**INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO**

**WEEK 1 AND 2**

Bem-vindos ao curso de Introdução à Computação. Nesta aula inicial, falaremos um pouco sobre o que é ciência da computação e quais são os objetivos do curso. Trata-se de um curso promovido pelo Departamento de Ciência da Computação do Instituto de Matemática e Estatística da Universidade de São Paulo e esperamos que você goste muito. A ciência da computação é uma nova ciência que surgiu em meados do século XX, abrange aspectos de várias áreas do conhecimento humano. Matemática, podemos dizer que é a base, a base inicial da ciência da computação. Vários matemáticos começaram a trabalhar em diferentes aspectos computacionais que levaram então, em meados do século XX, ao que chamamos de ciência da computação. Então haverá muita matemática e ciência da computação. Mas não só isso. A ciência da computação também tem muitos aspectos de engenharia, porque o que os cientistas da computação fazem é construir sistemas de computador complexos envolvendo hardware e principalmente software. Então esses sistemas de software são sofisticados, complexos e às vezes feitos de milhões de linhas de código, e precisamos de muitas técnicas de engenharia para chegar lá. Mas também há aspectos das ciências naturais, na química, na física, na biologia, os cientistas desenvolveram uma metodologia científica onde elaboramos hipóteses e então fazemos experimentos para verificar se essas hipóteses eram verdadeiras ou não. A partir daí desenvolvemos teorias e com as teorias tentamos prever novos comportamentos. E ciência da computação, nós fazemos tudo isso também, há um monte desta parte experimental, também, ciência da computação. Além disso, a ciência da computação tem aspectos mais estéticos. O que podemos dizer, aspectos criativos que são até semelhantes ao que um artista faz. Ao escrever código, este é um ato tremendamente criativo onde o trabalhador do conhecimento produzirá novas ideias que são codificadas na forma de software. Então, tem esse aspecto estético e criativo muito forte também. Mas podemos até dizer que ele tem até algo de esporte, porque você só vai aprender a ser um cientista da computação se você praticar muito se você fizer um monte de exercícios e se você manter essa prática. Então, um bom desenvolvedor de software, um bom programador, ele tem que programar muito para desenvolver, entre aspas, os músculos do cérebro para que ele seja fluente nessa linguagem, que é a linguagem da ciência da computação. [MÚSICA] [MÚSICA] Quais são as habilidades mais importantes de um cientista da computação? Eu diria que a habilidade mais importante é a capacidade de resolver problemas do mundo real de forma computacional, computacionalmente. E basicamente o que veremos neste curso é exatamente isso, queremos desenvolver sua capacidade de resolver problemas reais através de um método computacional. Especificamente falando, das habilidades mais básicas e concretas que vamos querer desenvolver aqui, a primeira coisa é a capacidade de formular problemas do mundo real em termos computacionais. Então, você tem um problema que gostaria de resolver, podemos definir como computacionalmente eu posso modelar esse problema para que eu possa resolvê-lo computacionalmente. Uma vez que você modelou o problema, então vamos trabalhar uma solução para este problema em termos computacionais. Muitas vezes o que vamos fazer é implementar um algoritmo que resolve esse problema de forma genérica. Depois de implementar este algoritmo, depois de projetar o algoritmo, temos que escrever um programa, em uma linguagem de programação que o computador entende, que implementa este algoritmo para que possamos executar este algoritmo em um computador. Então, é essencial que testemos o programa para verificar se ele realmente resolve o problema corretamente, se esse algoritmo não tem nenhum erro e se a implementação do algoritmo também está correta. Então, ao longo deste curso vamos exercitar esses quatro aspectos aqui muito fortemente. O profissional de informática, em geral, também deve ter outras habilidades mais avançadas que você pode pensar em desenvolver ao longo de sua carreira. Particularmente, um profissional de TI, ele tem que ser capaz de gerenciar grandes softwares, compostos por muitos programas e, às vezes, cada um desses programas pode ter milhares ou milhões de linhas de código. Portanto, essa capacidade de gerar grandes quantidades de software é uma habilidade importante para os profissionais dessa área. Além disso, hoje em dia, cada vez mais, temos que construir software para lidar com grandes quantidades de dados. E há várias vertentes da ciência da computação que estão surgindo que lidam com big data, mineração de dados, aprendizado de máquina que lidam com essas enormes quantidades de dados.

Outro aspecto importante são as pessoas, lidando com as pessoas porque o software hoje em dia quase nunca é desenvolvido por pessoas isoladas, mas por grandes equipes trabalhando juntas de forma colaborativa, seja no mesmo ambiente físico, ou distribuídas pela internet. Portanto, gerenciar essas equipes de desenvolvimento de software também é uma habilidade muito importante de um profissional de tecnologia da informação hoje em dia. Por fim, comunique-se com clientes e usuários para entender seus problemas, dificuldades e necessidades. Então, o bom profissional de informática, ele não sabe como se comunicar apenas com o computador, ele tem que saber como se comunicar com outras pessoas de uma maneira que ele faz, tem maior impacto. Neste curso especificamente, queremos desenvolver as seguintes habilidades. Queremos que você aprenda a formular perguntas, a pensar criativamente para encontrar uma solução para essa questão, para este problema, e para que você seja capaz de expressar a solução de forma clara e precisa, em particular, vamos expressar a solução em uma linguagem de programação, então um subproduto do curso é que você aprenderá a programar em uma linguagem de programação de alto nível. [MÚSICA] Esses programas que vamos escrever vão realmente implementar algoritmos. O que é algoritmo? Algoritmo é uma lista de instruções, passo a passo, para resolver um determinado problema, geralmente um problema do mundo real. Você pode pensar que é como uma receita de bolo que diz, passo a passo, primeiro pegue os ingredientes, depois misture dessa forma, depois coloque no fogo, depois esfrie-o e coloque no congelador e depois monte e sirva dessa forma. Então, a sequência passo a passo do que deve ser feito para resolver um determinado problema, este é um algoritmo. Então, simplificando, o que fazemos computacional é, analisar o problema, criar um algoritmo genérico que resolve esse problema, escrever um programa que implemente esse algoritmo, e então testar o programa para verificar se ele realmente funciona, que está tudo bem como gostaríamos. Então é basicamente isso que vamos fazer. Por isso, precisamos aprender linguagens de programação, saber programar uma linguagem de programação. Uma linguagem de programação é uma linguagem formal e precisa cujas instruções podem ser executadas por um computador. Se pensarmos em hardware de computador, na verdade o chip lá, ele só entende uma linguagem de baixo nível, que chamamos de linguagem de máquina, que na verdade é uma sequência de zeros e um código binário enorme, isso é o que o hardware entende. Mas esta é uma linguagem muito difícil de programar, então depois de alguns anos que os primeiros computadores apareceram, linguagens mais sofisticadas apareceram, por exemplo, linguagem de montagem, linguagem de montagem, em vez de uma sequência de bits você tem o comando adicionar para a soma, por isso é um pouco mais fácil de lembrar. Mas mais tarde, apareceram idiomas, que chamamos de idiomas de alto nível, que têm comandos um pouco mais semelhantes à linguagem natural, onde os comandos são palavras, por exemplo, em inglês ou português. Linguagens como python, é uma linguagem que usa este modelo de programação, por isso dizemos que é uma linguagem de alto nível. E há muitos outros, Java, Ruby, C, C++, Java Script e muitos outros são linguagens de alto nível que são apropriadas para o desenvolvimento de sistemas de software hoje. Como essas línguas são entendidas pelo computador? Primeiro, existem dois tipos diferentes de idiomas. As línguas interpretadas funcionam da seguinte forma. Você tem código-fonte, que é um arquivo de texto onde você tem certos comandos neste idioma de alto nível e esse arquivo de origem é dado como a entrada para outro programa chamado um intérprete. Este intérprete lê as linhas do arquivo de origem, uma de cada vez, e para cada linha, interpreta o que essa linha significa, e executa algo internamente no computador, que provavelmente vai gerar alguma saída na tela, ou algum outro tipo de saída, vai imprimir algo, ele vai enviar alguns bytes pela internet, mas ele vai fazer alguma coisa. Então essas são as línguas interpretadas. A língua Lisp é um exemplo de uma linguagem interpretada. Há outra família que são línguas compiladas. Por exemplo, a linguagem C, C++ são idiomas compilados. Como funciona lá? Temos o código-fonte, um novo arquivo de texto com o programa, e entregamos este arquivo de texto para o programa que chama o compilador. Este compilador transforma este programa em código de máquina, que são aqueles zeros e uns, que chamamos de código de objeto. E então, este objeto que escrevemos em nosso disco como um arquivo executável e, em seguida, este arquivo executável, mais tarde, depois que a compilação é concluída, ele pode ser executado e terá um programa executável, tipicamente, o próprio sistema operacional.

O sistema operacional executará este arquivo executável que irá gerar uma determinada saída, fará alguma interação com o usuário. Então essas são as línguas compiladas. Muitas línguas modernas usam ambas as abordagens em combinação para tirar proveito de ambas as abordagens. Assim, por exemplo, linguagens como Java e Python são interpretadas e compiladas linguagens. Então, na primeira etapa, antes de iniciar a execução do programa, o compilador traduz o código-fonte, que é aquele código que digitamos, em código byte, que chamamos de código byte, que é uma representação condensada desse mesmo programa, só que mais fácil para o computador interpretar isso. E então, nesta segunda etapa, este código de byte, este arquivo pré-compilado, é submetido ao intérprete, que então lê os códigos byte de forma muito eficiente e executa os comandos correspondentes. [MÚSICA] Por onde começar? Então, nós já tivemos uma visão geral do que este curso vai fazer e como os programas podem ser escritos e executados. Mas vamos começar a aprender devagar, certo? Então, uma boa maneira de começar a aprender computação é aprender a escrever códigos simples e muito simples em uma linguagem de alto nível, onde escolheremos uma linguagem de alto nível, executaremos esses pequenos programas, teremos alguma prática interagindo com os programas e, em seguida, gradualmente aprender novos conceitos, conceitos de cada vez, e aumentar progressivamente a sofisticação e o tamanho desses programas. Então, esperamos que até o final deste curso você esteja desenvolvendo programas muito interessantes, em uma linguagem de programação de alto nível.

**SEGUNDO DIA**

**1.**

**Pergunta 1**

Sobre o profissional de Ciência da Computação pode-se afirmar que:

**1 / 1 ponto**



precisa desenvolver e estimular sua capacidade de resolver problemas reais.



utiliza muito o raciocínio lógico matemático, portanto pouco precisa de habilidades de relacionamento.



é uma carreira em que basta a graduação, pois dificilmente algo muda.



não possui nenhuma similaridade com artistas, em termos de criatividade.

**Correto**

**2.**

**Pergunta 2**

\_\_\_\_\_\_\_\_\_\_\_\_ são instruções para o computador. (escolha todas as opções corretas)

**0.75 / 1 ponto**



Software



Programas

**Correto**



Hardware



Teclado

Você não selecionou todas as respostas corretas

**3.**

**Pergunta 3**

Um profissional de Ciência da Computação deve se preparar para desenvolver várias habilidades, exceto:

**1 / 1 ponto**



gerenciar software de grande porte composto por muitos programas.



comunicar-se com clientes e usuários para entender problemas, dificuldades e necessidades.



gerenciar equipes de desenvolvimento de software.



gerenciar projetos de hardware de novos computadores.

**Correto**

**4.**

**Pergunta 4**

O hardware dos computadores pode executar diretamente programas em:

**0 / 1 ponto**



linguagem de alto nível.



linguagem de montagem.



linguagem de máquina.



linguagem natural.

**Incorreto**

Você não selecionou uma resposta.

**5.**

**Pergunta 5**

\_\_\_\_\_\_\_\_\_ é responsável por traduzir programas escritos em linguagem de alto nível para linguagem de máquina.

**1 / 1 ponto**



O compilador.



Uma CPU.



O sistema operacional.



O montador.

**Correto**

**6.**

**Pergunta 6**

As metodologias modernas de desenvolvimento de software dizem que devemos testar os programas que escrevemos a todo o momento.

Por outro lado, nos métodos mais convencionais utilizados tradicionalmente, as etapas usadas para construção de um programa, em ordem, são:

**1 / 1 ponto**



desenvolver, testar, analisar.



analisar, desenvolver, testar.



desenvolver, analisar, testar.



analisar, testar, desenvolver.

**Correto**

**7.**

**Pergunta 7**

Um programa que seja compilado seguirá o seguinte percurso:

**1 / 1 ponto**



código-fonte, código-objeto, compilador, executor, saída.



compilador, executor, código-fonte, código-objeto, saída.



código-objeto, código-fonte, compilador, executor, saída.



código-fonte, compilador, código-objeto, executor, saída.

**Correto**

**8.**

**Pergunta 8**

Como aprender a desenvolver software?

**1 / 1 ponto**



escrever códigos em linguagem de alto nível.



executar pequenos programas.



aprender novos conceitos de forma incremental.



aumentar a sofisticação e tamanho dos programas progressivamente.



