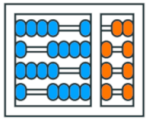


Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 – Aula 01

Introdução

Algoritmos e Programação de Computadores

Turmas

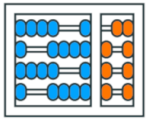
OVXZ

Prof. Lise R. R. Navarrete

lrommel@ic.unicamp.br

Quinta-feira, 17 de março de 2022

19:00h - 21:00h (CB06)



Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 – Algoritmos e Programação de Computadores

Turmas

OVXZ

https://ic.unicamp.br/~mc102/

Site da Coordenação de MC102

Aulas teóricas:

Terça-feira, 21:00h - 23:00h (CB06)

Quinta-feira, 19:00h - 21:00h (CB06)

Conteúdo

- Informações gerais
- Algoritmos e Programação de Computadores
- Organização Básica de Computadores
- História dos Computadores
- Ambiente Computacional
- Introdução a Python
- Primeira Aula de Laboratório
- Google Cloud Shell

Informações gerais

https://ic.unicamp.br/~mc102/

Material Didático

Materiais didáticos recomendados:

- Aula Introdutória [slides] [vídeo]
- Primeira Aula de Laboratório [slides] [vídeo]
- Python Básico: Tipos, Variáveis, Operadores, Entrada e Saída [slides] [vídeo]
- Comandos Condicionais [slides] [vídeo]
- Comandos de Repetição [slides] [vídeo]
- Listas e Tuplas [slides] [vídeo]
- Strings [slides]
- Dicionários [slides]

https://ic.unicamp.br/~mc102/

- Funções [slides]
- Objetos Multidimensionais [slides]
- Algoritmos de Ordenação [slides]
- Algoritmos de Busca [slides]
- Recursão [slides]
- Algoritmos de Ordenação Recursivos [slides]
- Arquivos [slides] (extra)
- Expressões Regulares [slides] (extra)
- Execução de Testes no Google Cloud Shell [slides] [vídeo] (extra)

https://ic.unicamp.br/~mc102/

Listas de Exercícios

- Primeira Lista - Tipos, Variáveis, Operações Matemáticas e Comandos Condicionais
- Segunda Lista - Comandos de Repetição
- Terceira Lista - Lista e Tuplas
- Quarta Lista - Strings
- Quinta Lista - Funções
- Sexta Lista - Recursão

https://ic.unicamp.br/~mc102/

Atividades Práticas

- As atividades práticas serão disponibilizados no SuSy.

https://susy.ic.unicamp.br:9999/mc102

https://ic.unicamp.br/~mc102/

Horários de Atendimentos e Laboratórios

A planilha com os horários de atendimentos e laboratórios pode ser consultada aqui.

Horários na cor amarela indicam laboratórios (exclusivos para alunos matriculados nas turmas indicadas) e horários na cor branca indicam plantões de atendimentos (abertos para todos os alunos matriculados em MC102).

Os números em cada horário se referem aos RAs dos monitores (PADs e PEDs) que realizarão o atendimento. A aba **Monitores** possui a lista com os nomes e RAs de todos os monitores.

Cada horário possui um link para uma sala do Google Meet, onde será realizado o atendimento ou o laboratório.

Reforçando: apenas alunos das turmas devem acessar os laboratórios específicos. Todos os alunos matriculados em MC102 podem acessar os horários de atendimentos, sempre usando a conta Google da Unicamp.

https://ic.unicamp.br/~mc102/

Início	Término	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
10:00	11:00	Turmas AB					Atendimento 187506, 224188, 235121
11:00	12:00						
12:00	13:00	Atendimento 118982, 187990, 191075, 230222, 234837	Atendimento 183573, 188511, 209823, 216698, 246997	Atendimento 165880, 167072, 173846, 216698, 226005	Atendimento 167072, 187506, 226005, 234837, 247361, 262884	Atendimento 182206, 187990, 209823, 230222, 246983	Atendimento 186531, 219591, 234921
13:00	14:00						
14:00	15:00	Turma F Turmas KL	Turma C	Turmas 45	Turmas GH		Atendimento 170561, 235121, 246983
15:00	16:00						
16:00	17:00	Turma E Turmas MN		Turma I		Turmas 67	Atendimento 182206, 186397, 230293
17:00	18:00						
18:00	19:00	Atendimento 118982, 170561, 191075, 219591, 246997	Atendimento 118982, 183573, 218592, 230293, 262884	Atendimento 165880, 183573, 218592, 234921, 246997, 247361	Atendimento 165880, 173846, 188511, 191075, 262884	Atendimento 170561, 173846, 219591, 234921, 247361	Atendimento 188511, 230293
19:00	20:00		Turma O			Turma VX	
20:00	21:00						
21:00	22:00					Turma Z	
22:00	23:00						

https://ic.unicamp.br/~mc102/

Avaliação

A avaliação será realizada através de tarefas práticas, ou seja, programas em Python que deverão ser implementados e submetidos para correção automática através do SuSy.

Serão propostas n tarefas práticas, que deverão ser realizadas pelos alunos. Os programas desenvolvidos serão testados com um conjunto pré-determinado de testes, subdividido em testes abertos (que podem ser acessados pelos alunos) e testes fechados (que não podem). A nota de cada atividade prática será proporcional ao número de testes, abertos ou fechados, que executarem corretamente.

Juntamente com o enunciado de cada tarefa prática será indicado o peso $P_i \in \{1, 2, 3, 4\}$ dessa tarefa. A média das tarefas práticas (P) será a média ponderada das notas das tarefas.

A média final F e a situação de cada aluno serão definidas de acordo com as regras a seguir.

- Caso $P \geq 5$:

O aluno estará aprovado por nota e frequência com média final (F):

```
F = P
```

https://ic.unicamp.br/~mc102/

- Caso $2{,}5 \leq P < 5$:

O aluno poderá realizar o exame. O exame será composto por um subconjunto das tarefas práticas disponibilizadas ao longo do semestre. O aluno poderá refazer as tarefas indicadas no período do exame ou aproveitar os programas já submetidos anteriormente naquelas tarefas (sem necessidade de refazer a tarefa).

A nota do exame (E) será calculada como a média ponderada das tarefas selecionadas para compor o exame. Os pesos das tarefas para fins do exame poderão ser diferentes daqueles previamente utilizados para o cálculo da média das tarefas práticas (P). O cálculo da média final (F) será feita da seguinte forma:

```
F = min{5, (P + E)/2}
```

Caso $F \geq 5.0$ o aluno estará aprovado por nota e frequência. Caso contrário, estará reprovado por nota.

