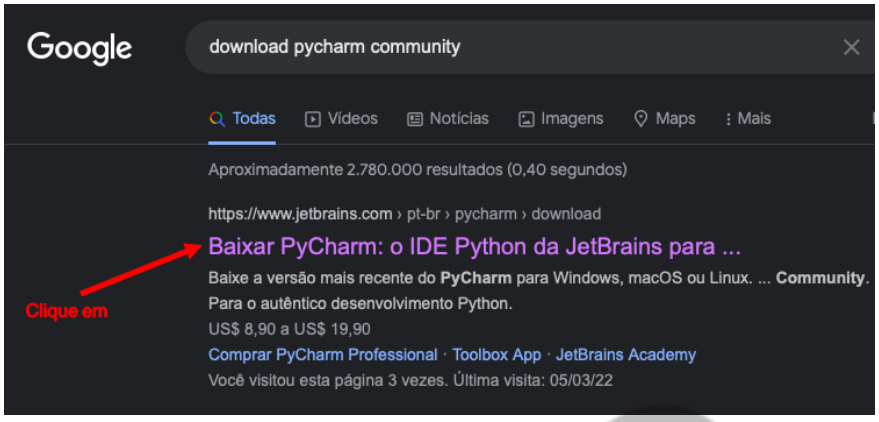

Figura 6 – Como fazer o download de Pycharm (2)
Fonte: Elaborada pelo autor (2022)

Neste site você deve escolher o SO e a versão do PyCharm desejada:

- ✓ Escolha a versão gratuita (Community)
- ✓ Dê dois cliques e Execute o arquivo baixado
- ✓ Selecione “Next” seguindo o padrão de instalação

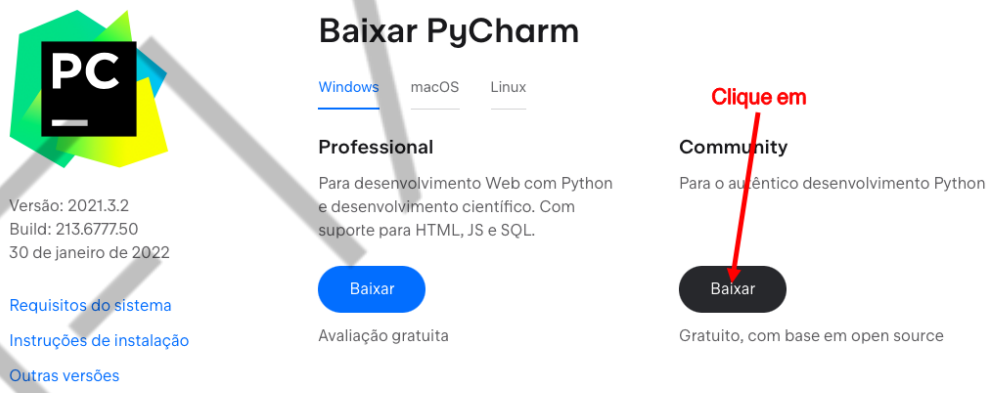

Figura 7 – Como fazer o download de Pycharm (3)
Fonte: Elaborada pelo autor (2022)

Criando um projeto no Pycharm:

- ✓ Clique em “New Project”
- ✓ Direcione o caminho da pasta onde deseja salvar os projetos
- ✓ Clique no botão “Create”