**Todas as demais estão corretas.**

**Correto**

**INTRODUÇÃO PYTHON**

Então vamos agora escrever nossas primeiras linhas de código Python. Para fazermos isso, precisamos ter um intérprete Python instalado em nossa máquina. Computadores Mackintosh ou Linux muitas vezes já vêm com o intérprete Python implementado, então, por exemplo, se você vem aqui apenas no terminal, abrindo-o, você pode digitar python e ele vai entrar no seu interpretador python, caso contrário você precisa instalá-lo, baixá-lo da internet e instalá-lo. No Linus você pode abrir terminal, "iii" qualquer que seja e digitar python, no Mac aqui você abre o programa que chama terminal, no Windows o prompt de comando e então podemos digitar comandos interativamente. Então, digitando python aqui vemos que ele está executando uma versão 2.7.10 do Python que eu instalei recentemente na minha máquina e aqui já podemos tipor interativamente alguns comandos. Então a coisa mais simples sobre matemática que eu posso pensar é o que crianças muito pequenas já aprendem, eles aprendem que mais é igual a dois. Como podemos fazer mais como dois Pythons? Basta digitar mais e bater enter. Mais dois, vemos a resposta aqui que mais é igual a dois. Podemos fazer matemática mais sofisticada. 123874 mais 342. Então vemos o resultado aqui. Podemos fazer subtração, cinco menos oito é negativo três. Podemos fazer uma multiplicação, oito vezes três ou algo 124967 vezes 345 e vemos o resultado aqui. Essas são as quatro operações básicas. Também podemos concatenar expressões agrupando mais dessas operações, para que eu possa fazer algo como dois mais cinco vezes dois, o que é isso? Dá 12, mas note que aqui o intérprete Python, seguiu essa regra da matemática que a multiplicação tem precedência sobre a adição, então primeiro fez as cinco vezes duas e depois o resultado foi dez, ele adicionou aos dois, então deu 12. Se eu quisesse fazer o contrário, primeiro fazer a soma e depois fazer a multiplicação, então eu posso, como faríamos matemática, colocar entre parênteses os dois mais cinco e depois multiplicar por dois, então sim, ele fez dois mais cinco e deu sete vezes dois é 14. Matemática usamos parênteses, depois suportes quadrados, depois aparelhos. Linguagem de programação nós só usamos parênteses, mas compensação você pode ter quantos níveis de parênteses quiser, eu posso ter algo como três menos, parênteses abertos, quatro, vezes 20 mais parênteses abertos, três, menos três vezes sete. Então eu tenho que fechar os parênteses aqui na mesma quantidade, dois, três, quatro parênteses, ele faz toda essa matemática e dá -59. Python também temos um operador de energia, então, por exemplo, dois cubos são oito. Dez para o poder de seis dá lá 1000 000. Para o poder de 120 dá, eu estava errado, para o poder de 120 dá 1. Então temos todas essas operações que estão à nossa disposição e podemos agrupar as operações para fazer expressões matemáticas mais completas usando os parênteses. Outra coisa interessante que podemos fazer é comparar, podemos perguntar ao intérprete python: dois a menos de 20? E então ele responde: verdade, dois é menos de 20. Ou eu posso ver: dez é maior que cinco? Sim, dez é maior que cinco. Dez é maior que 100? Falso, dez não é maior que 100. Então poderemos fazer comparações e usar o resultado dessas comparações no meio de nossos programas. Eu poderia muito bem perguntar: cinco é menor ou igual a cinco? Verdade, cinco é menor ou igual a cinco. Porque cinco é igual a cinco, então cinco é menor ou igual a cinco. Dez são maiores ou iguais a 20? Falso, dez não é maior ou igual a 20. Eu também poderia querer fazer comparações, se duas coisas são iguais, então, por exemplo, é nove ao quadrado igual a 80 mais? Sim, é verdade porque nove ao quadrado é 81 e 80 mais é 81. Note que para fazer uma comparação aqui Python usamos um "iii" da linguagem C que são dois sinais iguais. Então: == é quando queremos comparar se dois valores são iguais, então 100 é igual a dez vezes dez? Sim, são iguais. Agora 100 é igual a dez vezes mais? Não, é falso, não é a mesma coisa. Posso também querer comparar se duas coisas, se uma é diferente da outra, então, por exemplo, é 20 diferente de 30? E note, o sinal diferente é! =. Ponto de exclamação matemática às vezes significa não, então não é igual é o símbolo de diferente. 20 é diferente de 30? Sim, 20 é diferente de 30. Agora, 20 quadrados são diferentes de 400? Falso, não é diferente de 400 porque é igual a 400. Assim, com este tipo de operação de comparação podemos comparar valores diferentes, e particular para cada um dos valores pode ser o resultado de uma expressão aritmética complexa envolvendo as quatro operações, particular eu poderia fazer dez divididos por dois e ver o resultado, são cinco. Não, dez divididos por dois não é diferente de cinco, porque dez divididos por dois é igual a cinco. Então, eu gostaria que você agora entrasse no interpretador Python em seu computador e praticasse todas essas operações, praticasse expressões mais complexas. E comparação entre diferentes expressões aritméticas. Nós só aprendemos computação praticando, você não vai aprender apenas assistindo vídeo. Então, entre no seu intérprete Python e pratique muito do que vimos nesta aula.

### 1.

**Pergunta 1**

O que o interpretador de Python responderá se você digitar: 2 + 5 \* 3

**1 / 1 ponto**



30



13



17



21

**Correto**

### 2.

**Pergunta 2**

Considerando a versão 3.x do Python, qual é o resultado da expressão: 20 / 3

**1 / 1 ponto**



7



6



6.666666666667



0

**Correto**

### 3.

**Pergunta 3**

Qual é o resultado de: 20.0 / 3

**1 / 1 ponto**



0



7



6.666666666667



6

**Correto**

### 4.

**Pergunta 4**

O que o interpretador de Python responderá se você digitar o seguinte comando?

10 > 10

**1 / 1 ponto**



Incompatibilidade de tipo



Falso



Erro de sintaxe



Verdadeiro

**Correto**

### 5.

**Pergunta 5**

O que o interpretador de Python responderá se você digitar o seguinte comando?

9 \*\* 2 == 80 + 1

**1 / 1 ponto**



Falso



Incompatibilidade de tipo



Erro de sintaxe



Verdadeiro

**Correto**

### 6.

**Pergunta 6**

Qual o resultado gerado pelo comando ((5 - 3) \*\* 3)?

**1 / 1 ponto**



-8



22



8



-22

**Correto**

### 7.

**Pergunta 7**

Como devo escrever a seguinte expressão para que o interpretador de Python a calcule corretamente?

{ 4 x [ ( 33 - 31 ) x 2 ] }

**1 / 1 ponto**



( 4 x ( ( ( 33 - 31 ) x 2 ) )



( 4 \* ( 33 - 31 \* 2 ) )



{ 4 \* [ ( 33 - 31 ) \* 2 ] }



( 4 \* ( ( 33 - 31 ) \* 2 ) )

**Correto**

**VARIÁVEIS E PRIMEIRO PROGRAMA**

Olá, na aula de hoje vamos aprender um importante conceito de computação, que são variáveis. Vamos ver como definir uma variável python e também para que elas são. Para usar o ambiente python muito moderno na classe de hoje. Eu vim aqui no python.org e baixei esta versão daqui. No meu caso, a versão Python três ponto cinco ponto. Ele tem disponível para todos os sistemas operacionais, para Linux, para Macintosh, para Windows. Você pode baixar para o seu sistema operacional e instalar. E então, além do intérprete python, ele também vai instalar o ambiente de desenvolvimento python, que é este ocioso. Então eu vou ficar ocioso aqui e ele abre esta janela. Olha, já está dizendo que é o três ponto cinco ponto um. versão de python. E aqui eu posso digitar comandos python. Oops! Dez mais 45, por exemplo, certo? Mas o tema de hoje são variáveis. Então, o que é uma variável? Variável é um pouco de memória onde podemos armazenar valores e esse pedaço de memória tem um nome, que é o nome da variável. Então, por exemplo, eu posso dizer x recebe cinco. O que estou dizendo ao intérprete python? Estou falando pegar o número cinco e armazená-lo na variável x. Ou seja, guarde-o na parte de memória onde estes cinco serão armazenados que eu vou ser capaz de usá-lo mais tarde. Então, se eu perguntar qual é o valor, qual é o valor de x e ele vai dizer que o valor de x é cinco. Eu posso dizer algo como, y recebe dez vezes 32 mais x. Então ele está fazendo este cálculo, ele fez dez vezes 32 é 320. 320 ele adicionou ao valor de x, que x naquele momento valia cinco. E o resultado aqui foi 325. Observe que uma variável pode alterar seu valor ao longo do tempo. Se uma variável não pudesse variar, não seria chamada de variável! Então uma variável, se a variável x vale cinco, nada me impede de fazê-lo: x fica 345, então estou pedindo para armazenar o valor 345 em x e então notar que o valor se torna 345. Que cinco que era o valor anterior desaparece, é apagado. E python escreve para essa parte de memória que tinha cinco; Não há mais cinco, agora há 345. Então, em particular, eu posso fazer x obter x mais dez. Então, o que está acontecendo aqui? Está pegando o antigo valor de x, que era 345, adicionando dez dá 355, então agora o novo valor de x é 355, então as variáveis variam de valor ao longo do tempo, como o programa corre e modifica essas variáveis, hein? Eu posso fazer algo como, soma, obter x mais y. Então aqui estou eu dizendo ao intérprete python, calcule essa expressão aritmética que é x mais y e o resultado dessa expressão aritmética armazene outra variável chamada soma, então eu posso pedir para ver o valor da soma; o valor da soma é 680. Há outra maneira de fazer isso. Podemos pedir para imprimir, imprimir lá o valor da soma; ele vai imprimir 680 lá. Portanto, essas são variáveis, mas note que até agora só armazenamos inteiros nessas variáveis, mas podemos armazenar outras coisas também, um tipo particular de dados que podemos ter em uma linguagem de programação é um tipo de string que é texto, é string, uma string; para que eu possa ter algo assim, uma variável chamada frase. Frase recebe e, em seguida, entre aspas eu coloco algo assim: a massa ainda vai comer do biscoito fino, biscoito que eu faço. Então este aqui é um texto que foi armazenado nesta variável chamada frase. Se eu pedir o valor da variável frase, o texto que foi armazenado aparecerá lá. Assim, podemos ter variáveis que possuem diferentes tipos de dados lá. Que outros exemplos? Uma coisa interessante é sempre o nome da variável diz algo sobre o que essa variável significa, por exemplo eu posso dizer que o peso é igual a 78 e posso dizer que a altura em centímetros é de 83. Em seguida, observe que o nome da variável está explicando o significado dessa variável. Portanto, não é interessante ter nomes de variáveis muito curtos que não explicam o significado dessa variável. Até agora, todos os comandos python que digitamos, digitamos uma única linha e imediatamente vemos o resultado dessa linha. Mas alguns casos não são o que queremos fazer. Em alguns casos queremos escrever um programa Python, uma série de comandos que serão todos executados em sequência ao mesmo tempo. Como poderíamos fazer isso? Para fazer isso, não vou digitar aqui dentro do intérprete. Vou abrir outro editor de texto. ele pode ser qualquer editor de texto que você gosta, e ele tem que ser capaz de escrever arquivos em formato de texto simples como .txt, ou algo assim, texto simples sem qualquer formatação. Então eu abri aqui editor de texto que eu gosto que é editor de software livre, mas você pode usar o seu próprio; e aqui eu posso digitar programa; por exemplo eu posso fazer um recebe dez, b recebe 20; e então eu vou enviá-lo para imprimir: a soma dos números é. E eu quero imprimir o a plus b. Eu já sei. Vou armazenar primeiro em uma soma variável que vai receber um mais b, porque a soma recebe um mais b, e eu quero imprimir o valor dessa soma. Então agora este é o meu programa; e meu programa está pronto agora eu quero executá-lo para ver se eu cometi algum erro, ou se o programa está correto. Como é que eu faço? Primeiro eu salvo. Então eu vou salvá-lo lá em um lugar, em um diretório, em uma pasta que é fácil para mim encontrar e eu vou e nomeá-lo. Vou chamá-lo de adder, este meu programa vai chamar de adder. E a extensão será .py, em vez de chamá-lo de txt porque é um arquivo de texto, eu vou torná-lo .py para deixar claro que é um programa Python, ok? Então eu escrevi lá .py. Pronto meu programa está agora armazenado, está funcionando lá. Então eu venho aqui no meu terminal. Primeiro eu tenho que ir para aquele diretório meu, onde eu guardei o, Aqui, que, feito, eu já estou no lugar certo, onde eu salvei meu arquivo. E agora eu posso correr. Para me executar, em particular, no meu computador aqui eu tenho duas versões do Python. Tenho python versão dois e Python versão três, que é que três ponto cinco que eu mostrei aqui que eu instalei. Então eu posso executar qualquer uma das versões, eu sempre gosto de executar a versão mais recente e no meu computador o mais recente eu coloquei python nome três. É bastante comum para nós colocar este nome Python três para executar Python versão três. Eu simplesmente coloquei Python três. E o nome do arquivo de texto que gravei, que no meu caso foi summator.py. E eu pressiono enter e ele vai executar todas as linhas desse arquivo de texto. Nesse caso, veio o resultado, a soma dos números é soma. Opa, algo está errado. Eu gostaria que tivesse aparecido aqui não soma, mas a soma dos números. Então, qual foi o meu erro? Meu erro foi colocar essa soma aqui dentro das citações, se eu quiser imprimir o valor da variável então não tinha que ser algo entre aspas, eu teria que colocá-lo aqui depois, fora das cotações. Então, como é com a impressão, podemos imprimir várias coisas, separadas por írgulas. Então, se algo está entre aspas, então isso é uma string e vai imprimir exatamente o que está entre aspas, se não estiver entre aspas, então ele vai assumir que esse é o valor de algo, o valor de uma variável e assim por diante ele vai imprimir o valor de uma variável. Então vamos ver se eu consegui consertá-lo, eu vou salvar o arquivo e voltar aqui para o meu terminal e digitar a mesma coisa novamente: python três adder.py e agora é: a soma dos números é 30. Então funcionou. Acabamos de executar nosso primeiro programa Python. Então este é um programa Python e quando você chama dessa forma, o computador executa todas as linhas de uma só vez e dá a saída. Então, a partir de agora você tem duas opções para sempre executar o seu programa Python. Ou você pode ir aqui em um ambiente ocioso e tipo interativa de comando por comando e ver o resultado; este é o intérprete interativo. Ou você pode escrever seu editor favorito em um arquivo de texto, salvá-lo em um determinado diretório lá no seu computador; e, em seguida, venha aqui no terminal e digite Python e o nome do seu arquivo; ou Python três e o nome do seu arquivo para executar nesse programa como este. Então eu quero que você pratique isso agora em seu computador; primeiro, mexendo um pouco lá em ocioso e, em seguida, criando alguns pequenos programas que fazem alguns cálculos interessantes e imprimem a saída disso.

### 1.

**Pergunta 1**

A extensão de um arquivo fonte escrito em Python normalmente costuma ser:

**1 / 1 ponto**



.py



.exe



.obj



.prg

**Correto**

### 2.

**Pergunta 2**

Qual será o resultado desta sequência de comandos?

x = 10

y = 15

x + y

**1 / 1 ponto**



25



10 + 15



Verdadeiro



Erro de sintaxe

**Correto**

### 3.

**Pergunta 3**

Qual será exatamente a resposta do interpretador para o código abaixo?

soma = 5.5

impressão ("soma =", soma)

**1 / 1 ponto**



soma 5.5



5.5 5.5



5.55.5



soma = 5.5

**Correto**

### 4.

**Pergunta 4**

No Python, qual o valor final que estará armazenado em x?

x = 10

y = 30

x = x + 10

y = x + 10

x = x + y

**1 / 1 ponto**



50



60



40



70

**Correto**

### 5.

**Pergunta 5**

No Python, qual o valor final que estará armazenado em x e em y?

x = 50

y = 20

a = x

x = y

y = para

**1 / 1 ponto**



x terá o valor 20 e y terá 50



x terá 20 e y terá 20.