- Caso $P < 2{,}5$:

O aluno estará reprovado por nota, com média final (F):

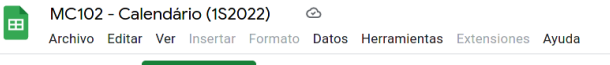
```
F = P
```

https://ic.unicamp.br/~mc102/

Observações:

1. Não haverá atividades práticas substitutivas.
2. As tarefas práticas serão disponibilizadas às segundas-feiras de manhã e ficarão disponíveis por 3 semanas (até domingo a noite).
3. Qualquer tentativa de plágio ou fraude nas atividades práticas implicará em nota final F = 0 (zero) para todos os envolvidos, sem prejuízo de outras sanções. Exemplos de plágios e fraudes:
   - Cópia ou compra de programas.
   - Submissão de programas que produzam as saídas esperadas dos testes abertos a partir da comparação de trechos da entrada, sem de fato implementar os algoritmos solicitados nas tarefas práticas.
4. Casos de tentativa de plágio ou fraude nas tarefas práticas poderão ser detectadas automaticamente entre todas as submissões (de todos os alunos, de todas as turmas) ao longo do semestre.
5. Todos os casos de tentativa de plágio ou fraude automaticamente detectados serão verificados manualmente pela coordenação da disciplina até o final do semestre.
6. As notas das atividades práticas serão divulgadas no site da disciplina até, no máximo, uma semana após o final do prazo de submissão das tarefas.
7. De acordo com a fórmula acima, caso um aluno seja aprovado após realizar o exame final, sua nota final será F = 5 (cinco).

https://ic.unicamp.br/~mc102/



Feriados | Avaliação de curso | Exame
Dias com Aulas | Dias sem Aulas

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Sem	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**	**Sábado**	**#Aulas**			**Conteúdo das Aulas**				**Atividades Práticas**
1	14/03/2022	15/03/2022	16/03/2022	17/03/2022	18/03/2022	19/03/2022	**SS**	**TQ**	**QS**	Apresentação da Disciplina	**Número**			**Conteúdo**
2	21/03/2022	22/03/2022	23/03/2022	24/03/2022	25/03/2022	26/03/2022	3	3	4	Tipos e operações	01			Entrada e Saída
3	28/03/2022	29/03/2022	30/03/2022	31/03/2022	01/04/2022	02/04/2022	5	5	6	Comandos Condicionais	01	02		Tipos e Operações
4	04/04/2022	05/04/2022	06/04/2022	07/04/2022	08/04/2022	09/04/2022	7	7	8	Comandos de Repetição	01	02	03	Comandos Condicionais
5	11/04/2022	12/04/2022	13/04/2022	14/04/2022	15/04/2022	16/04/2022	8	8	9	Comandos de Repetição	04	02	03	Comandos de Repetição
6	18/04/2022	19/04/2022	20/04/2022	21/04/2022	22/04/2022	23/04/2022	9	9	10	Listas e Tuplas	04	05	03	Comandos de Repetição
7	25/04/2022	26/04/2022	27/04/2022	28/04/2022	29/04/2022	30/04/2022	11	11	12	Listas e Tuplas	04	05	06	Listas e Tuplas
8	02/05/2022	03/05/2022	04/05/2022	05/05/2022	06/05/2022	07/05/2022	13	13	14	Strings e Dicionários	07	05	06	Listas e Tuplas
9	09/05/2022	10/05/2022	11/05/2022	12/05/2022	13/05/2022	14/05/2022	15	15	16	Funções	07	08	06	Strings e Dicionários
10	16/05/2022	17/05/2022	18/05/2022	19/05/2022	20/05/2022	21/05/2022	17	17	18	Matrizes	07	08	09	Strings e Dicionários
11	23/05/2022	24/05/2022	25/05/2022	26/05/2022	27/05/2022	28/05/2022	19	18	20	Matrizes	10	08	09	Matrizes
12	30/05/2022	31/05/2022	01/06/2022	02/06/2022	03/06/2022	04/06/2022	21	20	22	Ordenação e Busca	10	11	09	Matrizes
13	06/06/2022	07/06/2022	08/06/2022	09/06/2022	10/06/2022	11/06/2022	23	22	24	Recursão	10	11	12	Matrizes
14	13/06/2022	14/06/2022	15/06/2022	16/06/2022	17/06/2022	18/06/2022	24	23	25	Recursão	13	11	12	Ordenação e Busca
15	20/06/2022	21/06/2022	22/06/2022	23/06/2022	24/06/2022	25/06/2022	26	25	27	Recursão	13	14	12	Recursão
16	27/06/2022	28/06/2022	29/06/2022	30/06/2022	01/07/2022	02/07/2022	28	27	29	Algoritmos de Ordenação Recursivos	13	14	15	Recursão
17	04/07/2022	05/07/2022	06/07/2022	07/07/2022	08/07/2022	09/07/2022	30	29	31	Revisão / Tópicos Opcionais		14	15	Pré-Exame
18	11/07/2022	12/07/2022	13/07/2022	14/07/2022	15/07/2022	16/07/2022	32	31	33	Revisão / Tópicos Opcionais			15	Pré-Exame
19	18/07/2022	19/07/2022	20/07/2022	21/07/2022	22/07/2022	23/07/2022								Exame
20	25/07/2022	26/07/2022	27/07/2022	28/07/2022	29/07/2022	30/07/2022								

https://ic.unicamp.br/~mc102/

Programação em Python:

- Google Cloud Shell
- Python Online (Repl.it)
- The Python Tutorial
- The Python Language Reference
- Python Programming Examples
- Python Tutorial for Beginners
- LearnPython.org

https://ic.unicamp.br/~mc102/

Ambientes de Desenvolvimento Integrado (IDE) para Python:

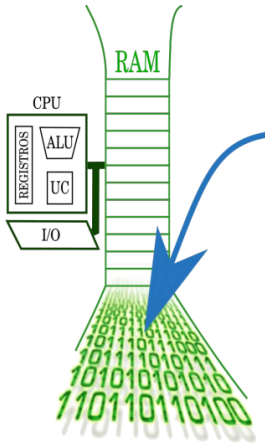
- PyCharm
- Visual Studio Code
- Jupiter
- Atom
- Spyder

Algoritmos e Programação de Computadores

https://ic.unicamp.br/~mc102/aulas/aula01.pdf

Algoritmos e Programação de Computadores

- Algoritmo é uma sequência bem definida de passos para realizar uma dada tarefa.
- Programa é uma sequência de comandos que indicam as operações que um computador deve executar para realizar uma dada tarefa.
- Programação é o processo de escrita, teste e manutenção de um programa de computador.
- Programação é uma habilidade importante para qualquer engenheiro ou cientista:
  - Ajuda a exercitar a capacidade de resolução de problemas.
- Exemplos de aplicações:
  - Desenvolvimento de ferramentas computacionais.
  - Automatização de processos industriais.
  - Simulação de modelos científicos.