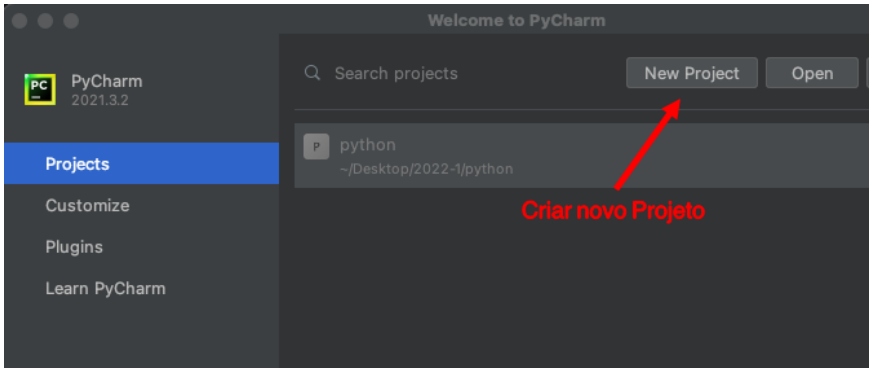

Figura 8 – Criando um projeto no Pycharm (1)
Fonte: Elaborada pelo autor (2022)

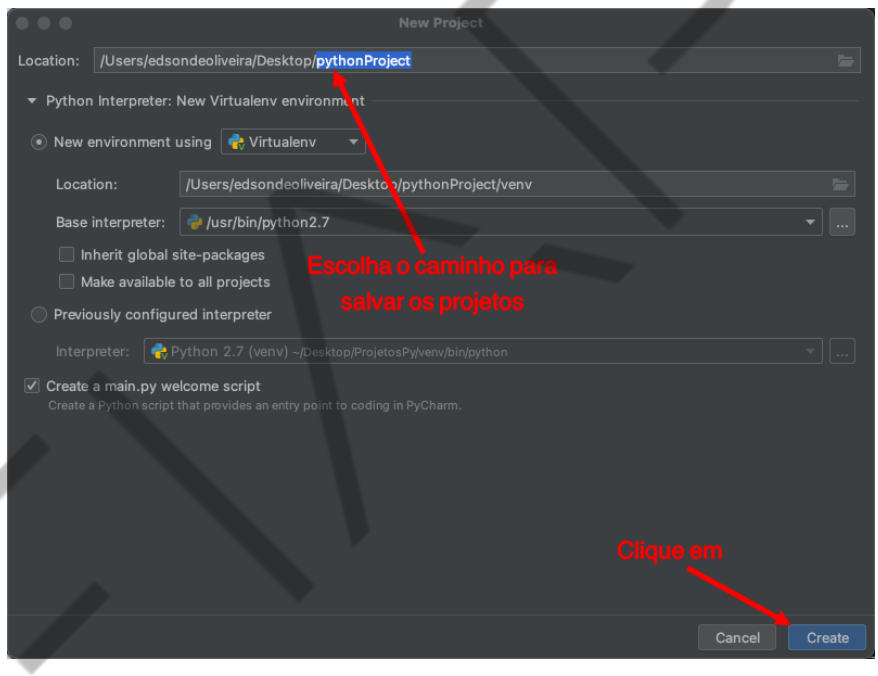

Figura 9 – Criando um projeto no Pycharm (2)
Fonte: Elaborada pelo autor (2022)

Crie uma pasta para o seu projeto:

- ✓ Clique com o Botão direito no projeto
- ✓ Escolha “New”
- ✓ Escolha “Directory”
- ✓ Digite o nome desejado para a pasta

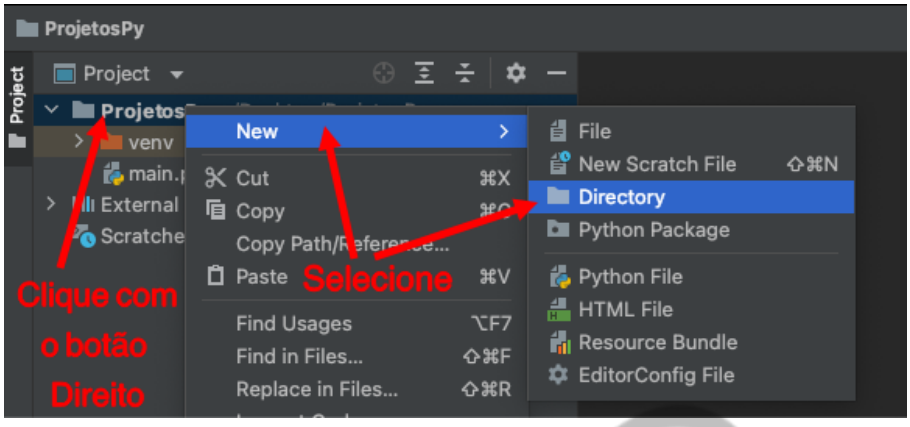

Figura 10 – Criando uma pasta para o projeto no Pycharm (1)
Fonte: Elaborada pelo autor (2022)