Nenhuma das alternativas



x tem 50 e y terá 50.



x terá o valor 50 e y terá 20

**Correto**

### 6.

**Pergunta 6**

Qual será a saída do seguinte código?

a = 1

b = 2

impressão (a + 2 \* b)

**1 / 1 ponto**



5



2



a + 2 \* b



1

**Correto**

### 7.

**Pergunta 7**

Suponha que x = 2, qual declaração é verdadeira? (True)

**1 / 1 ponto**



x == 8



x != 8



x > 2



x != 2

**Correto**

### 8.

**Pergunta 8**

Suponha que x = 1. Assinale **todas** as alternativas verdadeiras (True)

**1 / 1 ponto**



x >= 1

**Correto**



x < 1



x == 1

**Correto**



x != 1

**O QUE É SOFTWARE LIVRE**

Neste curso e na vida real, utilizamos muito software livre, que é uma forma muito interessante de desenvolver software e usar software, mas ao invés de falar sobre isso, trouxe aqui o gerente técnico do Centro de Software Livre e Competência do Ime Usp, que é um grande especialista e tem muita experiência com software livre e ele vai te dizer o que é software livre. Então eu vou chamar nelson dentro e enquanto isso eu vou fazer alguma magia aqui e desaparecer. Obrigado, Fabio. Espero que esteja gostando do curso. Então, como o Fábio comentou, vou falar um pouco sobre o que é essa história de software livre e talvez alguém de vocês tenha ouvido falar sobre isso, mas você realmente não sabe o que é. Software livre é um que é gratuito? O que encontrei na internet, o que não está amarrado em algum lugar? Vamos ver o que é. Primeiro vamos lembrar um pouco. Na primeira aula, Fábio falou um pouco sobre a ideia geral do que é computação e programação, e uma das coisas que ele disse foi que, que o trabalho que fazemos na área da computação geral é analisar um problema, criar uma ideia, algoritmo para resolver esse problema, e depois escrever um programa que realmente implementa esse algoritmo e testar para ver se funciona. Então, basicamente, o trabalho é entender o problema e representá-lo na forma de um programa em uma linguagem de programação. Agora, este programa que você cria quando você faz isso é uma coisa um pouco peculiar. Por que? Se você pensa, por exemplo, de um livro, um livro é algo que tem conhecimento embutido nele, essa sua ideia, tem arte, em suma, tem conhecimento lá. Esse conhecimento, pode ser sobre ferramentas, coisas que vão ajudá-lo a resolver problemas, mas é o conhecimento, não é a ferramenta. Se você pensar em uma ferramenta, na verdade, seja um martelo, uma máquina, um automóvel ou um programa de computador, essa ferramenta tem um uso. Direita? Mova você para frente e para trás ou martela alguma coisa. Mas, em geral, o conhecimento que é incorporado nesta ferramenta não é diretamente acessível na ferramenta em si, é o conhecimento que você consegue em outro lugar, exceto quando você está falando sobre programas de computador. No caso dos programas de computador, o programa acaba sendo ambos, tem dentro dele o conhecimento sobre o problema e a solução para esse problema que está lá, descrito na linguagem de programação. E, ao mesmo tempo, o programa é uma ferramenta para resolver o problema, quando você executa esse programa. Então, o programa é um pouco híbrido, diferente daqueles que estão acostumados a resolver problemas, ter conhecimento de um lado, com livros, tal, e a ferramenta do outro. E essa não é a única peculiaridade. Os programas também, eles podem ser muito facilmente copiados, o que não é o caso das máquinas, por exemplo, você tem uma segunda máquina, você já tem uma, você tem que construir a segunda. O programa não é, você simplesmente copia, é automático e praticamente não tem custo. E os programas podem ser facilmente modificados para incorporar melhorias ou resolver novos problemas. Pegar uma máquina e adaptá-la a um novo problema, ou fazer uma melhoria, é um processo muito mais complicado do que fazê-la em um programa de computador. Então o que acontece é que com essas duas coisas, onde ela pode ser facilmente copiada e facilmente modificada, você tem um fenômeno muito interessante, a mesma ferramenta, que é o programa, pode ser usado e adaptado por várias pessoas e o conhecimento que está incorporado nessa ferramenta que é o programa, pode ser acessado por todos, o conhecimento está lá, na linguagem de programação. Então, isso é muito interessante, mas tradicionalmente, do ponto de vista legal que temos hoje, os programas são protegidos por direitos autorais. O que isso significa? Basicamente, tratamos programas de forma metafórica, fingimos que são objetos que podemos comprar e vender. Em outras palavras, ignoramos essa vantagem de poder copiar e modificar de tal forma que o software tenha uma função de, em suma, interesses de mercado, a ideia era facilitar a circulação de dinheiro nesse meio para permitir a criação de novos produtos. Mas o resultado disso é que o uso dessa ferramenta acaba tendo restrições, você acaba não sendo capaz de usá-la para nada, você tem que pagar para poder usá-la, e em suma, os recursos que você tem nessa ferramenta acabam sendo limitados. E esse conhecimento que era público para o mundo inteiro, nem sempre é público para o mundo inteiro e nem sempre pode ser usado para o que você quer. Particularmente, quando o código fonte é secreto, então esse conhecimento é realmente obscurecido. E se eu trabalhar junto com esse meu amigo, vamos acabar trabalhando mais rápido, mais eficazmente para resolver problemas que, possivelmente, nós dois temos, mesmo que não seja o mesmo problema, é semelhante. Ele tem algumas variações sobre o seu assunto que não são exatamente o que eu preciso, eu tenho algumas coisas que não são exatamente o que ele precisa, mas nós dois juntos trabalhamos em uma solução comum que acaba sendo bom para nós dois. E o que é mais interessante: não precisa ser meu amigo, pode ser uma comunidade com várias pessoas, pode ser empresas junto com uma universidade, pode ser empresas concorrentes trabalhando juntas, podem ser pessoas trabalhando juntas na mesma sala ou trabalhando internacionalmente. Existem inúmeros projetos de software livre que envolvem várias instituições, algumas governamentais, algumas empresas, algumas pesquisas envolvendo várias pessoas, às vezes países diferentes. Existem vários exemplos, o Linux, por exemplo, é um exemplo que muitas vezes é citado, talvez seja o exemplo mais óbvio e tenha literalmente milhares de pessoas envolvidas no desenvolvimento em todo o mundo. Então, isso acaba gerando uma sinergia muito grande, e uma produtividade e qualidade muito altas, que são, precisamente, algumas vantagens muito fortes do software livre. Então, como você define o que é software livre? A definição tradicional é a definição da Free Software Foundation, é a Fundação para o Software Livre, que define que o software é gratuito se atender a quatro pré-requisitos, tem que ter quatro liberdades para o usuário. O usuário tem que ser capaz de executar o programa, ou seja, qualquer pessoa tem que ser capaz de executar o programa para qualquer propósito. Se tiver alguma restrição, seja no tipo de pessoa ou no tipo de propósito, não é mais software livre. A segunda liberdade é a liberdade de estudar e modificar o programa. Isso significa que eu tenho que ser capaz de acessar o código fonte. Em línguas que são compiladas eu não tenho acesso ao código fonte, não posso estudar, muito menos modificar como funciona. A terceira liberdade é que eu tenho a liberdade de redistribuir o programa, tenho o programa na mão, se tem alguém que precisa de uma cópia, eu vou lá e copio e não preciso de autorização, tudo já está resolvido, já está gratuito, essa autorização já está dada antecipadamente para eu poder redistribuí-la. E, finalmente, se eu fiz melhorias neste programa, eu tenho que ser capaz de redistribuir essas melhorias também. Então, o que eu criei para ser vantajoso para mim, acaba sendo vantajoso para os outros também. E é nisso que essa ideia de trabalhar de forma colaborativa é baseada, eu melhoro um pouco aqui, você melhora um pouco lá, esse outro cara melhora um pouco lá, todos nós juntos fazemos um produto que acaba sendo melhor para todos nós. Portanto, existem várias vantagens do software livre. Os termos comerciais têm uma redução no custo e risco de desenvolvimento. Em vez de ter uma empresa que tem que contratar toda uma equipe que vai fazer todo o investimento para resolver esse problema e desenvolver o produto, você tem uma empresa que está trabalhando com outras pessoas e talvez outras empresas nesse desenvolvimento. Assim, o risco de desenvolvimento e o custo do desenvolvimento se espalha entre essas diversas entidades de lá. Potencialmente, a qualidade do software livre é maior, porque se você tem mais desenvolvedores trabalhando, você tem mais usuários usando esse programa e, portanto, relatando bugs e você também tem o desenvolvedor que, quando ele coloca seu programa abertamente com o código fonte disponível para todos olharem, ele vai dizer algo assim: eu quero que essa coisa seja bonita, caso contrário, as pessoas vão pensar que eu sou um programador ruim. Essas coisas ajudam a qualidade do software livre a crescer. E para o usuário também é muito interessante, pois como você tem acesso ao código fonte, como você tem várias entidades, empresas, pessoas trabalhando com esse código fonte, você tem mais opções na hora de escolher o seu prestador de serviços, ser o cara que vai fazer a implementação para você, consultoria, treinamento e tal. E assim, às vezes pode parecer um pouco esotérico, uau, eu nunca tinha ouvido falar dele, mas na verdade, software livre é muito grande hoje em dia. Você tem um monte de material de software livre por aí, por exemplo, existem algumas linguagens de programação que são muito usadas, como java, é uma linguagem extremamente forte hoje no mundo dos negócios, ruby, muito usado em sistemas web, Python, é a linguagem que estamos usando neste curso, todos eles são software livre. São linguagens de boa qualidade, não só para produtos, mas também são bons termos de expressividade da língua, razão pela qual são usados, muitos deles, em cursos como este. E existem várias ferramentas que as pessoas estão acostumadas a usar que são softwares gratuitos, um exemplo muito fácil é o navegador firefox. É feito por uma ONG que coordena uma equipe de voluntários em todo o mundo e faz o programa firefox. O sistema operacional Android do celular, é praticamente todo software livre, é um projeto que é liderado pelo Google, mas que tem um cliente fora do Google que funciona, que colabora e que pega esse código e faz variações, então existem versões do Android diferente do Google, com outros nomes, [INAUDÍVEL] outros. E vários dos sistemas web que temos hoje são baseados em software livre que são muito tradicionais, como o servidor Web Apache, ele tem um banco de dados MySQL que é muito poderoso, tão poderoso que a empresa responsável pelo desenvolvimento do MySQL foi comprada pela Oracle, que é uma empresa especializada em bancos de dados. Todos esses produtos são software livre. Então, se você está interessado em saber um pouco mais sobre software livre, há muito na internet, talvez dois pontos principais, para começar, vamos colocar desta forma, é a Iniciativa de Código Aberto, que é um grupo de pessoas e empresas que estão interessadas em promover a ideia de software livre como uma coisa comercialmente interessante, e então eles dão toda a explicação de por que vale a pena termos de negócios para você trabalhar com software livre. Você tem a Free Software Foundation, que também faz todo movimento, ativismo no sentido de promover software livre lá fora. Mas focando mais na ideia de, olha, é importante que os usuários tenham liberdade, é importante ter colaboração e conhecimento aberto, e nós aqui no Centro de Competência e Software Livre temos nosso site, que está lá no slide, onde basicamente colocamos algumas atividades que fazemos, desde palestras até treinamentos, coisas assim, e falar um pouco sobre os projetos de pesquisa que estão acontecendo dentro do IME USP e relacionados ao software livre. Espero que ache esse assunto interessante e se envolva e saiba um pouco mais.

**Como dito, esse curso tem um livro-texto que você deve ler para consolidar seu aprendizado e que serve também de consulta. A leitura dessa semana é introdutória também:**

<https://panda.ime.usp.br/aulasPython/static/aulasPython/aula01.html>

O índice do livro completo se encontra aqui para referência:

<https://panda.ime.usp.br/aulasPython/static/aulasPython/index.html>

Olá, agora que aprendemos a construir expressões aritméticas, fazer comparações e lidar com variáveis Python, vamos aprender um pouco mais sobre valores e tipos de valor. Então, vamos abrir nosso intérprete ocioso e quando digitarmos algo como dez, dez é um número, mas eu posso perguntar python que tipo esse valor é, se eu der tipo dez aqui, ele vai dizer dez é valor da classe int, da classe inteiro. É definido, uma classe de valores que são números inteiros, então dez é número inteiro. Posso perguntar que tipo é. Vou colocar uma frase: tudo bem? Que tipo de "como você está?" "Tudo bem" é uma espécie de classe de cordas, este str significa corda. Eu posso ter objetos de tipo inteiro, eles são de classe int, objetos de corda tipo, que são de classe str. Particularmente, se eu fizer cinco divididos por dois, são dois e meio, 2,5, então qual é o tipo de cinco dividido por dois? O tipo de cinco dividido por dois é flutuar, a classe flutuante, esta palavra flutuar vem de ponto flutuante, que é uma representação interna que o computador usa para representar esses números fracionados onde há lugares após a círia. O número 2.5 é um número fracionado, dizemos que é um número flutuante. Uma coisa interessante, já que estamos falando de números de pontos flutuantes, é que às vezes queremos fazer uma divisão Python dando um resultado fracionado, por exemplo dez dividido por três, o que dá uma repetição decimal e note que neste caso é uma decimal repetindo, mas não tem infinitos lugares decimais. Os números em computadores modernos têm precisão limitada, eles não têm precisão infinita porque a memória do computador é finita e usamos um número finito de bytes para representar números. Aqui, vemos, há muitos lugares decimais, mas há um certo limite, particularmente o último lugar decimal está errado aqui porque é um erro de arredondamento, um erro de cálculo que é comum dar o último lugar decimal, ou os últimos lugares decimais. Mas este é um número flutuante, mas em alguns casos queremos fazer a divisão de inteiros, queremos saber qual é o resultado da divisão de inteiros e qual é o restante da divisão. Python podemos fazer isso usando o operador: //, então dez // três, então a divisão inteiro de dez por três é três, eu quero saber o que o restante da divisão, o restante da divisão nós usamos o operador por cento, dez por cento três, ele realmente vai dizer-lhe o que o restante da divisão de dez por três é realmente, e o restante da divisão é porque dez divididos por três é três e isso é sobra. Também temos essa opção ao fazer divisões para usar ou uma divisão que o resultado é de tipo flutuador, número de ponto flutuante, fracionado, ou o resultado é número inteiro, número de classe int. O que podemos fazer dado que temos esses pontos flutuantes? Por exemplo, eu posso ter variáveis que possuem valores de ponto flutuante. Então, posso dizer que o peso de uma pessoa é 78 e se a altura dessa pessoa é 1,83, nota, há uma variável que contém um valor inteiro e outra variável, essa variável é a altura, ela possui um número de classe flutuante, o valor desse número é 1,83. Agora eu posso fazer uma matemática, por exemplo, o IMC, o índice de massa corporal dessa pessoa vai ser peso dividido por altura ao quadrado. O IMC agora também é uma variável de ponto flutuante, fracionado, essa pessoa tem um IMC de 23,29, ele está bem ali. Uma coisa que podemos fazer é converter entre os tipos. Se temos uma IMC variável, ela tem um valor fracionado, de tipo flutuador, por alguma razão eu não quero ter os lugares depois da círia, eu só quero ter esse valor como um inteiro, por exemplo, para consultar uma tabela. Sabe o que posso fazer? Eu posso armazenar em outra variável IMC inteiro, eu vou fazer o seguinte, eu quero converter este valor flutuante em valor inteiro, int. Como faço isso? Eu escrevo int e, em seguida, entre parênteses, o valor que eu quero converter. Então, se eu fizer isso aqui: o inteiro IMC recebe int IMC, ele vai converter o IMC em um valor inteiro e armazená-lo na variável IMC inteiro. Então, qual é todo o valor do IMC? Tem 23 anos, em particular, ele trunca, ele joga fora o que está atrás da círia, independentemente do que está lá, ele joga fora e só leva a parte toda. Eu poderia fazer outros tipos de conversões. Por exemplo, há coisas que são de tipo string, nós já vimos. Então eu posso ter uma variável aqui, texto, que leva string: bom dia, como você está? O texto aqui diz: bom dia, como você está? Com o texto há certas coisas que eu posso fazer, por exemplo, eu posso perguntar qual é o comprimento desse texto. Há uma função aqui python chamada len, de comprimento, comprimento em inglês, que se eu perguntar qual é o comprimento, comprimento, do texto, ele vai me dizer: este texto tem 18 caracteres de comprimento. Se eu perguntar qual é o comprimento do IMC. Deu uma mensagem de erro, dizia: o objeto do tipo flutuar, ele não tem essa função chamada len aqui, não é possível saber qual é o comprimento de um número de tipo de flutuação. Mas lembre-se que eu disse que posso fazer conversão de tipo. Então eu poderia perguntar qual é o comprimento e, em seguida, pegar o IMC e converter em tipo de string. Como posso fazer isso? Vamos armazená-lo em uma variável temporária. Uma variável temporária recebe, quero converter meu texto em tipo de string. Fazendo isso, essa variável temporária, não eu fiquei confuso, na verdade não é o texto. Queremos fazê-lo na variável temporária, quero convertê-la em tipo de string, não faria sentido porque o texto já é uma string. Então, o que queremos converter? O IMC converterá então o IMC em string e, em seguida, qual é o valor? Note que agora ele tem o mesmo valor apenas entre apóstrofes, e os apóstrofes são para indicar que isso é uma corda, então isso não é mais um valor numérico, é uma sequência de personagens: personagem dois, personagem três, trilha sonora do personagem. Então, já que é uma corda, agora eu posso perguntar qual é o comprimento desta sequência temporária. E esse comprimento é 18. Na verdade, ele poderia fazer as duas operações acorrentadas, por exemplo, poderia perguntar qual o comprimento do IMC é convertido em string. Como faço isso? Converte-se em string, o IMC. Note, eu estou colocando uma função dentro da outra e então ele também dá 18, eu não preciso desta variável temporária. Então você vê que podemos então fazer a conversão entre diferentes tipos, já vimos pelo menos três tipos: tipo inteiro, tipo de flutuação para números fracionados e tipo de corda e podemos passar coisas de um tipo para outro. Então eu queria que você entrasse no intérprete e brincasse com esses vários tipos de variáveis e fazendo conversões entre os tipos.