Estruturas de dados

DADOS

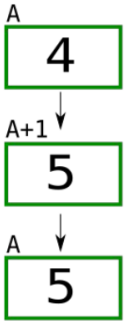
um inteiro
um real
um pixel
um som

Programa

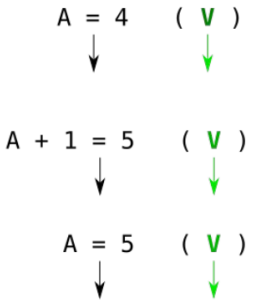
```
// criar

// inicializar
A = 4
// operação, pela lógica, correta
A = A + 1
// mostrar o resultado
print (A)
```

Programa

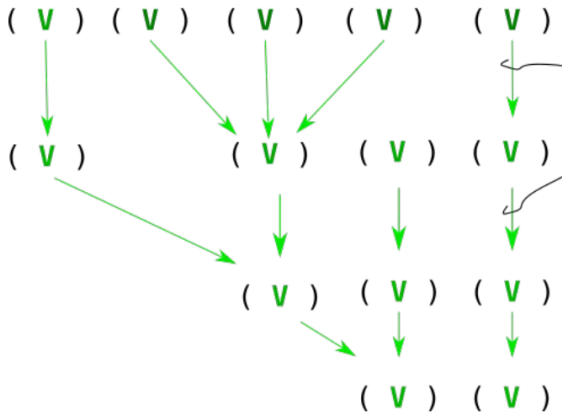
```
// criar

// inicializar
A = 4;
// operação, pela lógica, correta
A = A + 1;
// mostrar o resultado
print(A)
```



Programar



Programar

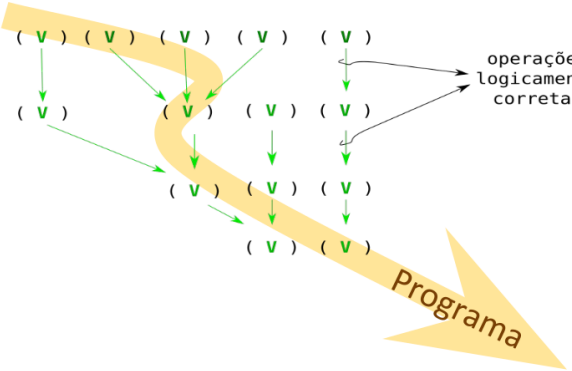


operações logicamente corretas

operações que são implicações lógicas

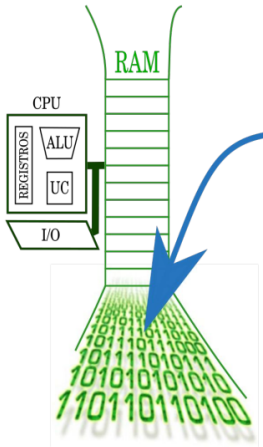
operações válidas

operações corretas



Programar

1. Resolve meu problema?
2. Existe outra maneira melhor?



Estruturas de dados

DADOS

um inteiro
um real
um pixel
um som

Variável
(na linguagem de programação)

é um **mecanismo**
para poder **manipular**
uma **região de memória**
onde será **armazenado**
um **dado**.

Variável
(na linguagem de programação)

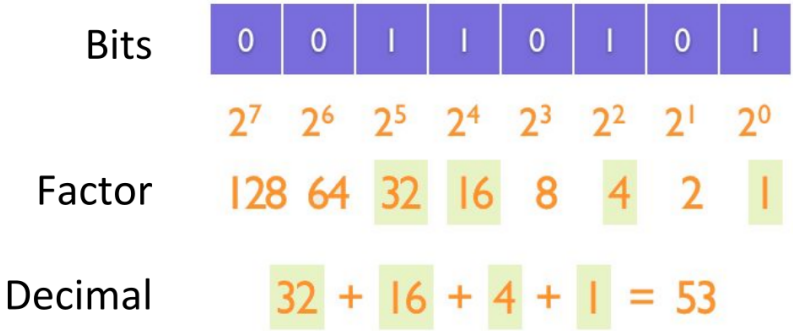
é um **mecanismo**
para poder **manipular**
uma **região de memória**
onde será **armazenado**
um **dado**.

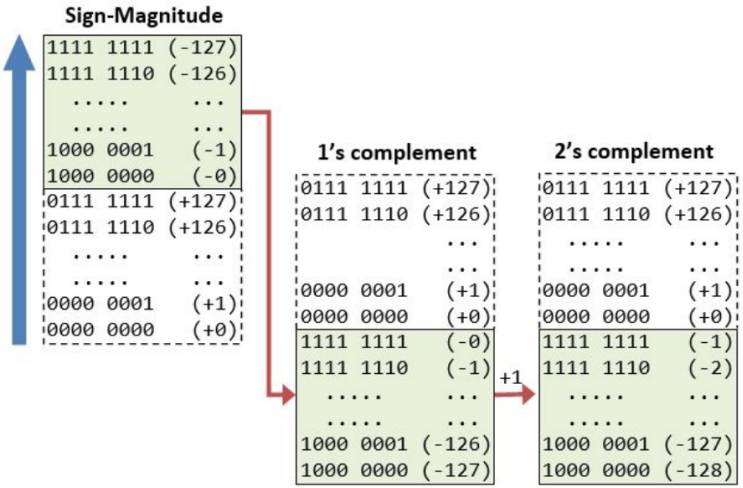
Possui um **identificador**
`A`

Possui um **tipo de dado**
determina um tamanho

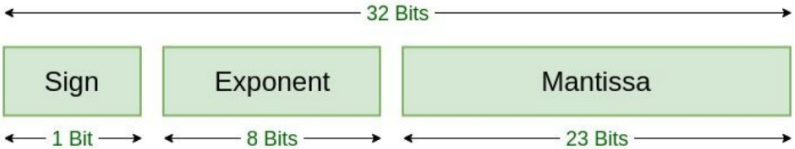
Está associada a um **endereço**
memória RAM

inteiros


Bits
0 0 1 1 0 1 0 1
2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
Factor
128 64 32 16 8 4 2 1
Decimal
32 + 16 + 4 + 1 = 53



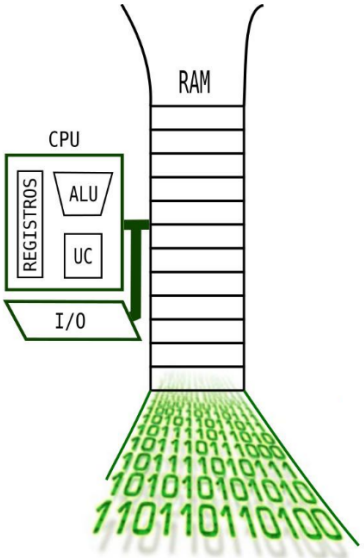
Ver: https://www3.ntu.edu.sg/home/ehchua/programming/java/datarepresentation.html



Single Precision
IEEE 754 Floating-Point Standard

o que é um programa?