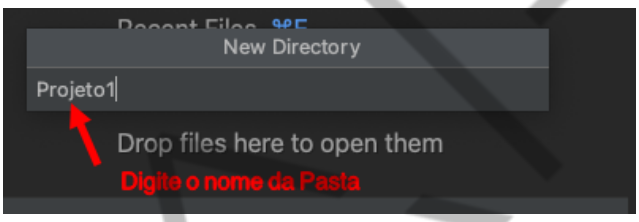

Figura 11 – Criando uma pasta para o projeto no Pycharm (2)
Fonte: Elaborada pelo autor (2022)

Crie um arquivo para o seu projeto:

- ✓ Clique com o Botão direito na pasta “Projeto1”
- ✓ Escolha “New”
- ✓ Escolha “File”
- ✓ Digite o nome do arquivo

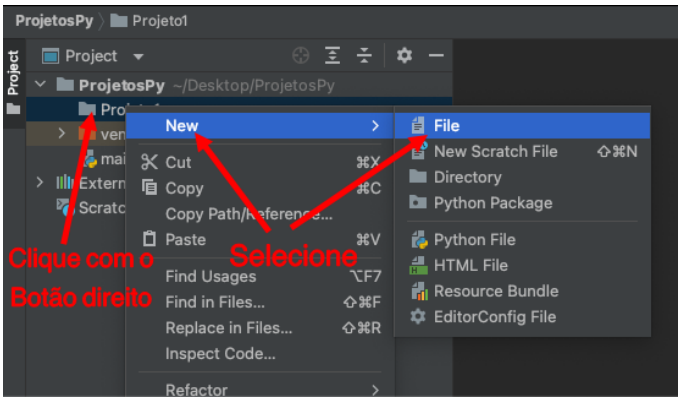

Figura 12 – Criando um arquivo para o projeto no Pycharm (1)
Fonte: Elaborada pelo autor (2022)

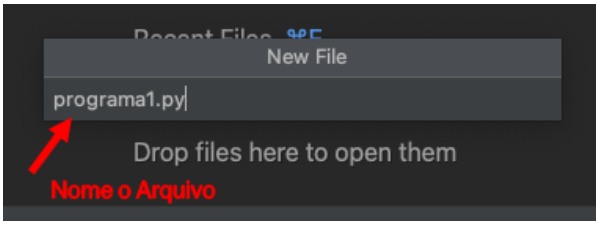

Figura 13 – Criando um arquivo para o projeto no Pycharm (2)
Fonte: Elaborada pelo autor (2022)

- ✓ Pasta “Projeto1”
- ✓ Arquivo “programa1.py”
- ✓ Edição do Primeiro programa (comando exemplo)

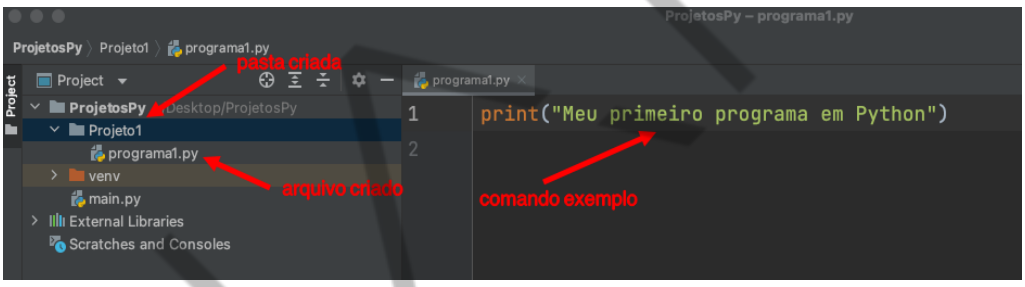

Figura 14 – Arquivo programa1.py
Fonte: Elaborada pelo autor (2022)