## Tipos de Dados

**Total de pontos**6

### 1.

**Pergunta 1**

Qual operador é usado para divisão inteira?

**1 / 1 ponto**



\*\*



//



%



/

**Correto**

### 2.

**Pergunta 2**

Como armazenar em uma nova variável a quantidade total de caracteres das três variáveis abaixo?

cumprimento = "Olá"

nome = "Ana"

turno = "bom dia"

**1 / 1 ponto**



len (compliance) + len(nome) + len(turn)



print ("Olá Ana, bom dia!")



print (len(cumprimento + nome + turno))



tamanho = len(cumprimento) + len(nome) + len(turno)

**Correto**

### 3.

**Pergunta 3**

Qual a ordem de precedência em Python?

i) parênteses

ii) exponenciação

iii) divisão e multiplicação

iv) adição e subtração

**1 / 1 ponto**



i, ii, iii, iv



i, ii, iv, iii



ii, i, iii, iv



ii, i, iv, iii

**Correto**

### 4.

**Pergunta 4**

Qual o valor resultante da expressão (22 % 3)?

**1 / 1 ponto**



1



7.333333333333333



5



7

**Correto**

### 5.

**Pergunta 5**

Qual o resultado da expressão 2\*1\*\*2?

**1 / 1 ponto**



2



8



4



1

**Correto**

### 6.

**Pergunta 6**

Analise as duas afirmativas abaixo:

i - int(x) devolve o valor da variável x convertido para um inteiro (caso ela contenha um número).

ii - é possível somar strings com números inteiros diretamente

É correto dizer que:

**1 / 1 ponto**



as duas são verdadeiras



As duas são falsas



i é verdadeira e II é falsa



i é falsa e ii é verdadeira

**Correto**

Oi! Agora vamos aprender um conceito muito interessante, que é o conceito de entrada de dados. Até agora aprendemos a fazer alguns programas simples que geram uma certa saída de dados, mas não temos mecanismos para definir os dados que entram no programa. Vamos ver isso aqui agora. Vamos supor, comece com um problema do mundo real para nos motivar. Vamos supor que você foi para os Estados Unidos, você estava andando na rua e você viu que a temperatura era, no dia seguinte, ia ser de 85 graus. Então você se pergunta: "Está 85 graus quente, está frio?". Mas então você se lembra que aprendeu no ensino médio a converter entre as temperaturas. Então você pensa: "Qual será a fórmula de conversão?". Então você luta e você se lembra que a fórmula é Celsius mais de cinco é igual a Fahrenheit menos 32 dividido por nove, você se lembra disso. E então você se lembra das aulas python que você teve e dizer "Eu sei! Vou fazer um programa Python que converterá a temperatura de Fahrenheit para Celsius e saberei como fazê-lo". Então você vai lá, você abre seu editor de texto e começa a fazer um novo programa para implementar isso aqui. Então, vamos ver o que podemos fazer. A temperatura, digamos que é 78 Fahrenheit, quero fazer uma comparação com esse número 78. Então, a primeira coisa, eu vou armazená-lo em uma variável. Lembre-se que as variáveis devem ter um nome que indique seu significado. Então, um bom nome para esta variável seria temperatureFahrenheit. Então eu vou dizer: temperaturaFahrenheit vai armazenar o valor 78 porque é o valor da temperatura. E então eu vou querer calcular uma temperatura Celsius. Agora, qual será o valor desta temperatura Celsius? Vamos olhar aqui para nossa fórmula. Para isolar o Celsius do lado esquerdo, o que eu preciso fazer é pegar este cinco e multiplicá-lo lá em cima. Então minha fórmula seria: temperatura fahrenheit menos 32, multiplicada por cinco. Mas eu quero fazer a subtração primeiro, então é por isso que eu preciso desses parênteses aqui, e depois multiplicar por cinco e finalmente dividir por nove. Então, eu acho que minha fórmula aqui já está fazendo o que eu preciso e agora eu posso imprimir o resultado: "A temperatura Celsius é, então eu vou imprimir a temperatura Celsius aqui. Esse programa é nosso? Deixe-me ver. Vou gravar em um arquivo chamado "ConversorDeTemperatura.py". Eu vou salvá-lo aqui no meu diretório, que tem nossos programas, eu vou salvá-lo e então eu venho aqui e digo: "python3 ConverterDeTemperatura.py". Não preciso me desesperar porque deu um erro, porque, em geral, quando dá uma mensagem de erro, a mensagem de erro está dando uma dica de onde está o erro. Então, por exemplo, aqui ele disse: neste arquivo ConversorDeTemperatura, na linha três, esta é a linha três, há um certo erro. Qual é o erro aqui? Vejamos: "NameError". "O nome 'F' não está definido". Eu disse: "O F de Fahrenheit não está definido?" na minha fórmula, mas não no programa. No programa eu tinha chamado temperaturaFahrenheit. Então eu vou substituir este F aqui pela temperatura Fahrenheit. Então eu devo ter corrigido o erro. Então, você vê que quando há um erro, não é uma razão para desespero; podemos ler a mensagem de erro e entender, e depois corrigir e executar novamente. Executando-o novamente, deu a temperatura Celsius: é 25.555. É uma temperatura muito agradável; Posso sair no dia seguinte em paz. Mas vamos supor que dois dias depois isso mostra que a temperatura é de 20 graus. Então eu digo: "20 graus? 20 graus é agradável ou não?". E depois quero usar meu programa de novo. Então, o que eu tenho que fazer? Eu preciso voltar para o editor de texto, vir aqui para modificar meu programa, salvar, então voltar para o meu terminal, executar o programa novamente, e aqui vem: temperatura Celsius é -seis. Uau, está muito frio! Então eu preciso me vestir muito bem. Mas note que isso é meio desagradável. Do jeito que fizemos, toda vez que eu quero saber uma nova conversão, eu tenho que mudar meu programa, ir para outro lugar, para o editor, modificar o programa e voltar aqui. Não é um programa muito flexível, é? Seria mais interessante se o programa me perguntasse qual temperatura eu quero converter e cada vez que convertesse uma temperatura diferente. Como podemos fazer isso? Então, o que queremos fazer é programar isso, em vez de ter esse número fixo, que é uma entrada de usuário, esse número que eu estou passando aqui. Como posso fazer isso? Usando uma função que existe no Python, que é a função "entrada". Então, para isso, eu coloco "entrada" aqui e dou uma mensagem ao usuário: "Digite uma temperatura Fahrenheit". [SOM] Então, em vez de ter esse número fixo, o usuário terá a oportunidade de entrar em uma temperatura Fahrenheit. Vamos ver se o programa está funcionando agora. Eu salvei e correrei até aqui. "Entre em uma temperatura Fahrenheit". Por exemplo, eu quero temperatura 20. Oops! Mais uma vez deu uma mensagem de erro. Mas acalme-se, vamos ler a mensagem de erro. Ele disse: "Na mesma linha aqui, temperatura Fahrenheit...". Houve um erro de tipo. Erro de tipo? Então há algo errado, não é? Vejamos, ele está falando de subtração aqui, temperatura Fahrenheit-32, ele está falando sobre um erro de operador sem suporte nesta subtração. Ele está dizendo que estou tentando fazer uma subtração de cordas, "str", e inteiro. 32 é inteiro. Mas essa temperatura Fahrenheit é uma corda? Por que é corda? Nós veremos. Porque na verdade essa função de "entrada", ele lerá algo que o usuário digita a partir do teclado, e então essa função pode ser qualquer número, caractere. Assim, a função "entrada" sempre retorna string. E, se eu quiser usar mais tarde este valor que ele retorna em uma conta numérica, então eu preciso converter esta sequência, seja para um inteiro, ou para um carro alegórico. Aqui, no caso, eu quero flutuar. Como posso fazer isso? Eu poderia ter outra variável temporária, que vai manter este valor. Então, esta variável temporária, eu quero que ele flutue minha temperatura Fahrenheit. E aqui, em vez de usar a temperatura Fahrenheit, posso usar esta minha variável temporária. Então esta é uma maneira de fazer isso. Outra maneira alternativa seria, eu vou desfazer o que eu fiz. Uma maneira alternativa seria, o problema é aqui, a temperatura Fahrenheit é corda, eu posso convertê-lo aqui. Flutuar, eu quero que você converta esta corda em flutuar. E então eu guardo isso, vamos ver se funciona agora. Eu corro aqui de novo e digo: "Agora, eu quero que você converta a temperatura em 34". Funcionou: a temperatura 34 Celsius é uma temperatura muito fria, 1.111. Eu quero corrigir, eu não gostei que este 34 estava preso em Fahrenheit. Como podemos consertar isso aqui? Vou colocar um cólon e um espaço para torná-lo mais elegante. Então, vou tentar agora. "Entre a temperatura Fahrenheit: ", ele colocou um cólon, espaço. Eu quero, por exemplo, escolher uma temperatura muito grande agora: 200 graus Fahrenheit. A temperatura celsius é 93, a água está prestes a ferver. Mas note que agora fizemos um programa que se tornou um programa genérico. Porque toda vez que eu executá-lo, ele pede ao usuário dados de entrada, número, usuário pode digitar qualquer número aqui. Então ele pode digitar 65 e então ele converte, dá 18 Celsius. Então, fizemos um programa muito mais genérico, que recebe um valor de entrada, que realiza um cálculo e gera uma saída. Vamos praticar isso um pouco mais? Vou criar outro programa aqui. Agora quero fazer um programa, por exemplo, que diga "oi" para a mãe e "oi" para o papai. Então, por exemplo, o nome da mãe recebe informações: "Qual é o nome da sua mãe?". E o nome do pai: "Qual é o nome do seu pai?". Então, este é um programa muito educado. Ele pode dizer o seguinte: "Bom dia Sra." Então vou colocar o nome da mãe aqui. [NO\_AUDIO] O nome da mãe. Então eu vou colocar um ponto de exclamação. Pode ser bom dia... três pontos de exclamação para ser um bom dia muito enfático. E então eu também digo: "E bom dia senhor." e aqui o nome do pai. O nome do pai. Então vamos gravar isso aqui em um programa. Vou chamar o programa de "Educado". Educated.py. Desta vez eu não esqueci a extensão, Educado.py. E então podemos executar o programa aqui. Sem essa. Python3 Educado. Acho que gravei. Temos que ter cuidado com isso, certo? No lugar onde você grava eu acabei gravando outro diretório. Eu quero gravar isso aqui, eu vou dar salvar como. Salvo como, eu quero escrever para o diretório aqui do nosso curso, então educado "iii" salvar para o diretório do curso. Aqui, python3. Agora ele apareceu aqui. Educated.py. Então ele perguntou: "Qual é o nome da sua mãe?" Vamos supor que o nome da minha mãe seja Josefina. Qual é o nome do pai dela? O nome do meu pai é Josebetio. Então ele disse: "Bom dia, Sra. Josefina" e "bom dia, Sr. Josebétio". Se você é muito perfeccionista, você vai gostar disso... você não vai gostar que a parada completa está faltando aqui. Vamos acabar com isso para ser muito fofo. Então nós executamos nosso programa novamente. Aproveite e você pode dar outros nomes: Qual é o nome da sua mãe? O nome da minha mãe será Madonna. Qual é o nome do seu pai? Será paul. Sim, Sra. Madonna e bom dia, Sr. Paul. Então hoje fizemos um programa aqui. Este programa leva duas entradas, que são duas cordas, e gera uma saída que é uma string também. Só como exemplo, vamos abrir outro programa que seja um pouco mais sofisticado, um pouco mais complicado, para que possamos analisá-lo. Este é um programa que ele faz o cálculo, dada uma quantidade de segundos, então eu vou dizer um número de segundos, ele diz, ele quebra esses segundos em horas, minutos e segundos. Então vamos ver, primeiro vamos ver o programa funcionando, então vamos ver como ele faz. Este programa se chama ContaSegundos.py. Então, se eu executá-lo aqui, python3 CountSeconds.py, ele pergunta: "Por favor, digite o número de segundos que você quer converter." Então eu quero converter, por exemplo, 17345. Converte, veja: 17345 segundos são quatro horas, 49 minutos e cinco segundos. Vamos dar uma olhada neste código? Então primeiro há segundos, então há esses segundos, str é uma variável que vai armazenar os segundos em um formato de string e é o que a entrada retorna aqui quando pedimos para digitar isso. Então eu faço uma conversão. Para parar de ser uma corda eu convertê-lo em um inteiro e manter uma variável chamada segundos totais. Então eu faço alguns cálculos. Eu digo: as horas, uma hora tem 60 vezes 60, que é 3.600 segundos, então quantas horas há nessa quantidade de segundos, eu preciso fazer isso? Faça uma divisão inteira. Eu levo os segundos totais e divido por 3600, mas eu só tomo a parte inteira da divisão e guardo aqui nas horas. Para saber os segundos restantes, depois que eu tenho o número de horas, eu fico com o resto da divisão. Então aqui os segundos restantes estão esperando o total de segundos, o resto dividido por 3600. É o que sobrou além dessas horas. E então eu posso calcular os minutos. Os minutos, novamente eu faço toda a divisão do que sobrou dos segundos por 60, certo, porque cada minuto tem 60 segundos. Eu fico com toda a divisão. E os segundos restantes no final, eu levo essa variável aqui, segundos restantes, e eu levo o restante da divisão por 60 e então eu já tenho essas três variáveis: a variável hora vai armazenar o número de horas, a variável minutos vai armazenar o número de minutos e que os segundos finais restantes vai ter o número de segundos. Então é assim que nosso programa funciona. E note que eu posso executá-lo várias vezes, cada vez dando uma entrada diferente e, portanto, a saída será diferente. Então, se eu perguntar: 20.000 segundos, quanto é isso? 20.000 segundos são cinco horas, 33 minutos e 20 segundos. Vamos colocar um número muito grande aqui, que talvez seja maior que 24 horas. 120.000 segundos. Não espere, aqui agora 120.000 segundos. São 33 horas, 20 minutos e zero segundos. Se você quiser, há lição de casa para você, este programa para que, em vez dele dizendo 33 horas, para quebrar o número de dias, então ele diz: dia e, em seguida, o número de horas, número de minutos e número de segundos. Então, eu gostaria que você jogasse um pouco com as coisas que vimos nesta classe e, em particular, com entrada de dados e saída de dados, e pensar em pequenos programas interessantes que você poderia construir desta forma.