Uma maneira simples de entender **o que é um programa?** é imaginá-lo como uma **sequência de operações**, pela lógica, corretas, que **transformam regiões de memória RAM**, pelo uso de variáveis, transformando bits em outros bits

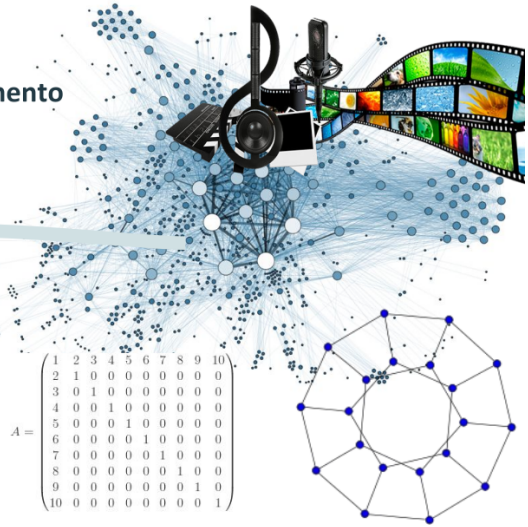


Programação

Estruturas de dados

São formas de **organizar o armazenamento** de **dados na memória** do computador com o objetivo de fazer mais **eficiente** seu uso.

vetores
listas
pilhas
....



$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Arquivo de texto

ou arquivo de **texto plano**, é um arquivo composto por **Bytes** que representam diretamente **símbolos de texto**, sob alguma **codificação** base:

ASCII,
UTF-8,
UNICODE,
...

Low Ascii

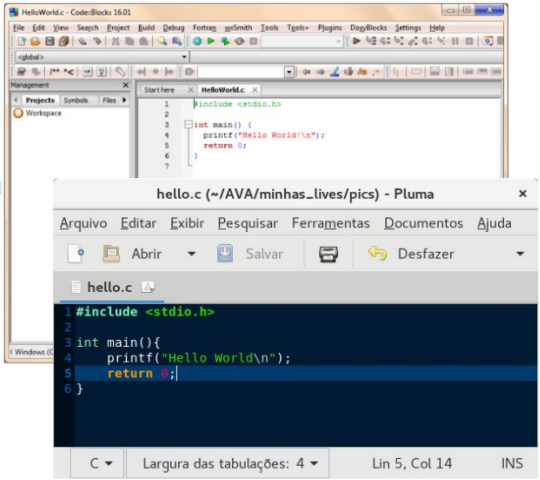
000:	013:♪	026:→	039:'	052:4	065:A	078:N	091:[	104:h	117:u
001:☺	014:♫	027:←	040:(	053:5	066:B	079:O	092:\	105:i	118:v
002:☻	015:☼	028:∟	041:)	054:6	067:C	080:P	093:]	106:j	119:w
003:♥	016:►	029:↔	042:*	055:7	068:D	081:Q	094:^	107:k	120:x
004:♦	017:◄	030:▲	043:+	056:8	069:E	082:R	095:_	108:l	121:y
005:♣	018:↕	031:▼	044:,	057:9	070:F	083:S	096:`	109:m	122:z
006:♠	019:‼	032:	045:-	058::	071:G	084:T	097:a	110:n	123:{
007:•	020:¶	033:!	046:.	059:;	072:H	085:U	098:b	111:o	124:|
008:◘	021:§	034:"	047:/	060:<	073:I	086:V	099:c	112:p	125:}
009:○	022:▬	035:#	048:0	061:=	074:J	087:W	100:d	113:q	126:~
010:◙	023:↨	036:$	049:1	062:>	075:K	088:X	101:e	114:r	127:⌂
011:♂	024:↑	037:%	050:2	063:?	076:L	089:Y	102:f	115:s	
012:♀	025:↓	038:&	051:3	064:@	077:M	090:Z	103:g	116:t	

High Ascii

128:Ç	141:ì	154:Ü	167:º	180:┤	193:┴	206:╬	219:█	232:δ	245:J
129:ü	142:Ä	155:¢	168:¿	181:╡	194:┬	207:╧	220:▄	233:Θ	246:÷
130:é	143:Å	156:£	169:⌐	182:╢	195:├	208:╨	221:▌	234:Ω	247:≈
131:â	144:É	157:¥	170:¬	183:╖	196:─	209:╤	222:▐	235:δ	248:°
132:ä	145:æ	158:₧	171:½	184:╕	197:┼	210:╥	223:▀	236:∞	249:·
133:à	146:Æ	159:ƒ	172:¼	185:╣	198:╞	211:╙	224:α	237:ø	250:·
134:å	147:ô	160:á	173:¡	186:║	199:╟	212:╘	225:ß	238:ϵ	251:√
135:ç	148:ö	161:í	174:«	187:╗	200:╚	213:╒	226:Γ	239:∩	252:ⁿ
136:ê	149:ò	162:ó	175:»	188:╝	201:╔	214:╓	227:π	240:≡	253:²
137:ë	150:û	163:ú	176:░	189:╜	202:╩	215:╫	228:Σ	241:±	254:■
138:è	151:ù	164:ñ	177:▒	190:╛	203:╦	216:╪	229:σ	242:≥	255:
139:ï	152:ÿ	165:Ñ	178:▓	191:┐	204:╠	217:┘	230:μ	243:≤	
140:î	153:Ö	166:ª	179:│	192:└	205:═	218:┌	231:τ	244:⌠	

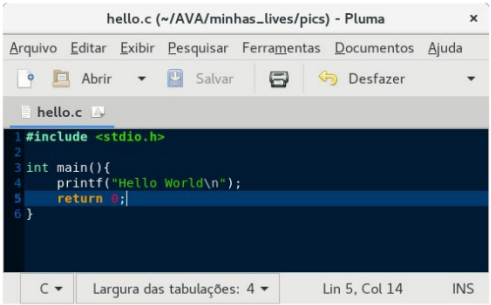
Editor de texto

É uma **ferramenta computacional** (ou **programa**) que permite **criar** e **modificar arquivos de texto**.



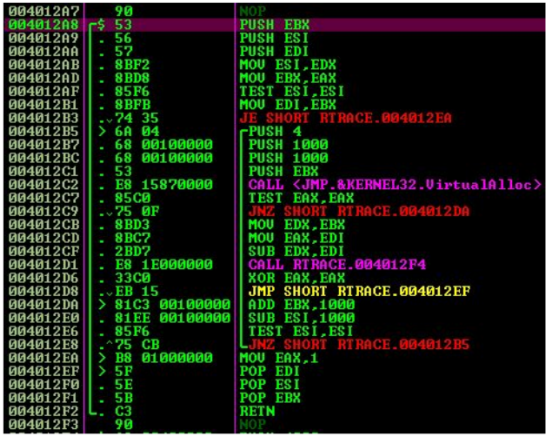
Arquivo fonte

É um **arquivo de texto** que representa um **programa**.
Está escrito seguindo a **sintaxe** de alguma **linguagem de programação**.