- ✓ Clique com o botão direito em “programa1.py”
- ✓ Para executar o programa escolha “Run programa1”

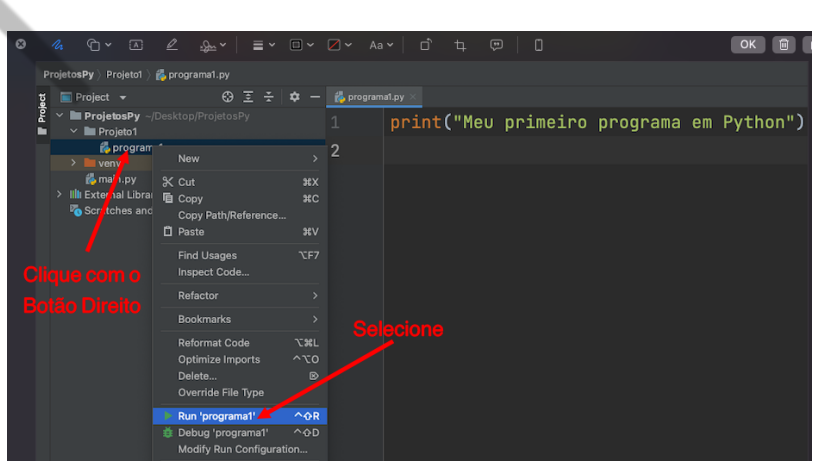

Figura 15 – Executando “Run programa1”
Fonte: Elaborada pelo autor (2022)

Execução do Programa:

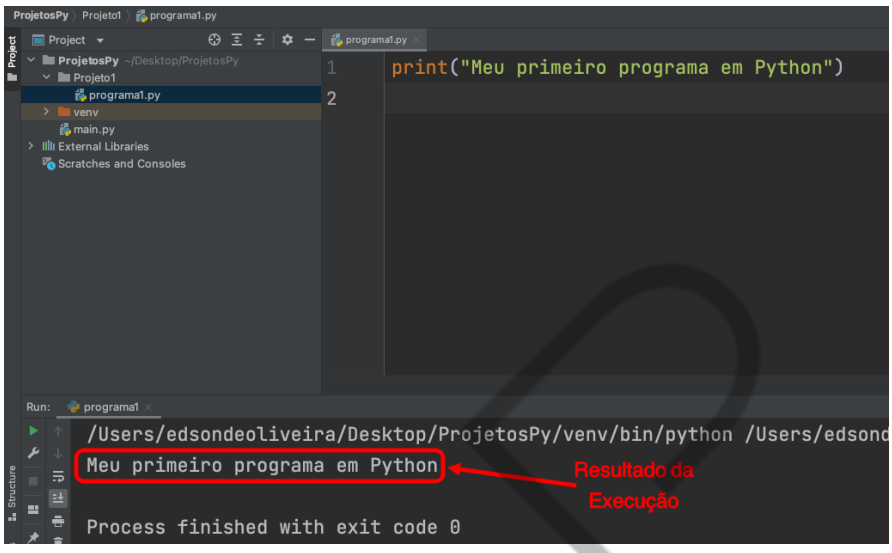

Figura 16 – Execução do programa
Fonte: Elaborada pelo autor (2022)

ANEXO II – INSTALAÇÃO E EXECUÇÃO DO JAVA

Para utilizarmos o Java precisamos do compilador e uma IDE. A IDE que utilizaremos é o Eclipse. Seguem algumas características do Eclipse:

- É um IDE gratuito e open-source.
- Originalmente criado pela IBM, hoje é mantido pela Eclipse Foundation.
- Além de possuir uma grande quantidade de ferramentas, ele é bem simples de usar e roda em qualquer sistema operacional, por ser escrito em Java.
- Um dos recursos mais importantes do Eclipse é a interface de plugins que permitem sua utilização com diversas linguagens além de Java, tais como C/C++, Python, PHP e inclusive para a plataforma Android.