## Entrada de Dados

**Total de pontos**7

### 1.

**Pergunta 1**

Para que serve o comando "input" do Python?

**1 / 1 ponto**



Imprimir dados na tela



Ler dados digitados pelo usuário



Ler dados de um banco de dados



Ler o conteúdo de um arquivo texto

**Correto**

### 2.

**Pergunta 2**

Como exatamente será a saída dos comandos abaixo?

1

2

3

print("Olá")

print()

print("bom dia!")





**1 / 1 ponto**



Nenhuma das alternativas anteriores



Duas linhas seguidas, sendo uma para Olá e outra para bom dia!



Olá bom dia! (tudo na mesma linha)



Olá e bom dia!, cada uma em uma linha, e uma linha em branco entre elas

**Correto**

### 3.

**Pergunta 3**

O que faz o comando abaixo?

1

x = input ("Qual a sua idade?")





**0 / 1 ponto**



Compara se o valor armazenado na variável x é igual a idade informada pelo usuário



Imprime a mensagem "Qual a sua idade?" na tela, lê um dado digitado pelo usuário e armazena na variável x



Pergunta a idade do usuário



Apenas armazena um dado na variável x

**Incorreto**

### 4.

**Pergunta 4**

O que será impresso pelo trecho de programa abaixo?

1

2

3

4

5

6

a = 10

b = 20

c = a

b = c

a = b

print(a,b,c)





**1 / 1 ponto**



20 20 20



10 20 10



10 10 10



0 10 20



20 10 0

**Correto**

### 5.

**Pergunta 5**

O que será impresso pelo trecho de programa abaixo?

1

2

3

4

5

a = 10

b = 5

c = a + b

b = 20

print(a,b,c)





**1 / 1 ponto**



20 10 20



15 10 20



5 10 20



10 20 15

**Correto**

### 6.

**Pergunta 6**

O que será impresso pelo trecho de programa abaixo?

1

2

3

4

5

A = 10

a = 20

a = 2 \* a

A = a + A

print(a)





**1 / 1 ponto**



30



50



10



20



40

**Correto**

### 7.

**Pergunta 7**

O que será impresso ao executar o seguinte comando?

1

print ("olá" 'mundo')





**1 / 1 ponto**



Em uma única linha, o texto: olámundo



Erro de sintaxe, o comando print não será executado



olá em uma linha e mundo na próxima linha



Em uma única linha, o texto: olá mundo

**Correto**

### 1.

**Pergunta 1**

Qual o tipo de dado armazenado na variável x pelo comando abaixo?

1

x = input ("Qual a idade? ")





**10 / 10 pontos**



a função input sempre devolve um string



float (número real), pois considera a idade fracionada do usuário



booleano, pois o sinal de igualdade é operador relacional



inteiro, pois considera apenas a idade completa do usuário

**Correto**

### 2.

**Pergunta 2**

Quais os valores finais das variáveis a e b, se o usuário digitar 1 e 2, respectivamente?

1

2

3

4

5

6

a = int(input("Qual o valor de a? "))

b = int(input("Qual o valor de b? "))

a = b

b = a

print(a)

print(b)





**10 / 10 pontos**



a será 1 e b será 2



a será 2 e b será 1



a e b serão 2



a e b serão 1

**Correto**

### 3.

**Pergunta 3**

Quais os valores finais das variáveis a e b, se o usuário digitar 1 e 2, respectivamente?

a = int(input("Qual o valor de a? "))

b = int(input("Qual o valor de b? "))

aux = a

a = b

b = aux

print(a)

print(b)





**10 / 10 pontos**



a será 2 e b será 1



a e b serão 2



a e b serão 1



a será 1 e b será 2

# Variáveis, expressões e comandos

## Tópicos

* [Capítulo 2: Variáveis, expressões e comandos](https://panda.ime.usp.br/pensepy/static/pensepy/02-Conceitos/conceitos.html)
* comando de atribuição
* escrita na tela (= input())
* leitura via teclado (= print())
* classes str e int
* função de conversão int()
* [Comando de repetição: while](https://panda.ime.usp.br/aulasPython/static/aulasPython/while.html)
* expressões lógicas e operadores relacionais

## Estrutura de um programa

Utilizaremos a seguinte estrutura para um programa em Python.

**def** main():

*# comandos*

...

*#------*

*# a linha a seguir inicia a execução do programa*

main()

## Variáveis

Uma **variável** é um nome que se refere a um valor. Um comando de atribuição cria uma nova variável e lhe dá um valor.

Variáveis são usadas para guardarmos valores que serão usados mais tarde no programa.

## Atribuição

Um **comando** é uma unidade de código que o interpretador pode executar. Um comando de **atribuição** tem a seguinte forma:

variável = expressão

Significado:

1. o valor da expressão é calculado;
2. a variável é criada;
3. a variável passa a fazer referência/guardar o valor

Exemplos

num = 15

soma = 0 *# este é um comentário*

soma = soma + num

a = 3

b = 4

Cada nome de variáveis é uma letra ou o símbolo '\_' (underscore) seguida de letras, números e '\_'. Atenção, maiúscula é diferente de minúscula. Assim maior e Maior são duas variáveis diferentes.

## Exercício 2.1

Dados dois inteiros a e b, calcular a sua soma.

Tentativa 0: execute o programa abaixo e veja o que acontece.



1

2

3

4

5

6

7

8

9

def main():

a = 3

b = 4

soma = a + b

print("A soma de a + b eh igual a soma")

main()

(aula02\_ex1\_1)

Run Save Load

Observe que tudo entre aspas (”) define um texto ou string. Mas queremos imprimir o **valor** das variáveis e não o **nome** delas.

### print()

A função print() é utilizada para o programa exibir mensagens. Por exemplo:

**print**("Olá, mundo!")

**print**("A soma é", soma)

Escreve na tela do computador

Olá, mundo!

A soma é XXXX

onde XXXX é o valor ao qual a variável soma se refere.

Tentativa 1: execute o programa abaixo e veja o que acontece.

Observe que o print() recebeu dois **valores** separados por vírgula:

* um string (entre aspas) e
* o nome de uma variável

O string é impresso diretamente (pois esse é o seu **valor**) e, ao invés de imprimir o nome da variável soma, o print() exibe o valor ao qual a variável soma se refere, ou, dito mais simplesmente, o valor da variável soma.

Mas e como imprimir os valores de a e b ao invés de seus nomes?

Tentativa 2: execute o programa abaixo e veja o que acontece.



1

2

3

4

5

6

7

8

9

def main():

a = 3

b = 4

soma = a + b

print("A soma de", a, "+", b, "eh igual a", soma)

main()

(aula02\_ex1\_3)

Run Save Load

Agora sim, a mensagem ficou bem mais clara. Mas como fazer com que os valores a serem somados sejam definidos por um usuário?

Tentativa 3: execute o programa abaixo e veja o que acontece.



1

2

3

4

5

6

7

8

9

def main():

a = input("Digite o primeiro numero: ")

b = input("Digite o segundo numero: ")

soma = a + b

print("A soma de", a, "+", b, "eh igual a", soma)

main()

(aula02\_ex1\_4)

Run Save Load

### input()

Um programa pode utilizar a função input() para receber os dados ou valores que um usuário fornece através do teclado. O formato típico de um comando leitura é

variável = input("Prompt")

O programa pára e espera pela digitação de algum texto seguido do ENTER. "Prompt" é opcional e pode indicar o que programa deseja. Por exemplo,

nome = input("Qual é o seu nome? ")

**print**(nome, ", me fale sobre você.")

O valor que o usuário fornece e que será retornada pelo input() é sempre um string e não um número. Quando o operador + é realizado sobre strings, eles são concatenados (“grudados”) em vez de adicionados. Como veremos mais tarde, a operação de concatenar strings através do operador + é muito útil.

No momento desejamos adicionar dois números inteiros. Precisamos, portanto, de um maneira para converter um string em um número inteiro, para que o Python obtenha a soma desses números. Para ler um valor e converté-lo para um número inteiro utilizamos a função de conversão int().

### Strings

Uma **string** é texto entre aspas (") ou apóstrofo (')

### Valores

Programas manipulam valores. Valores podem ser de diferentes **tipos** ou **classes**. Até agora vimos que um **valor** pode ser um número inteiro (classe int) ou um texto (classe str). Por exemplo, "Oi" é um string (valor da classe str), 23 é um inteiro (valor da classe int), e "23" é um string (valor da classe str).

**>>>** type("Oi")

<class 'str'>

**>>>** type(23)

<class 'int'>

**>>>** type("23")

<class 'str'>

>>>

### Operacões sobre strings

O operador + concatena dois strings.

Tentativa 4: execute o programa abaixo e veja como ele se comporta.



1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

def main():

# a\_str e b\_str guardam strings

a\_str = input("Digite o primeiro numero: ")

b\_str = input("Digite o segundo numero: ")

# a\_int e b\_int guardam inteiros

a\_int = int(a\_str) # converte string/texto para inteiro

b\_int = int(b\_str) # converte string/texto para inteiro

# calcule a soma entre valores que são números inteiros

soma = a\_int + b\_int

# imprima a soma

print("A soma de", a\_int, "+", b\_int, "eh igual a", soma)

main()

(aula02\_ex1\_5)

Run Save Load

### int()

A função int() converte um dado string para um número inteiro.

**>>>** a = int("123")

**>>>** type (a)

<class 'int'>

**>>> print**("valor inteiro =", a)

valor inteiro = 123

>>>

O string deve ser algo como "123" ou "-5", pois senão ocorrerá um erro.

**>>>** a = int("1.5")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: invalid literal for int() with base 10: '1.5'

**>>>** a = int("a156")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: invalid literal for int() with base 10: 'a156'

>>>

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: invalid literal for int() with base 10: 'a156'

Como não precisamos guardar as respostas na forma de texto, podemos simplificar o programa compondo as funções int() e input() da seguinte forma:

Tentativa 5:



1

2

3

4

5

6

7

8

9

def main():

a = int(input("Digite o primeiro numero: "))

b = int(input("Digite o segundo numero: "))

soma = a + b

print("A soma de", a, "+", b, "eh igual a", soma)

main()

(aula02\_ex1\_6)

Run Save Load

## Exercício 2.2

Dada uma sequência de números inteiros diferentes de zero, terminada por um zero, calcular a sua soma. Por exemplo, para a sequência:

12 17 4 -6 8 0

o seu programa deve escrever o número 35.

Tente escrever a sua solução primeiro e, depois, clique [aqui](https://panda.ime.usp.br/aulasPython/static/aulasPython/exercicios/ex22.html) para ver uma.



1

2

def main():

num = int(input("Digite um inteiro: "))

soma = 0

while num != 0:

soma = soma + num

num = int(input("Digite um inteiro: "))

print("A soma eh", soma)

#----------------------------------------------

main()

(aula02\_ex2\_tentativa)

Run Save Load

### Comando de repetição: while

Leitura sobre [o comando de repetição while](https://panda.ime.usp.br/aulasPython/static/aulasPython/while.html).

### Expressões lógicas

Condições ou **expressões lógicas** (boolean expressions) são expressões cujo valor é verdadeiro (=``True`` em Python) ou falso (=``False`` em Python) e usam, entre outros, os operadores relacionais:

* > (maior);
* >= (maior ou igual);
* < (menor);
* <= (menor ou igual);
* == (igual); ou
* != (diferente).

## Exercício 2.3

Nota: Exercício 4 da [lista sobre inteiros](http://www.ime.usp.br/~macmulti/exercicios/inteiros/index.html).

Dados números inteiros n e k, com k >= 0, calcular n elevado a k. Por exemplo, dados os números 3 e 4 o seu programa deve escrever o número 81.

Tente escrever a sua solução abaixo primeiro e, depois, clique [aqui](https://panda.ime.usp.br/aulasPython/static/aulasPython/exercicios/ex23.html) para ver uma.



1

def main():

n = int(input("Digite o valor de n: "))

k = int(input("Digite o valor de k: "))

pot = 1

i = 0

while i < k:

pot = pot \* n

i = i + 1

print("A potencia eh", pot) # print mais simples

print("O valor de %d elevado a %d eh %d" %(n, k, pot)) # mais elaborado

#--------------------------------------------

main()

(aula02\_ex3\_tentativa)

Run Save Load

### Exercício 2.4

Nota: Exercício 8 da [lista sobre inteiros](http://www.ime.usp.br/~macmulti/exercicios/inteiros/index.html).

Dado um número inteiro n >= 0, calcular n!.