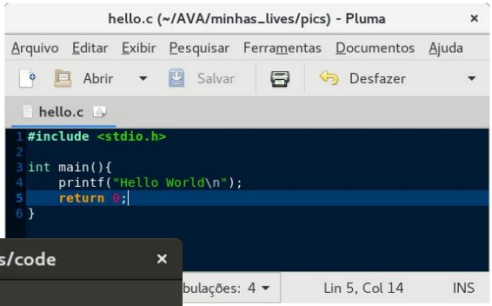
Arquivo executável

É um arquivo que contém **Bytes** que representam diretamente **instruções na linguagem maquina** e que pode ser **executado diretamente** pelo **sistema operacional**.



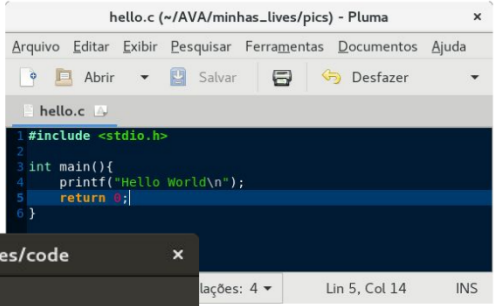
Compilação

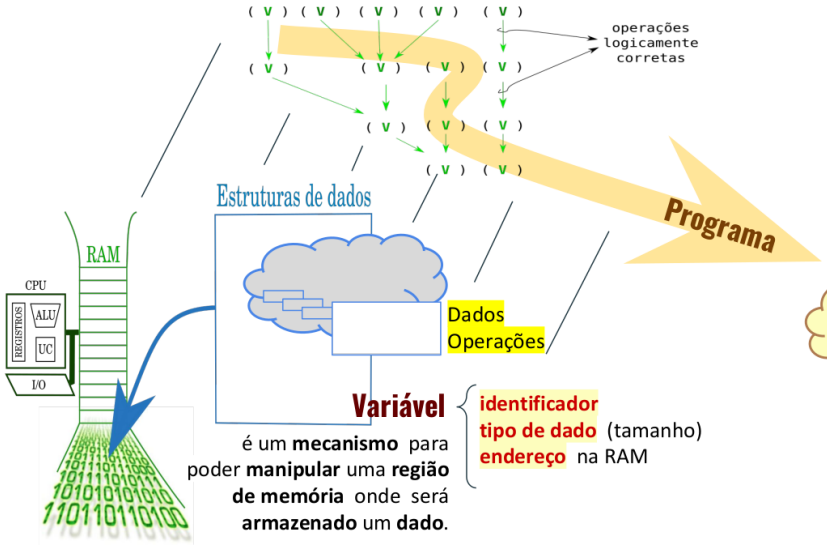
É o processo de **convertir** um **arquivo fonte** num **arquivo executável**.



Execução

pedir ao sistema operacional
a **execução** de
um **arquivo executável**




(V) (V) (V) (V) (V)
(V) (V) (V) (V)
(V) (V) (V)
(V) (V)
operações logicamente corretas
Programa
Estruturas de dados
Dados
Operações
RAM
CPU
REGISTRAS
ALU
UC
I/O
Variável
é um mecanismo para poder manipular uma região de memória onde será armazenado um dado.
identificador
tipo de dado (tamanho)
endereço na RAM
1 Corretude?
2 Eficiência?

Organização Básica de Computadores

- Um computador é uma máquina que, a partir de uma entrada, realiza um processamento sobre as informações e gera uma saída.
- Um computador normalmente é utilizado para executar tarefas extensas, complexas e repetitivas que, caso fossem realizadas manualmente, exigiriam um tempo muito maior e estariam sujeitas a erros.

- Hardware corresponde aos componentes físicos que compõem o computador, tais como unidade central de processamento (CPU), memória e dispositivos de entrada e saída (monitor, teclado, mouse, etc).
- Software corresponde aos programas que executam tarefas utilizando o hardware do computador, tais como sistema operacional, aplicativos e bibliotecas.

- Os computadores digitais operam com dois níveis de tensão, sendo o sistema binário de enumeração o mais natural.
- Bit (*binary digit*) é a menor unidade de informação que pode ser armazenada ou transmitida: pode assumir valores 0 ou 1.
- Byte: agrupamento de 8 bits em uma palavra.

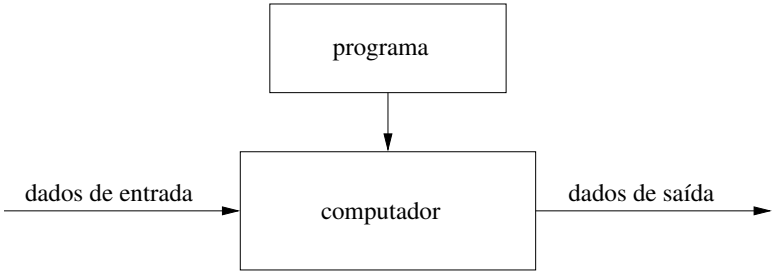
- Prefixos binários são nomes ou símbolos que precedem unidades de medidas, tais como bits ou bytes, para indicar a sua multiplicação por potências de dois.
- Geralmente estão associados a sistemas digitais, como computadores e dispositivos digitais de comunicação e de armazenamento de dados.
- Principais prefixos binários:
  - K (kilo) $= 2^{10} \approx 10^3$
  - M (mega) $= 2^{20} \approx 10^6$
  - G (giga) $= 2^{30} \approx 10^9$
  - T (tera) $= 2^{40} \approx 10^{12}$
  - P (peta) $= 2^{50} \approx 10^{15}$

- Programas são compostos por um conjunto de instruções que operam o hardware, como operações lógicas e aritméticas.
- Temos abaixo, por exemplo, três instruções para um computador de 32 bits:

```
01000010 00110101 01010100 00110110
01001110 11001100 10010110 01101000
00000101 11111110 11010011 00001100
```

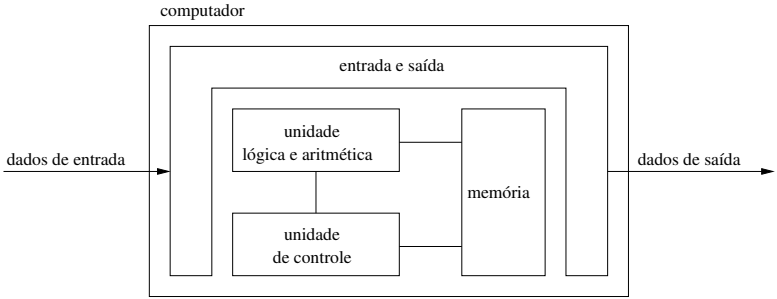
- Um software é composto por milhares de instruções deste tipo.