Siga os seguintes passos para efetuar a instalação:

- Acesse o site https://www.eclipse.org/downloads/
- Escolha o Sistema Operacional
- Faça o download e

- Conclua a instalação:

Figura 17 – Download Eclipse IDE 2021-12
Fonte: Elaborada pelo autor (2022)

Criando o primeiro projeto em Java:

- Ao abrir o eclipse aparecerá a seguinte tela:

Escolha o diretório (workspace) onde seus projetos serão criados:

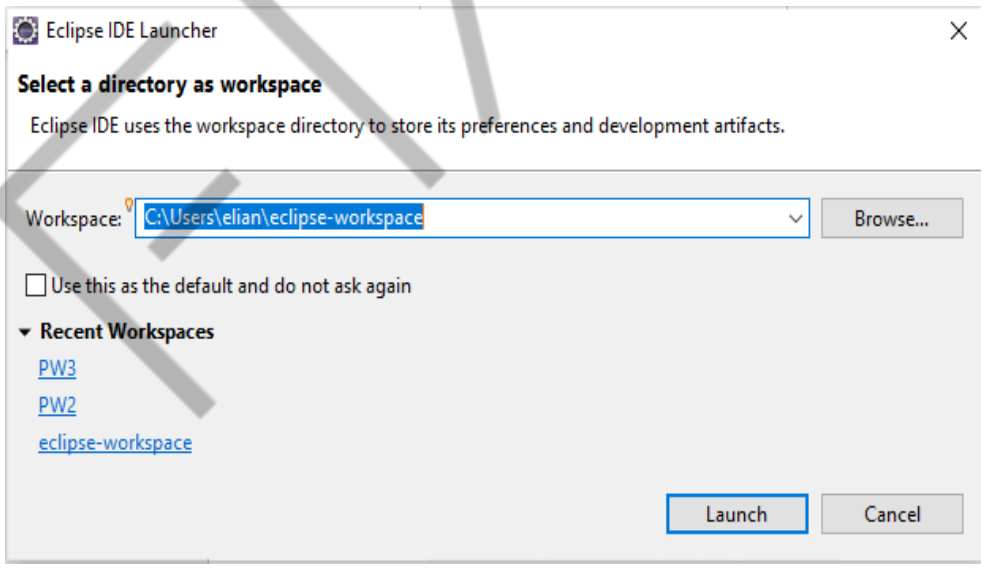

Figura 18 – Escolha de criação de diretórios para os projetos
Fonte: Elaborada pelo autor (2022)