Tente escrever a sua solução abaixo primeiro e, depois, clique [aqui](https://panda.ime.usp.br/aulasPython/static/aulasPython/exercicios/ex24.html) para ver uma.



1

2

3

def main():

n = int(input("Digite o valor de n: "))

fat = 1

i = 2

while i <= n:

fat = fat\*i

i = i + 1

print("O valor de %d! eh =" %n, fat)

#--------------------------------------------

main()

(aula02\_ex4\_tentativa)

Run Save Load

**WEEK 3 AND 4**

# **Expressões Booleanas**

Vamos então aprender novo tipo de valor que nós temos

diversas linguagens de programação e que é muito importante na computação,

que são os valores booleanos ou as expressões booleanas.

De onde vem esse nome booleano?

Vem, é nome homenagem ao matemático britânico George Boole,

que criou o que a gente chama de álgebra booleana.

Ele foi matemático que estudou valores que podem ter apenas dois tipos de coisa,

por exemplo, verdadeiro ou falso ou 0 e 1, ou ligado e desligado.

Ele criou toda uma álgebra para manipular esses valores e como os computadores ali,

internamente eles funcionam com código binário, essa álgebra booleana é muito

apropriada para tratar vários aspectos aí da computação.

Então particular, boa parte das linguagens de programação têm tipo de valor,

tipo de dado que é o tipo booleano.

Vamos então agora abrir o nosso interpretador

Phyton e ver como funciona Phyton esses valores booleanos.

Se eu fizer por exemplo 5 maior que 3, isso dá True.

Toda vez que eu faço uma comparação ela vai dar ou verdadeira ou falsa,

então esse tipo de valor que pode ser verdadeiro ou falso é valor booleano.

Se eu fizer 18 é igual a nove vezes dois, isso vai dar verdadeiro também.

Agora se eu botar x valor aqui assim e fizer x é menor que zero,

isso vai dar falso.

Particular se eu perguntar qual é o tipo de falso?

Ele vai dizer, o tipo de falso é bool,

o objeto falso é da classe bool, de booleano.

A mesma coisa se eu perguntar: qual é o tipo de x maior que zero?

Então a expressão x maior que zero é uma expressão da classe booleana,

da classe bool.

Agora com esse valores booleanos a gente pode criar expressões

booleanas e pra isso a gente pode pegar operadores lógicos,

operadores booleanos, tipo e, ou e agrupar esses valores booleanos.

Vamos ver aqui exemplo com o e, com and primeiro.

Então, eu posso dizer algo do tipo, se eu tenho o x que tem aquele valor,

eu posso falar: x é maior que zero and x ao

quadrado é maior que 100.

Então, essa é uma expressão aqui que a gente chama expressão booleana,

que eu tenho e lógico, o and que o resultado final só vai ser verdadeiro se

ambos os lados forem verdadeiros, então se x for maior que zero e x ao quadrado for

maior que 100, então o resultado final vai ser verdadeiro.

Se dos dois for falso então o resultado final é falso.

Nesse caso os dois são verdadeiros então o resultado final é verdadeiro,

se fosse x menor que zero and x é

igual a 12312.

Isso aqui dá falso, porque o primeiro, primeiro valor booleano aqui,

x maior que zero já é falso, então ele nem precisa verificar o segundo,

é com certeza falso, porque é e lógico o and que nós temos.

Outra opção é o or.

O or ele vai escolher, o resultado do or

vai ser verdadeiro se a primeira parte é verdadeira ou a segunda

parte é verdadeira, então se eu fizer x menor que zero or

x é igual a 12312, isso dá verdadeiro, porquê?

Porque a primeira parte é falsa mas a segunda parte é verdadeira então,

o resultado do ou,

o ou ele vai ser verdadeiro se ou outro ou ambos forem verdadeiros.

Outro tipo de valor que, operador lógico que nós temos,

que é o mais simples de todos é o not.

O not ele simplesmente inverte o valor de valor booleano.

Então se x maior que zero, se isso é verdadeiro,

not x maior que zero, vai ser falso.

Particular o not False é True e not True é False.

Eu posso fazer também: not not True vai ser verdadeiro,

ele inverteu duas vezes ali o valor do verdadeiro voltou a ser verdadeiro.

Uma coisa que nós podemos fazer é uma,

aqui, vamos ver isto, uma tabela que a gente

chama da tabela da verdade com os valores, eu abri aqui uma planilha no Libre Office.

Então aqui o operador lógico and, se a primeira,

primeiro valor é verdadeiro e o segundo valor é verdadeiro

o resultado dessa expressão booleana vai ser verdadeiro.

Se o primeiro é verdadeiro e o segundo é falso daí o resultado vai ser falso.

Porque o and exige que ambos sejam verdadeiros.

Se o primeiro é falso e o segundo é verdadeiro, sinto muito vai dar falso do

mesmo jeito e se ambos forem falso, obviamente vai dar falso.

Então, essa aqui é a tabela da verdade do operador and.

O operador or é operador muito mais bonzinho não é?

Porque, tanto faz se o primeiro for verdadeiro, ou o segundo for verdadeiro

o resultado vai ser verdadeiro, então se os dois forem verdadeiros é verdadeiro,

se o primeiro é verdadeiro e o segundo é falso é verdadeiro,

se o primeiro é falso e o segundo é verdadeiro, também é verdadeiro.

O único caso que or or dá falso é quando ambos são falso,

daí também dá falso, não é?

Aí ninguém é de ferro.

Já o operador not é operador aquele simples, ele simplesmente muda o valor,

então se eu tenho not True isso aqui da False,

False e se e tenho not False, isso aqui dá True.

Então isso aqui que a gente chama de: a tabela da verdade dos operadores lógicos,

é algo interessante de a gente ter isso na cabeça, não é?

Quando estivermos programando Phyton, muitos casos a gente vai ter,

querer fazer combinações lógicas de valores lógicos, por exemplo,

eu quero decidir se segunda-feira eu vou para a praia,

alguém pergunta para você: você vai para a praia na segunda-feira?

Daí você vai falar: eu vou para a praia na segunda-feira, se for

feriado mesmo e nós não tivermos que trabalhar e se fizer sol,

e se fizer sol, então eu posso dizer algo do tipo assim,

vamos supor que eu tenho uma variável que chamava FizerSol,

suponho que ela vale True e uma outra variável chamada,

forFeriado e vamos supor que essa variável é False,

daí eu digo que eu vouParaPraia

se fizerSol and forFeriado.

Então, o que é que o interpretador Phyton fez?

Ele fez esse cálculo aqui do fizerSol and

forFeriado e nesse caso aqui o resultado deu falso,

porque o forFeriado foi falso.

Se por acaso eu escolho outro dia que o

forFeriado é verdadeiro, então eu vou colocar aqui verdadeiro no for

feriado e vou calcular novamente aqui, vou refazer o cálculo.

Refazendo o cálculo agora, agora eu vouParaPraia deu verdadeiro, que bom né?

Porque vai estar fazendo sol e é feriado, então perfeito para eu ir para praia.

Já exemplo de operador lógico ou, vamos supor que amigo convida você para

ir num show de rock nesse final de semana e se perguntar, legal mas é muito caro?

É meio caro o ingresso.

Daí você fala: Eu vou perguntar para o meu pai,

ver se ele pode me emprestar dinheiro para comprar o ingresso ou então,

eu sei que tem uma rádio ali com uma promoção, que você ligando para

a rádio respondendo uma pergunta às vezes ele dá ingresso de graça.

Uma vez eu fiz isso, eu fui show do Hermeto Pascoal porque a Rádio Cultura

estava dando ingresso de graça, acertei a pergunta, ganhei o ingresso.

Então, como eu falaria isso?

Eu posso falar algo do tipo: vouAoShow se eu receber, se meu pai pagar,

então se eu receber paitrocinio ou, or,

se rolarPromocao da rádio,

então posso falar alguma coisa desse tipo,

vou ao show se tiver paitrocinio ou rolar a promoção.

Mas primeiro para eu poder calcular algo desse tipo eu preciso definir o

valor dessas variáveis, então vamos só de exemplo.

Se meu paitrocino aqui, por exemplo, não consegui convencer o meu pai ou vai

ver que o meu pai não achou que era bom para minha educação musical ir nesse show,

então não consegui aqui meu paitrocinio, por outro lado, ali o rolarPromoção.

Eu sou muito sortudo, liguei para a rádio, consegui e ganhei o ingresso.

Daí sim eu posso fazer aquele cálculo, vouAoShow igual a paitrocinio or

rolarPromocao e daí qual é o valor aqui do vouAoShow?

O vouAoShow é É verdadeiro, porque mesmo não tendo conseguido o patrocínio rolou

a promoção e eu consegui como era operador ou ele permite que,

ou ou outro, seja verdadeiro.

Outra coisa interessante que a gente tem

que olhar com cuidado, é a questão da precedência de operadores.

Se a gente olhar aqui,

essa é a tabela de precedência de operadores da linguagem Python.

Então, se eu tenho uma expressão complexa, com vários operadores diferentes,

tem uma certa ordem na qual esses operadores são calculados, então,

o operador aqui de mais alto nível é a exponenciação,

quando faço uma exponenciação com esse operador aqui.

Então, se tiver uma expressão qualquer bem complexa ali,

primeiro ele vai fazer a exponenciação, segundo lugar,

ele vai fazer os operadores multiplicativos, que é a multiplicação,

a divisão, a divisão inteira e o resto da divisão inteira.

Depois vêm os aditivos, que é a soma e a subtração,

depois os de comparação, igual, diferente, menor ou igual, maior ou igual, etc.

E depois vêm os lógicos, sendo que os lógicos são nessa ordem, o not,

depois o and e o que tem mais baixa precedência de todos é o or.

Então, vamos ver exemplo.

Se eu fizer uma operação como essa aqui: x maior que zero and

deixa eu definir uma variável y aqui para usar nesse meu exemplo, então,

vamos supor que o y é 50.

Então, faço x maior que zero and

not y é igual a 50 or

x mais

y é igual a 150.

Se eu fizer essa operação aqui, como que o Python vai interpretar isso?

Lembra?

Pela ordem dos operadores, de todos os que estão aqui,

essa soma é a que tem maior precedência.

Então, ele vai fazer o x mais y aqui e vai calcular e vai,

depois que tenhas as respostas de x mais y ele vai comparar para ver

se é igual a 150 ou não e nesse caso não é porque o x era valor bem maior então,

esse pedaço aqui vai dar false.

Depois, ele vai fazer todas as comparações e o

que tem precedência maior depois vai ser o not,

então, ele vai pegar esse y igual a 50 que é valor,

é verdadeiro, y igual a 50 é verdadeiro e vai fazer not,

vai virar false, esse pedaço aqui vai virar false.

Aí sim, ele vai executar o and então,

x é maior que zero é verdadeiro com essa parte que a gente viu que era

false ele vai fazer verdadeiro com false, vai dar false e daí,

por último ele vai fazer esse or, então tudo isso aqui que deu false,

mais com isso aqui que também, a gente viu é false então,

o resultado final inteiro vai dar false, mas note que tá meio confuso a ordem

que essas coisas vão acontecer, então quando você tem uma expressão complicada,

o ideal é você colocar uns parêntesis aqui para ficar mais claro.

É sempre ideal a gente escrever o código o mais claro possível.

Então, para evitar qualquer dúvida, mesmo sendo redundante, é bom a gente colocar os

parêntesis aqui para ficar bem claro que é que está sendo feito, por exemplo aqui eu

vou colocar bastantes parêntesis para ficar bem claro a ordem das coisas.

Então, assim, fica bem claro, não mudou nada,

continua sendo false ali, mas é jeito mais elegante de escrever o código.

Então agora quero que vocês entrem no interpretador Python no seu

computador e brinquem pouco com variáveis, com valores booleanos,

com essas expressões lógicas usando and, or, not,

e expressões complexas sofisticadas, misturando comparações

com operadores lógicos, com operadores aritméticos, então brinquem ali uns

cinco a dez minutos no interpretador para você praticar bastante isso.

### 1.

**Pergunta 1**

Assumindo que x está definido como um número inteiro, qual expressão booleana abaixo está escrita de forma correta e que sempre retornará True?

**0 / 1 ponto**



(x != 0) and (x == 0)



(x > 0) or (x <= 0)



(-10 < x < 0)



True and (4 => 3)



not (x > 0) and (x > 0)

**Incorreto**

Você não selecionou uma resposta.

### 2.

**Pergunta 2**

No Python, o operador relacional de igualdade é:

**1 / 1 ponto**



=



<>



!=



==



^=

**Correto**

### 3.

**Pergunta 3**

Se x = 5 e y = 3, qual será o resultado da expressão abaixo?

1

x > y





**1 / 1 ponto**



3



False



5



True



Sim

**Correto**

### 4.

**Pergunta 4**

Se x = 5, y = 3 e z = 7, qual será o resultado da expressão abaixo?

1

x > y and x < z





**1 / 1 ponto**



True



False



não



4



6

**Correto**

### 5.

**Pergunta 5**

Qual o resultado do trecho abaixo?

1

2

3

4

idade=15

maioridade=18

pode\_dirigir = idade >= maioridade

print (pode\_dirigir)





**1 / 1 ponto**



False



True



Sim



18

**Correto**

### 6.

**Pergunta 6**

Que tipo de erro ocorre ao executar o comando abaixo, considerando que você tenha acabado de entrar no interpretador Python?

1

fruta = laranja





**1 / 1 ponto**



NameError.



TypeError.



SyntaxError.



ValueError.

**Correto**

### 7.

**Pergunta 7**

Qual a saída gerada pelo trecho de programa abaixo?

1

2

3

4

x = 10

y = 15

z = 25

print(x == z - y and z != y - x or not y != z - x)





**1 / 1 ponto**



SyntaxError



False



Undefined



True

**Correto**

### 8.

**Pergunta 8**

Considere x = 10, y = 20 e z = 30, assinale quais das alternativas abaixo resultam em True:

**1 / 1 ponto**



print(x >= 10 or y != z - x)

**Correto**



print(x <= 30 and y >= 5)

**Correto**



print(not y < 10 or not z == 10)

**Correto**



print(not y > 10 or not z > 10)

### 9.

**Pergunta 9**

Assinale as opções que identificam os nomes dos tipos de dados em Python:

**1 / 1 ponto**



int

**Correto**



bool

**Correto**



10



True



Booleano



Syntax Error



str

**Correto**



float

**Correto**

### 10.

**Pergunta 10**

Assuma que a = 1 e b = 2. Qual declaração é verdadeira (True)?

**1 / 1 ponto**



a != 1 or b == 1



a == 2 and b != 1



not (a == 1)



a != 2 or b == 1

**Correto**

# **Execução Condicional**

Olá! Vamos então aprender uma coisa que

a gente chama de execução condicional.

Tem alguns momentos que a gente quer escrever determinado trecho de código,

que só vai ser executado caso alguma condição seja satisfeita.