- Modelo de Turing: a partir de um programa, o computador pode processar os dados de entrada e gerar dados de saída.



Modelo de Alan Turing (1936)

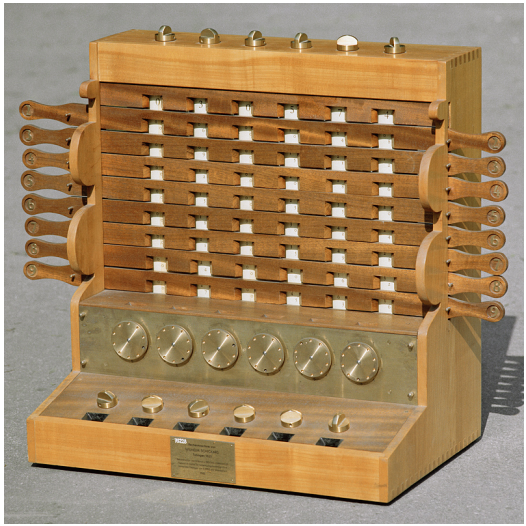
- Modelo de von Neumann: um computador é dividido em quatro componentes principais: dispositivos de entrada e saída, unidade lógica e aritmética, memória e unidade de controle.
- Os programas são armazenados na memória do computador.



Arquitetura de John von Neumann (1946)

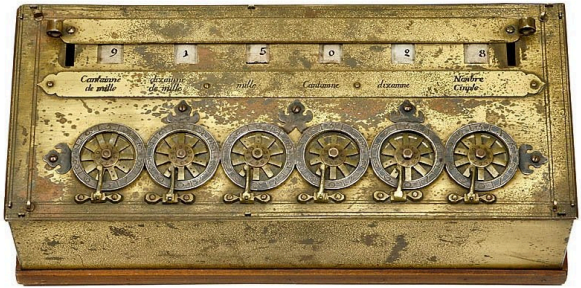
História dos Computadores

- Em 1623, Wilhelm Schickard construiu a primeira máquina de calcular mecânica, capaz de realizar as operações básicas de adição e subtração para números de seis dígitos.



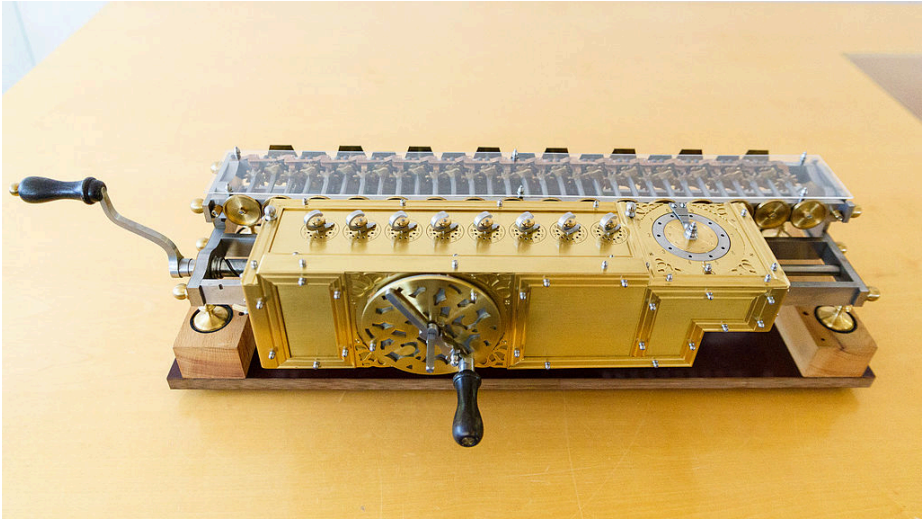
Réplica da máquina de Schickard

- Em 1642, Blaise Pascal inventou a calculadora mecânica chamada Pascaline, que realizava operações básicas de adição e subtração até oito dígitos.



Réplica da Pascaline

- Em 1673, Gottfried Leibniz aperfeiçou a máquina de Pascal e criou uma calculadora mecânica, conhecida como Roda de Leibnitz, que realizava operações de adição, subtração, multiplicação e divisão.

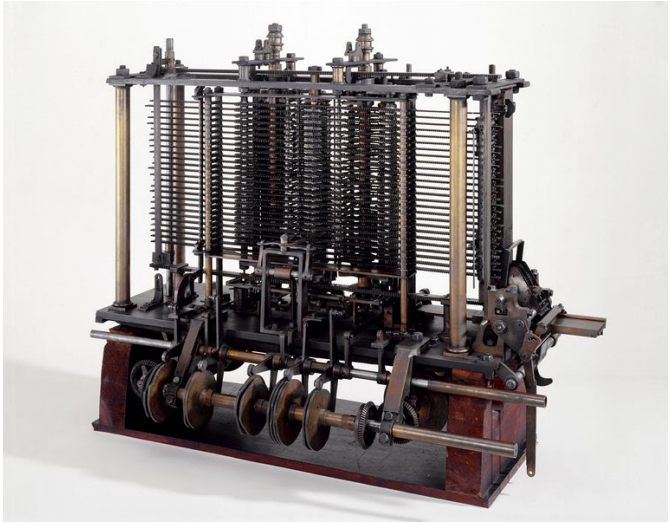


Réplica da Roda de Leibniz

- Em 1801, Joseph-Marie Jacquard inventou um tear mecânico controlado por cartões perfurados. O equipamento pode ser considerado como a primeira máquina mecânica programável da história, em que os cartões forneciam os comandos necessários para a tecelagem dos padrões nos tecidos.

Réplica da máquina de Jacquard

- Em 1822, Charles Babbage projetou a máquina diferencial para cálculos com polinômios.
- Em 1835, Charles Babbage projetou a máquina analítica, que é um projeto de computador mecânico programável de uso geral empregando cartões perfurados para a entrada de dados e uma máquina a vapor para fornecimento de energia.



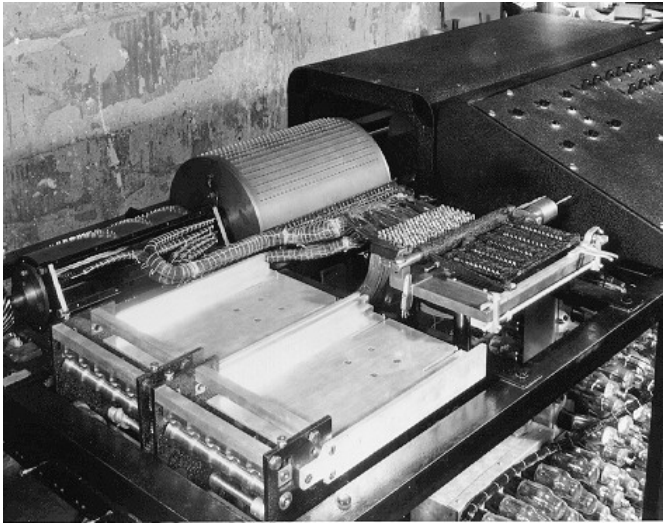
Réplica da máquina analítica de Babbage

- Em 1890, Herman Hollerith construiu uma máquina programável capaz de ler e processar dados armazenados em cartões perfurados. A máquina foi utilizada para auxiliar o censo de 1890.
- Hollerith foi um dos fundadores da International Business Machines (IBM).