Para criar um novo projeto clique em File → New → Java Project

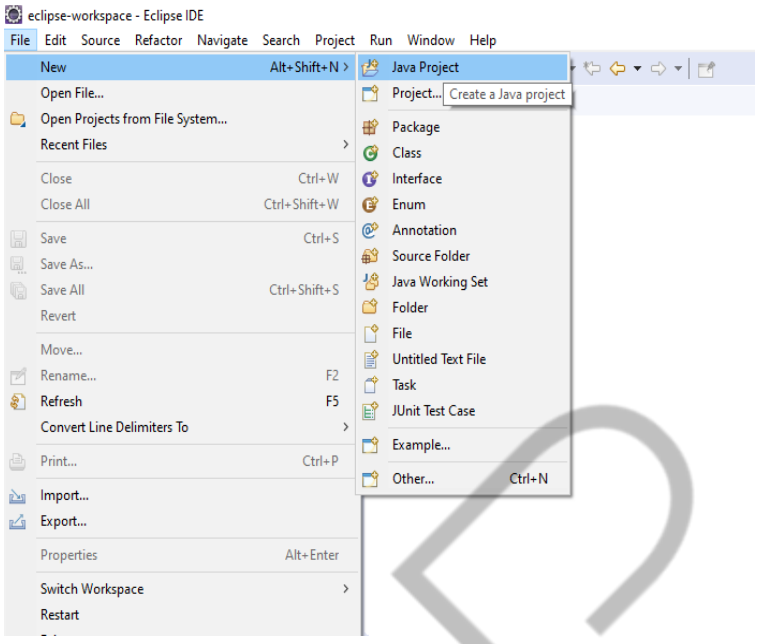

Figura 19 – Criação de novo projeto em Java (1)
Fonte: Elaborada pelo autor (2022)

Insira o nome do projeto e clique em Finish:

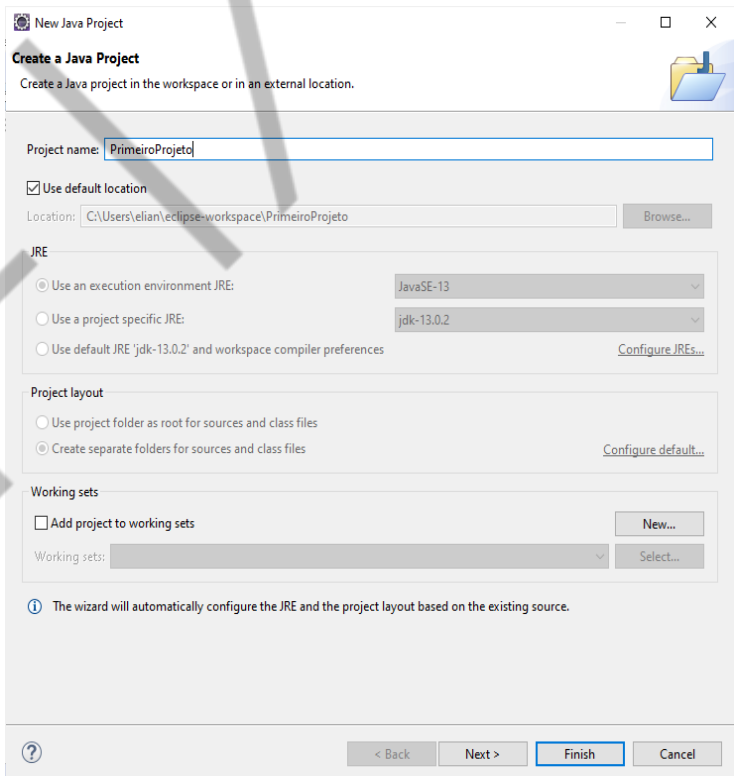

Figura 20 – Criação de novo projeto em Java (2)
Fonte: Elaborada pelo autor (2022)

Para criar uma classe:

- Clique com o botão direito do mouse em New / Class:

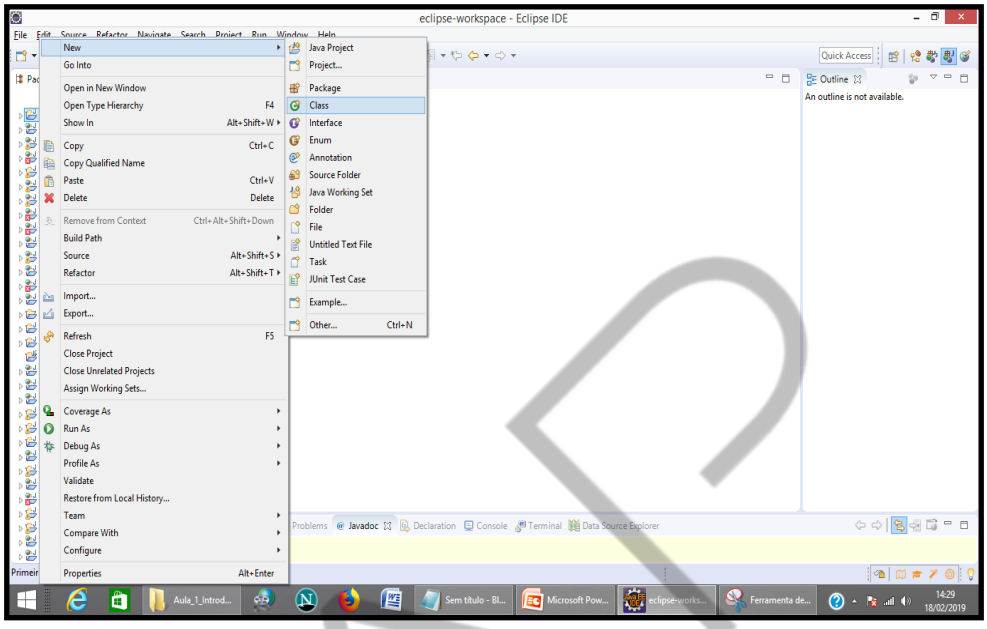

Figura 21 – Criação de nova classe em Java (1)
Fonte: Elaborada pelo autor (2022)

- Colocar o nome da Classe no campo name
- Flegar o campo: “public static void (String[] args)”
- Clicar em Finish

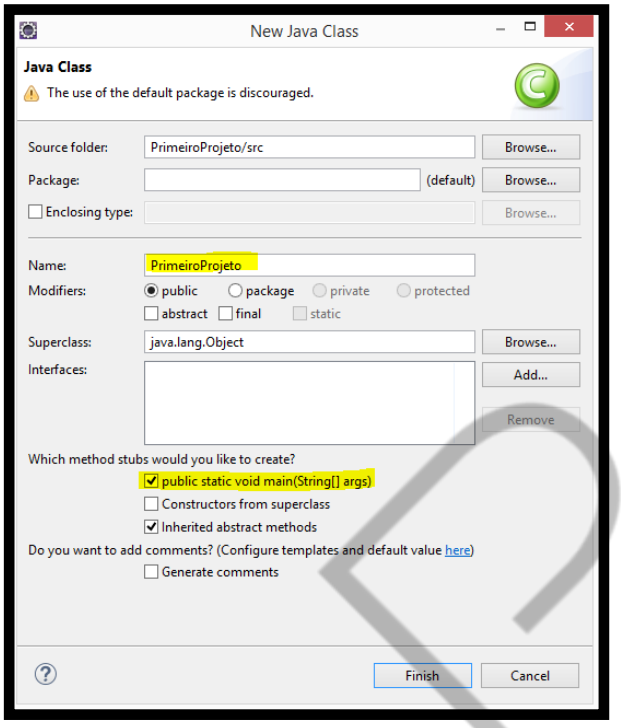

Figura 22 – Criação de nova classe em Java (2)
Fonte: Elaborada pelo autor (2022)

Digitar o código abaixo, conforme o lado direito da imagem:

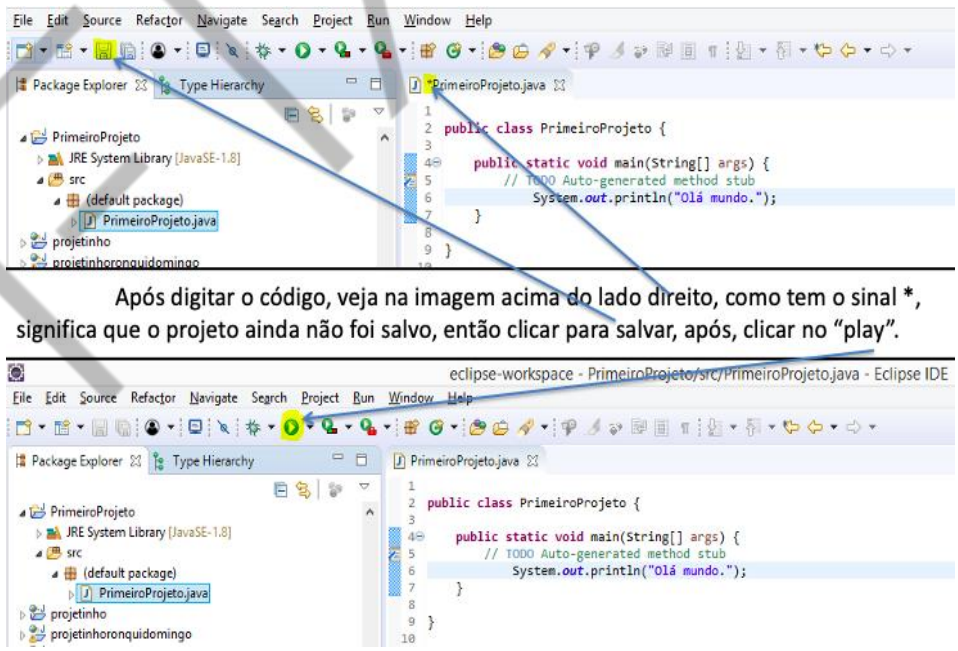

Figura 23 – Criação de nova classe em Java (3)
Fonte: Elaborada pelo autor (2022)

Após clicar no "Play", irá exibir a frase "Olá mundo", que está abaixo em amarelo:

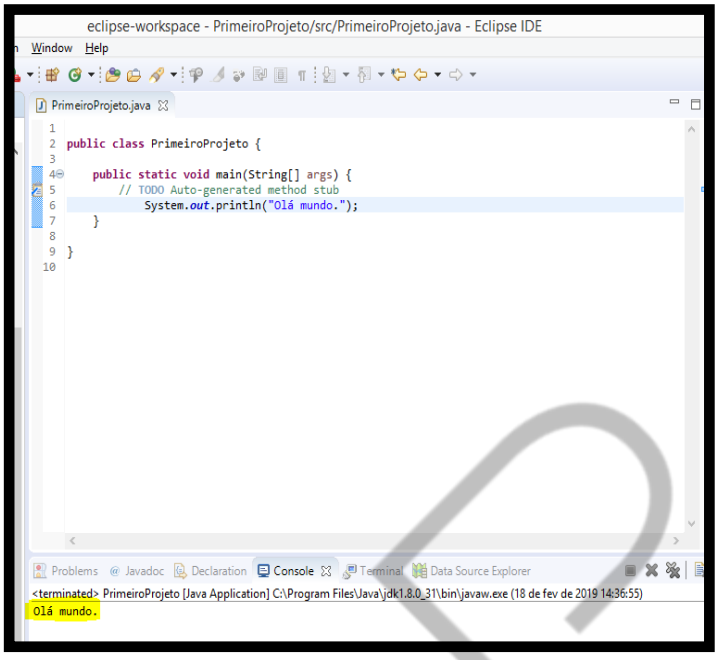

Figura 24 – Criação de nova classe em Java (4)
Fonte: Elaborada pelo autor (2022)

GLOSSÁRIO

Sintaxe	Regras e formatos para utilização de uma instrução.
Código-fonte	Arquivo no formato texto, com a extensão solicitada pela linguagem de programação, onde escrevemos o algoritmo numa linguagem de programação.
Comentário	É um texto inserido dentro do código-fonte que é ignorado pelo compilador ou interpretador. Ele serve como auxílio de entendimento da rotina ou para documentação. Em Python, iniciamos um comentário com #. Em Java (ou pseudocódigo) iniciamos o comentário de uma linha com // ou delimitamos um comentário de múltiplas linhas com /* e */
Casting	É o método de conversão de tipos de variáveis para outros tipos.