Então, a gente não vai executar aquilo sempre, só alguns casos.

E como que a gente faz isso?

A gente faz com o comando if,

que praticamente toda a linguagem de programação tem esse comando if, não é?

Como é que o comando if?

Eu digito aqui if, i f, não é, e daí eu digito uma condição.

Essa condição vai ser uma expressão booleana cujo valor é verdadeiro ou falso,

daí eu digito ':' e daí na linha de baixo eu coloco alguns espaços, por exemplo

dois espaços ou quatro espaços para dar o que a gente chama de indentação,

e daí, eu escrevo aqui o meu comando que eu quero que seja executado, está?

Esse aqui é o formato genérico ali do comando if e, sempre assim.

Se a condição for verdadeira ele executa o comando, caso contrário,

se a condição não for verdadeira, for falsa, ele não executa o comando,

ele vem aqui para a linha seguinte executar o,

por exemplo o comando seguinte que está aqui, está?

Vamos aqui exemplo no nosso interpretador.

Então, eu posso, por exemplo, se eu tenho aqui temperatura é igual a 102.

Daí, eu posso ter trecho de código que eu falo: if temperatura

é maior que 100,

daí, então, eu posso querer dizer que

a aguaFerve =, recebe true.

Se a temperatura for maior que 100 a água vai ferver,

então, coloquei valor true nessa variável de aguaFerve.

Então, se eu fizer executar isso aqui, se eu for

olhar agora o valor dessa variável aguaFerve, ela está valendo true, porquê?

Porque a temperatura é maior que 100.

Agora, vamos supor que eu quero na verdade eu quero fazer mais do que comando,

eu quero executar dois comandos caso a temperatura for maior que 100.

Daí, o que eu posso fazer, voltando aqui a esse formato genérico aqui, eu posso

colocar outros comandos aqui desde que eu use o mesmo, a mesma indentação.

Então, se eu coloquei aqui quatro espaços,

aqui eu coloco os mesmos quatro espaços, está?

Então, dessa forma eu estou como se fosse criando bloco aqui que,

todo esse bloco é executado, caso essa condição seja verdadeira.

Note que aqui pode ter quantos comandos eu quiser desde que eu sempre use

a mesma indentação.

Posso ter dezenas aqui se quiser, comandos dentro de único if.

Então, eu poderia escrever algo do tipo: se a temperatura é maior que 100,

não só a aguaFerve como a evaporação,

eu quero que ela receba "muito rápida".

Então, executando isso aqui,

não só a variável aguaFerve contém esse

valor booleano true, como a variável evaporação,

contém aqui string cujo conteúdo do string é "muito rápida".

Então, uma única condição levando à execução de dois comandos.

Então, isso é o que a gente chama de execução condicional, tem uma expressão

booleana que, define se determinado pedaço de código vai executar ou não.

Agora, tem alguns casos que você quer executar

alguns comandos, caso a condição seja verdadeira mas,

caso a condição seja falsa você quer executar outro grupo de comandos.

Daí, como que a gente faz isso?

A gente usa o if else.

Então, é assim, if condição, os comandos, else,

caso contrário, dois pontos, e daí, que a gente coloca outros comandos.

Então, esses aí vão ser os comandos que vão ser executados caso seja verdadeira,

três comandos diferentes, caso seja verdadeira e,

aqui, ou mais comandos que vão ser executados caso seja falso.

Então, vou colocar aqui, por exemplo, dois comandos mas,

pode ser quantos você quiser.

Então, nesse caso, se a condição for verdadeira ele executa apenas esses três,

se a condição for falsa, ele executa apenas esses dois.

E, ambos os casos depois que ele executou esses comandos do if else,

ele, ambos os casos pula para o comando seguinte aqui,

que é o comando que vem depois aqui do else sem essa indentação,

note que não tem mais esses espaços que eu coloquei aqui no começo.

Então, você vê que na linguagem Python esses espaços aqui são muito importantes.

Tem outras linguagens de programação que simplesmente ignoram os espaços,

os espaços não tem nenhum significado.

Python tem esse significado de definir os blocos, particular, os blocos aí do

if e do else, são identificados, são definidos com essa indentação.

Vamos fazer exemplinho aqui de programinha.

Eu vou abrir o meu editor de textos aqui,

e eu vou criar novo programa.

Então, vamos ver aqui exemplo de uso do if else,

num caso ali pouquinho mais concreto.

Vamos fazer programinha que eu vou perguntar aqui para o meu usuário,

qual que é o tempo de jogo.

Então, eu vou usar o comando input para fazer a pergunta e eu vou perguntar:

"quanto tempo temos já

jogado?" Eu vou

perguntar isso, só que eu quero esse valor como número inteiro.

Se eu fizer input ele sempre volta como string, como uma cadeia de carateres,

eu vou converter isso aqui para inteiro colocando esse int aqui,

e já resolve o meu problema.

Dai eu vou falar o seguinte: if tempoDeJogo é menor ou igual a 90,

"Quanto tempo temos já jogado?" não é,

singular, se eu fizer if tempoDeJogo menor ou igual a 90,

daí eu posso falar algo

do tipo: "Ainda tem jogo

pela frente", e posso falar alguma outra coisa,

se eu gosto de futebol posso falar: "Que bom, eu adoro futebol".

Você pode trocar pelo seu esporte preferido aqui,

está, não precisa se ater ao futebol.

Mas, por outro lado, else,

eu posso falar caso contrário,

ou seja se o tempo de jogo não é menor ou igual a 90, se ele é maior que 90,

daí eu digo "Putz, está acabando o jogo"

e, além disso,

se o seu time estiver ganhando você pode falar "Apita logo, juiz!!!" Então,

esse aqui é o nosso programa.

Salvei aqui, num arquivo chamado jogo.py.

Daí, eu posso executar aqui, Python, vou executar o Python 3, a versão 3,

jogo.py e ele me pergunta: "Quanto temos já jogado?".

Então, se já jogamos 45 minutos, ele vai falar: "Ainda tem

jogo pela frente" "Que bom eu adoro futebol" Por outro lado,

se já foram jogados ali 92 minutos, ele vai falar "Putz está

acabando o jogo" "Apita logo, juiz!!!" Então, a gente vê exemplo ali, que a gente

executa blocos diferentes de código, dependendo do que está acontecendo.

Agora, eu vou querer que você use o if else para fazer exercício,

desafio que eu vou dar para vocês.

E o desafio é bem mais complexo do que a gente viu até agora,

que é fazer programa que calcula as raízes de uma equação de segundo grau.

Vejamos aqui se a gente tem uma equação quadrática não é?

Uma equação desse tipo aqui ax2 mais bx mais c é igual a 0,

não é, o que é que a gente tem nessa equação?

x é variável, a gente tem essas constantes, esses parâmetros, o a,

o b e o c, que são os valores de entrada ali para o nosso programa e, a gente vai,

então, dizer qual é a a raiz dessa equação, a raiz

você tem uma parábola assim, são os pontos que a parábola encontra ali o eixo x.

Mas note que o if vai ser importante aqui porque tem três casos diferentes.

Primeiro, lembra qual é a fórmula?

A fórmula de Bhaskara, não é, Bhaskara é aquele matemático indiano que inventou

essa fórmula de Bhaskara aqui não é, que diz que x é igual

a menos b mais ou menos raiz quadrada de b2 menos 4ac dividido por 2a.

Está aqui a fórmula de Bhaskara.

Então, essa parte aqui que tem a raiz a gente chama de delta e,

porque é que o if é importante?

Porque se o delta for igual a zero, daí,

essa equação aqui ela só tem uma raiz,

porque essa parábola vai encostar aqui no ponto mínimo vai encostar aqui no zero,

então ele só vai ter uma raiz, isso se o delta for igual a zero.

Se o delta for negativo ela não vai ter raízes reais,

as raízes vão ser imaginárias.

Vamos fazer programa que calcule apenas raízes reais, então,

nesse caso do delta for menor que zero,

eu quero que o programa diga: "Esta equação não possui raízes reais".

Se o delta for igual a zero ela vai dizer: "A raiz desta equação é"

e o valor da raiz, se o delta for maior que zero estritamente,

daí o seu programa vai falar: "As raízes da equação são", e vai dar a primeira raiz

e vai dar a segunda raiz e, para isso você vai precisar do comando if,

ou talvez mais do que comando if para discernir esses 3 casos diferentes.

Agora, você já sabe tudo o que você precisa para calcular esse resultado

não é, para aplicar essa fórmula e dar o resultado, os três possíveis resultados,

com exceção de uma coisa, a gente ainda não viu como fazer raiz quadrada, não é?

Toda a vez que você tiver necessidade de procurar algo novo na linguagem Python

que você não sabe, o que é que você faz?

Você vai ali no Google e você digita, por exemplo: Python documentation,

nesse caso eu quero saber como se faz raiz quadrada, não é?

Então, eu vou digitar inglês: square root, raiz quadrada inglês.

E daí, o que é que a gente vê?

Os dois primeiros links é a documentação do Python da versão 2.7 e da versão 3.5.1.

Eu vou abrir essa versão mais recente aqui não é, versão 3.5.1, e a gente caiu nessa

página aqui que é a documentação do módulo Math da linguagem Python.

Então, é módulo que provê acesso a funções matemáticas definidas ali,

não é, mathematical functions, particular, se eu buscar aqui por square root,

a gente vê que essa função math.sqrt aqui,

ela devolve a raiz quadrada de x.

Então, vamos ver como usa ela.

Se eu abrir aqui o meu interpretador Python e perguntar, por exemplo,

qual a raiz quadrada math.sqrt de oito, e eu apertar enter, que é que acontece?

Opa, deu erro, ele falou: " o nome math is not defined".

E porquê?

Quando você usa módulo externo que não é padrão,

você precisa primeiro importar este módulo e a gente faz isso com o comando import.

Então, simplesmente escrevo import math e a partir daí todas aquelas funções

matemáticas do módulo math já estão disponíveis para mim.

Então, se eu repetir a mesma coisa aqui, fizer math.sqrt(8) aí

ele vai dar ali o valor da raiz quadrada de oito.

Só para a gente ter a certeza que ele está calculando certo,

vamos pegar a raiz quadrada de nove, que a gente sabe qual é, não é,

a raiz quadrada de nove é três, então, pronto já temos uma forma de calcular.

Então, se tiver uma variável delta lá que, vai ter todo cálculo ali,

vamos supor que o delta deu isso aqui, e daí, você pode simplesmente fazer

math.sqrt(delta) e você já tem ali a raiz quadrada do delta.

Então, tudo isso é o que você precisa para fazer esse exercício.

Então, vamos lá, com afinco, quero que vocês façam esse programa.

Relembrando, esse programa vai receber como entrada três valores, o a,

b e c, que são as constantes ali da equação de segundo grau e daí

você vai usando a fórmula de Bhaskara, imprimir as raízes, sendo que tem três

casos se delta é menor que zero você vai dizer que não tem raízes reais,

se delta igual a zero você vai dizer que tem uma raiz real que é tal, se delta for

maior que zero, você vai dizer que tem duasraízes reais que são: tal e tal.

Então, trabalhe com afinco nesse problema, não desistam,

está, não vai na primeira dificuldade já cair fora, fica lá até sair, está?

Então, ao trabalho.

## Execução Condicional

**Total de pontos**8

### 1.

**Pergunta 1**

O que é impresso ao digitar: 1 + 1 + 1 == 3?

**1 / 1 ponto**



6



True.



False.



Não há garantias de que 1+1+1 == 3 é verdadeiro.

**Correto**

### 2.

**Pergunta 2**

Considere que a variável is\_ready é do tipo booleano. Qual declaração está correta e é a forma mais sucinta de testar se is\_ready é verdadeiro?

**0 / 1 ponto**



if (is\_ready = True):



if (not is\_ready == False):



if (is\_ready == True):



if (is\_ready):



if (not is\_ready = False):

**Incorreto**

### 3.

**Pergunta 3**

A linguagem Python permite o uso de funções matemáticas. Porém, não são nativas da linguagem e ficam localizadas em módulos externos.

Para se usar a função sqrt, por exemplo, é necessário usar o seguinte comando:

**1 / 1 ponto**



include <math.h>



import math



use math



export math

**Correto**

### 4.

**Pergunta 4**

Considere o trecho de comandos abaixo. Qual o número da linha que será responsável pela saída (apresentação do resultado ao usuário) desse código?

5

5.      print ("texto com 10 ou menos caracteres")





**1 / 1 ponto**



linhas 3 e 5.



linha 3.



nem linha 3, nem 5.



linha 5.

**Correto**

### 5.

**Pergunta 5**

A legislação de trânsito do Brasil permite que apenas pessoas com no mínimo 18 anos possam habilitar-se para dirigir.

Considere que exista uma variável idade e você deverá testar se o usuário pode dirigir ou não e, em seguida, exibir uma mensagem correspondente à sua situação. Todas as opções abaixo dão o resultado correto, porém, qual delas utiliza melhor o recurso do if..else?

14

15

16

17

18

19

20

21

22

23

24

         if (idade >= 18):

               print ("Pode tirar a carteira de habilitação")

IV: if (idade < 18):

       print ("Não pode tirar carteira de habilitação")

    else:

      if (idade == 18):

          print ("Pode tirar a carteira de habilitação")

      else:

           if (idade > 18):

               print ("Pode tirar a carteira de habilitação")





**1 / 1 ponto**



I



III



Nenhuma das alternativas



II



IV

**Correto**

### 6.

**Pergunta 6**

Após executar a atribuição x = 10, qual afirmativa é verdadeira?

**1 / 1 ponto**



x != 10



not(x == 10)



x == 20



x != 20

**Correto**

### 7.

**Pergunta 7**

Qual(is) dos seguintes comandos é(são) equivalente(s) a x != y?

**1 / 1 ponto**



x > y or x < y

**Correto**



x > y and x < y



x >= y or x <= y



not (x == y)

**Correto**

### 8.

**Pergunta 8**

Considere a=0, b = 2 e c = 1. O que será impresso pelos comandos abaixo? (Primeiro ajuste corretamente a indentação.)

8

if (a > 0):

    if (b > 0):

        print ("Tudo ok para decolagem!")

    else:

        print ("Tanque principal vazio; voando com combustível reserva!")

else:

    if (c > 0):

        print ("Foguete não tem piloto, mas há outros foguetes disponíveis!")





**1 / 1 ponto**



Foguete não tem piloto, mas há outros foguetes disponíveis!



Tudo ok para decolagem!



Tanque principal vazio; voando com combustível reserva!



nada.

**Correto**

Olá, vamos então agora aprender como a gente faz uma

repetição numa linguagem de programação, o que a gente chama de laço, ou loop.

E uma das formas de fazer isso é por meio do comando while da linguagem Python.

Várias linguagens de programação têm esse comando while.

Qual é o o formato?

A gente vê daqui a pouco, vamos primeiro ver exemplo.