Réplica da máquina de Hollerith

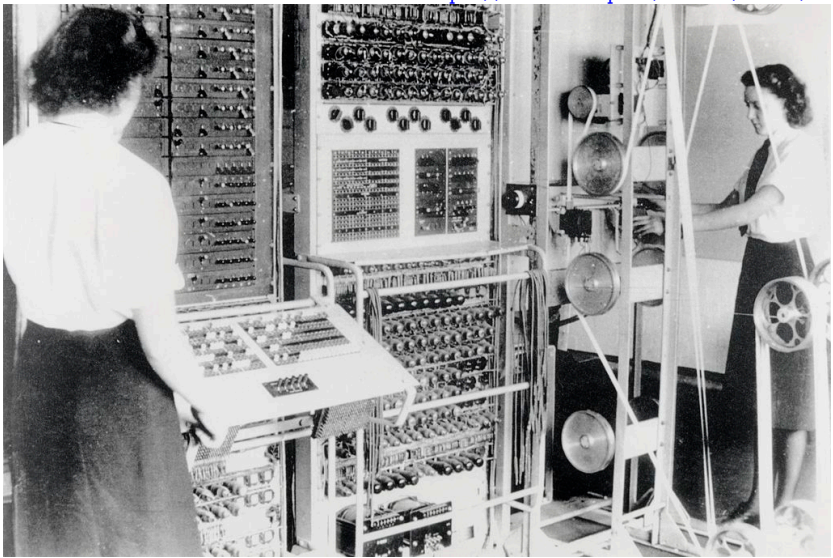
- Em 1936, Alan Turing desenvolveu um modelo teórico de um computador (chamado de "máquina universal") , restrito aos aspectos lógicos do seu funcionamento (memória, estados e transições). A ideia de computabilidade, ou seja, a definição de quais problemas poderiam ser resolvidos por um computador, começou a ser delineada.
- Em 1938, Konrad Zuse construiu o primeiro computador eletromecânico completamente funcional, conhecido como Z1. A máquina usava relés que executavam os cálculos e dados lidos em fitas perfuradas e utilizava o sistema binário de numeração.

- Em 1942, John Atanasoff e seu assistente Clifford Berry construíram o primeiro computador eletrônico digital, conhecido como ABC (Atanasoff-Berry Computer).
- O computador foi projetado originalmente para resolver um sistema de equações lineares.



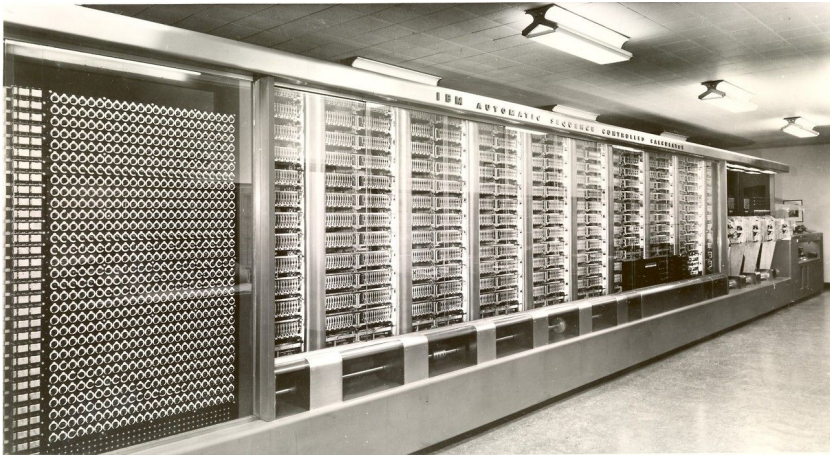
ABC

- Em 1944, Allan Turing ajudou a construir o computador Colossus, projetado para decifrar códigos secretos dos alemães durante a segunda guerra mundial, conhecidos como Enigma Alemão.



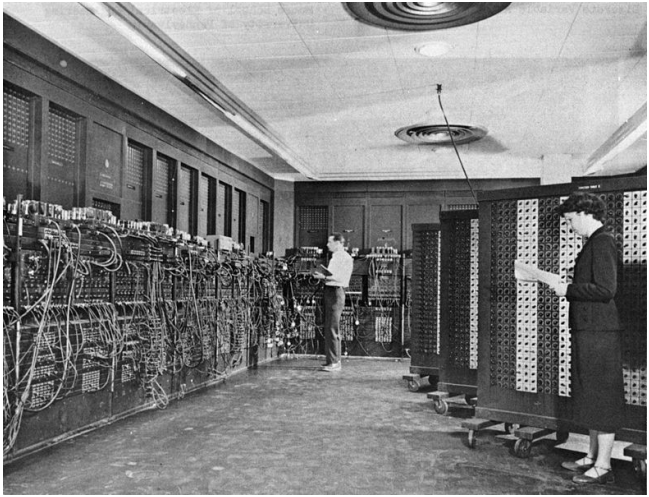
Colossus

- Em 1944, a Marinha dos Estados Unidos, a Universidade de Harvard e a IBM desenvolveram um computador conhecido como Mark I, com base na máquina analítica de Babbage.
- O computador utilizava componentes elétricos e mecânicos, funcionava com relés e era programado por fita de papel. Possuía 10m de comprimento, 2m de largura e pesava 70 toneladas.
- O Mark I foi projetado para calcular trajetórias balísticas de canhões de longo alcance.



Mark I

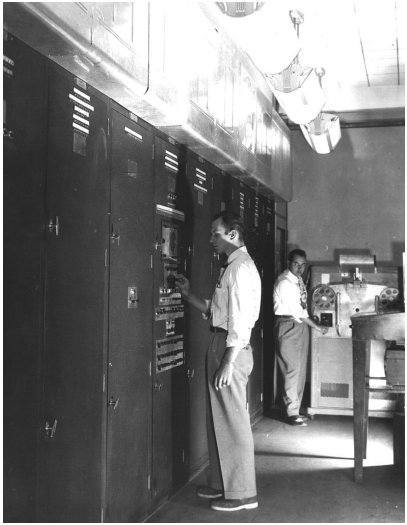
- Em 1946, o Exército dos Estados Unidos desenvolveu o computador eletrônico ENIAC (*Eletronic Numeric Integrator And Calculator*).
- O computador utilizava 18000 válvulas, possuía cerca de 30m de comprimento e 3m de largura, pesava 30 toneladas e consumia 178 kW de energia.
- Foi projetado para calcular trajetórias balísticas de mísseis.
- O programador tinha que conectar um grande número de fios, relés e sequências de chaves para definir códigos a serem executados.