Vamos supor que eu quero imprimir as potências de 2, software, programação.

As potências de 2, computação geral,

são muito importantes porque os computadores internamente usam todos

números binários e toda a arquitetura do computador é baseada potências de 2.

Então, vamos supor que eu quero imprimir as 10 ou as 16 primeiras potências de 2.

Por exemplo, como eu poderia fazer isso?

Posso fazer algo do tipo print 2 elevado

a 0 e isso vai imprimir a primeira potência de 2,

depois 2 elevado a 1, depois, 2 elevado a 2,

e assim por diante, repetindo aqui várias vezes.

Coloco 2 elevado a 1, elevado a 2, elevado a 3, elevado a 4,

5, 6, 7, 8, 9, 10, não, 0 não,

10, 11, 12,

13, acho que está bom por aqui.

Então, gravei programa assim,

eu vou chamar esse programa de potências ponto py,

e vou salvar aqui as potências.py.

E daí, se eu quiser executar esse programa, eu venho aqui e falo,

python3, potências.py.

E ele imprime ali as potências de 2: 1, 2, 4, 8, 16, 32, 64, etc, né.

Aqui até 8192.

Vocês podem perceber que isso aqui não é uma coisa muito elegante,

então particular tem muita repetição ali, comando quase idêntico repetido,

repetição linguagem de programação sempre é uma má ideia.

Como a gente pode fazer jeito mais elegante,

mais inteligente e muito mais enxuto para fazer a mesma coisa?

É aí que entra o comando while,

que é esse comando que permite à gente fazer os laços.

Como é que funciona esse comando?

Então, a gente escreve aqui while, daí uma expressão booleana,

algo que vale verdadeiro ou falso, então uma condição aqui,

dois pontos, e daí uma indentação e aqui, ou mais comandos

que você quer que seja executado, por exemplo, comando 1 e comando 2.

Pode ser mais comandos também, o importante é a indentação aqui definir o

tamanho, onde começa e onde termina o bloco.

O que vai acontecer?

Enquanto essa condição for verdadeira,

o computador vai continuar executando esse laço aqui.

Quando essa condição se tornar falsa pela primeira vez,

ele deixa de executar e vai aqui para o comando seguinte, para o comando 3.

Então, como ele funciona?

Ele entra no while, se a condição é falsa de cara,

ele já não executa nenhuma vez e vem para o comando 3 direto.

Se a primeira vez é verdadeiro, o que ele faz?

Executa o comando 1, executa o comando 2 e daí testa de novo a condição.

Executa o comando 1, comando 2, testa a condição, comando 1, comando 2,

testa a condição.

Então, enquanto a condição é verdadeira, sempre depois do comando 2 aqui ele vai

testar a condição, se for verdadeira vai, volta para o comando 1, comando 2.

Se for falsa, ele pula ali para o comando 3.

Então, como a gente podia reescrever isso aqui de uma forma mais inteligente?

Então, primeiro, a gente tem que identificar aqui o que

é a parte constante e o que é a parte variável.

Constante é tudo daqui do daqui para "iii", dessa exponenciação para cá,

tudo constante, o que está variando é só este valor.

Se esse valor está variando,

a gente precisa de uma variável para guardar esse valor, então, por exemplo,

uma variável i, que eu vou começar, o valor inicial é zero.

E a gente vai precisar aqui, esse comando, a gente vai usar o i,

vez de usar valores fixos, então, a variável é o i.

Só que agora eu preciso colocar isso dentro do while, então, eu vou falar,

enquanto daí eu preciso aqui de uma determinada condição.

Enquanto essa condição for verdadeira, daí você imprime 2 elevado a i.

Agora eu preciso de critério de parada,

quando é que eu vou fazer disparar falso nessa condição?

Daí eu preciso decidir, ou posso fazer programa que imprime,

por exemplo, as dez primeiras potências, se eu quiser fazer programa que imprime

as dez primeiras potências, daí eu posso fazer: enquanto o i for

menor ou igual a 10, aí, eu vou repetindo isso aqui.

Só que daí eu preciso ir variando o valor do i, então,

depois que eu imprimir aqui o 2 elevado a i, daí eu preciso o que a gente

chama de incrementar o valor de i, somar no valor de i, como eu faço isso?

Eu faço: i recebe, quanto, i mais 1.

Então, o novo valor de i vai ser o valor antigo de i mais 1.

Então, fazendo isso ele vai repetido aqui até que o i fique valendo 11,

e daí quando o i valer 11, isso aqui vai valer falso e daí ele para de executar.

Vamos testar, vamos ver se tá funcionando isso aqui, esse é o potências.

Deu certo.

Então programa muito mais enxuto, imprimiu ali, se eu quiser mudar o valor aqui,

eu posso mandar, imprimir as primeiras vinte potências de 2,

as vinte primeiras potências de 2 estão aqui.

Se eu quisesse, eu poderia alterar isso para vez de vinte ser valor fixo,

você lê o valor do teclado, o usuário vai dizer qual é o valor.

Mas essa é a ideia do comando while.

Vamos ver outras coisas que podemos fazer com o while.

Por exemplo, vamos supor que você quer fazer a soma de uma sequência de números,

então você quer que o usuário vá digitando vários números e você vai calculando

a soma e, no final, você fala a soma completa de todos esses números.

Como é que a gente poderia fazer isso?

Eu vou criar novo aqui.

Então a gente vai querer ter uma variável soma ali,

que vai lendo vários números e a gente vai somando esses números na soma.

Então vejamos, para ler no teclado a gente vai precisar pegar

valor aqui, a gente vai usar o input.

Digite valor a ser

somado.

É.

E, agora eu vou querer colocar isso dentro de while,

dentro de laço de repetição de forma que eu vou poder digitar vários valores.

Então, vou fazer o while, e aqui de novo eu vou precisar de uma condição.

De novo, qual vai ser o critério de parada?

Ou eu posso, por exemplo, posso perguntar para o usuário inicialmente

quantos valores tem nessa minha sequência.

E daí ele digita e eu faço laço que vai executar exatamente aquele

número de interações, de repetições, ou então eu posso falar, por exemplo,

digite uma sequência de valores terminada por zero, e quando ele digitar zero, daí,

eu posso parar.

Vamos usar esse valor inicial, aqui assim, dessa forma.

Então, eu vou falar: print digite uma

sequência de valores

terminada por zero, pelo número zero.

Vou pedir para ele digitar isso,

e daí eu vou fazer aqui while, minha condição vai ser o quê?

While o valor é diferente de zero, então, enquanto o valor não for

zero, daí digite valor a ser somado e daí,

para cada desses valores eu vou fazer aqui

soma recebe o valor antigo da soma mais o que acabou de ser digitado pelo usuário.

Então, soma recebe soma mais valor,

ele vai pegar na soma e ele vai cumulativamente calculando essa soma.

Mas note que eu preciso definir o valor inicial aqui.

Como? Vou fazer uma sequência de somas e vou

começar com o elemento neutro da soma que é igual a zero, e então,

começo com a soma zerada e daí eu vou repetindo, sempre tendo o valor da soma.

Agora eu tenho outro problema.

Note que a primeira vez que eu entro nesse while, e eu estou falando enquanto o valor

diferente de zero, só que a primeira vez que eu entro no while o valor não tem

nenhum valor, não foi definida a variável aí, então, isso vai me dar erro,

porque isso só é definido aqui.

Daí tem duas formas de resolver isso: uma forma é digitar

valor qualquer que não seja, que seja diferente de zero,

de forma que ele entre aí dentro, então por exemplo.

Se eu fizer: valor é igual a simplesmente é suficiente para ele já entrar no while.

Na primeira vez ele já vai ler valor correto aqui.

E daí finalmente, depois que eu calculei a soma,

eu preciso dizer o valor da soma final.

Então, a soma dos valores

digitados é e daí, eu coloco aqui a soma.

Vamos ver se eu acertei ou se eu me esqueci de alguma coisa.

Eu vou chamar esse programa aqui de soma.py.

Vamos ver se dá certo.

Python3 soma.py.

O que é que eu fiz?

Gravei num outro diretório, gravei sem querer no desktop, tudo bem,

vamos para o desktop.

Cd desktop, e daí posso repetir aquele python3 soma.

Soma.py, eu errei alguma coisa.

Que é que ele falou aqui?

Nessa linha oito, na soma igual a soma mais valor ele falou,

'unindent does not match any outer indentation level'.

Eu errei alguma coisa no nível de indentação.

Vamos ver nessa linha aqui.

Eu acho que o problema foi que,

aqui eu coloquei espaços e aqui eu coloquei tab, eu misturei.

Isso é uma coisa que você nunca deve fazer.

Aqui eu coloquei tab e aqui eu coloquei vários espaços e, ele ficou muito confuso.

Você precisa sempre ser consistente, então se usou tab, use sempre tab.

Agora deve dar certo.

Repetindo.

Agora deu certo.

Digite uma sequência de valores terminados por zero.

Digite valor a ser somado, dez, e errei de novo, que é que eu errei?

Ele falou de novo, soma.py, na linha oito aqui,

soma recebe soma mais valor deu tipo de operação não suportada.

Eu tentei somar inteiro a string.

Como você pode ver,

isso tentou somar inteiro a string é porque provavelmente você esqueceu aqui no

input que o input sempre devolve string e você precisa converter para inteiro.

Não se esquecer disso.

Agora eu acho que vai dar certo.

Dez, 20, 30,

três, -345, 1399,

posso digitar vários números,

lembra que ele vai terminar quando eu digitar zero.

Legítima sequência de valores terminados por zero.

Vamos digitar zero.

Pronto.

Daí terminou e ele falou: a soma dos valores digitados é 1449.

Essa é uma forma de usar o comando while para ler vários

valores e ir fazendo alguma computação com esses valores.

Fica de lição de casa aí para você, tentar vez de fazer uma soma terminada por zero,

primeiro, você pergunta para o usuário quantos números tem essa sequência e daí

você faz uma repetição exatamente daquele número de números que o usuário digitou,

vez de ser terminado por zero, fica uma variante para você experimentar.

Uma variante que eu quero mostrar aqui, por exemplo,

se vez das somas a gente quisesse a multiplicação dos números,

o produto dos números, como que a gente faria?

Primeiro, obviamente mudar o nome da variável para produto, porque

é importante que os nomes das variáveis indiquem o que a variável significa,

então, mais importante mudar o nome da variável para produto,

nunca ter nome de variável errado.

E aqui, simplesmente vez de ser soma, multiplicação.

Agora, tem outra coisa importante, lembra que eu falei que a gente

inicializou a variável com o elemento neutro da soma.

Agora o elemento neutro do produto não é o zero, se eu colocar zero,

vai dar sempre zero.

Eu preciso inicializar com o elemento neutro do, da multiplicação que é o.

Então, se eu falar: digite uma sequência de valores terminados por zero,

digite valor a ser multiplicado.

O produto.

O produto dos valores digitados é: produto.

E agora vou dar save as, salvar como,

salvar como produto.py.

[ÁUDIO\_EM\_BRANCO] E

aqui eu posso executar o meu produto.py,

então, digite valor a ser multiplicado,

dois, três, quatro, cinco, seis,

terminado por zero, o produto dos valores digitados é zero.

Já deve dar para desconfiar o que é que está errado.

Esse último zero que eu digitei aqui também foi multiplicado,

então a gente vai precisar organizar esse código de jeito diferente.

Esse valor, quando ele chega zero não pode ser multiplicado.

Tem várias formas de corrigir isso, vou corrigir,

aproveitar para ilustrar como é que a gente faria isso se,

vez de digitar uma sequência de valores terminados por zero a gente quisesse que

o usuário definisse quantos números vão ter de ser digitados.

Por exemplo, vou guardar nessa variável i.

Então eu vou falar que i recebe,

vai ser inteiro, input, digite

o tamanho da sequência de

números.

O tamanho da sequência de números vai estar no i,

e se o while vez de ser valor diferente de zero a gente vai

basear esse while no i.

Vou fazer, enquanto, i é o valor máximo o

tamanho da sequência, vamos chamar de tamanho e daí, eu vou usar i aqui.

Enquanto i é menor que o

tamanho daí sim repete e eu vou inicializar esse i com zero.

Então i começa zero e eu vou contando de até no tamanho.

Uma outra coisa que eu não posso esquecer é de somar

no i a cada interação do meu while.

A cada interação o i vai receber o valor do i anterior somado de.

Vamos ver se deu certo agora.

Mais errinho aqui, uma sintaxe inválida.

Produto recebe na linha quatro, eu esqueci alguma coisa.

Vejamos que é que.

Alguma coisa que eu esqueci, o produto ali,

tamanho recebe, abri o parêntesis e esqueci de fechar o parêntesis.

Então pronto.

Digite o tamanho da sequência de números.

Vão ser cinco números, a serem multiplicados,

vão ser o dois, o três, o quatro, o cinco, seis,

pronto, são cinco números, dois, três, quatro, cinco.

A multiplicação desses números deu 720, então, esta

é uma forma de resolver o problema, tem outras formas de resolver esse problema.

Agora, vou deixar desafio para vocês aqui, para vocês fazerem aí casa,

que é o seguinte: vamos supor que eu tenho número inteiro grande, com muitos dígitos,

e eu quero que você faça programa que vai calcular a soma dos dígitos desse número.

Então, se o número é 6532,

ele vai somar seis mais cinco mais três mais dois.

E ele vai dizer: seis mais cinco é onze, com cinco, 16.

Então, dado isso aqui ele tem que dar

a soma dos dígitos que nesse caso aqui seria 16.

Como que você vai conseguir fazer isso?

Primeira coisa, você vai ter que, de alguma forma descascar os números,

pegar número a número, cada dígito separadamente,

então você precisa while que vai tirando cada número disso.

Como é que a gente pode fazer isso?

É muto fácil se a gente lembrar da divisão inteira.

Por exemplo, esse meu número 6532,

vou guardar isso aqui numa variável X, X recebe 6532.

Se eu pegar o resto da divisão por dez desse número,

ele dá dois, então, o resto da divisão por dez sempre devolve o último dígito,

o dígito ali menos significativo daquele número.

Então, num while, cada vez pegando o último dígito,

fazendo o porcento dez, você pega o último dígito,

então precisa de uma forma de jogar fora esse último dígito para, numa próxima

interação pegar aqui o três depois pegar o cinco e depois pegar o seis.

Como é que faço isso?

Faço isso usando divisão inteira.

Se eu fizer o X, a divisão inteira de X por dez, quanto dá?

Dá 653, note então,

é jeito de eu jogar fora o último dígito e pegar os dígitos seguintes.

Então vou precisar fazer laço,

onde eu vou sempre pegando o último dígito e somando numa soma.

E, depois que eu somei o último dígito eu tenho que jogar

fora aquele último dígito e aqui você tem os dois elementos para fazer isso,

pegar o último dígito e jogar fora o último dígito.

então fica aí, para vocês fazerem esse desafio casa.