ENIAC

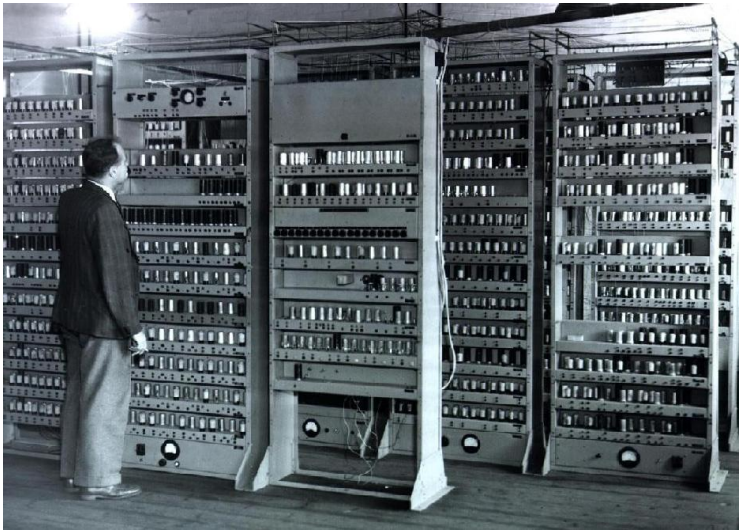
- Em 1946, John von Neumann propôs que um programa fosse armazenado em um computador da mesma forma que os dados. Esta proposta, chamada de "Arquitetura de von Neumann", é a base para os computadores programáveis modernos e é composta por 3 características principais:
  - Codificação das instruções de modo a serem armazenadas na memória do computador;
  - Armazenamento em memória das instruções e de toda e qualquer informação necessária na execução da tarefa;
  - Busca das instruções, a cada passo do processamento, diretamente na memória, e não nos então utilizados cartões perfurados.

- Em 1947, John von Neuman, John Eckert e John Mauchly começaram a trabalhar em uma versão melhorada do ENIAC, denominada EDVAC (*Electronic Discrete Variable Automatic Computer*), que incorporou o conceito de armazenamento de programas em memória.
- O EDVAC usava memórias baseadas em linhas de retardo de mercúrio, com maior capacidade de armazenamento.



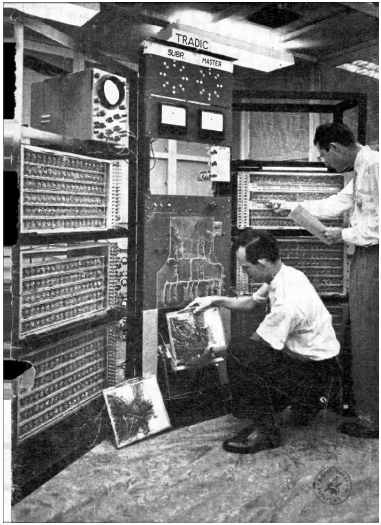
EDVAC

- Em 1949, foi construído o EDSAC (*Electronic Delay Storage Automatic Calculator)*, outro computador que armazenava programas em memória.



EDSAC

- Em 1955, os laboratórios da AT&T Bell anunciam a construção do TRADIC (*Transistorized Airborne Digital Computer)*, o primeiro computador totalmente transistorizado.
- Ele possuía aproximadamente 800 transistores ao invés das antigas válvulas, o que permitia trabalhar com menos de 100W de consumo de energia.



TRADIC

- Em 1958, Jack Kilby desenvolveu um dos primeiros circuitos integrados, contendo 5 componentes em uma peça de germânio com meia polegada de comprimento. Esses circuitos são um conjunto de transistores, resistores e capacitores construídos sobre uma base de silício (material semicondutor).
- Em 1969, a agência americana ARPA (*Advanced Research and Projects Agency*) desenvolveu a rede ARPANET, cujo objetivo era interligar as bases militares e os departamentos de pesquisa do governo americano. Esta rede iniciou dentro do Pentágono e foi a precursora da Internet.
- Em 1969, foi lançado do Kenbak-1, considerado o primeiro microcomputador (computador pessoal).
- Em 1971, Ray Tomlinson implementou um sistema de correio eletrônico (e-mail) na ARPANET.

- Em 1972, Alan Kay descreveu uma proposta de um dispositivo portátil (chamado "Dynabook"), precursor dos atuais *notebooks* ou *laptops*.
- Em 1973, Robert Metcalfe criou o sistema de conectividade Ethernet para interligação de computadores em redes locais no centro de pesquisa da Xerox Corporation, em Palo Alto (EUA).
- Em 1975, Bill Gates e Paul Allen fundaram a Microsoft Corporation.
- Em 1976, Steve Jobs, Steve Wozniak e Ronald Wayne fundaram a Apple Computer, Inc.
- Em 1977, a Apple lançou o microcomputador Apple II.

Microcomputador Apple II

- Em 1981, a IBM lançou o microcomputador IBM PC (*Personal Computer*) 5150, que se tornou o padrão de computador pessoal.
- O computador possuia processador Intel 8088 de 4,77 MHz, 64 Kbytes RAM, uma unidade de disquetes de 5 1/4" (de até 720 Kbytes), sem disco rígido.
- A empresa Microsoft foi contratada para desenvolver o sistema operacional MS-DOS (*Microsoft Disk Operating System*).

Microcomputador IBM 5150

- Em 1984, a Apple lançou o computador pessoal Macintosh (Mac).
- Em 1989, a Apple lançou o *Macintosh Portable*, o primeiro computador com funcionamento por bateria.

Macintosh Portable

- Em 1993, a NSF (*National Science Foundation*) criou a InterNIC (*Internet Network Information Center*), uma organização do Departamento de Comércio dos Estados Unidos responsável pelo registro de domínios utilizados na Internet.
- Em 1993, a Intel batizou de *Pentium* a sua nova geração de processadores, os quais utilizavam registradores de 32 bits, com 3,1 milhões de transistores.
- Em 1993, a Apple lançou o primeiro PDA (*Personal Digital Assistant*), o pioneiro dos computadores de mão.
- Em 1997, o termo telefone inteligente (*smartphone*) foi utilizado pela Ericsson para descrever seu aparelho GS 88 Penelope.
- Em 1998, Larry Page e Sergey Brin, dois estudantes de doutorado da University de Stanford, criaram a Google.

- Em 2001, a Apple lança o sistema operacional Mac OS X e o aparelho iPod.
- Em 2001, foi lançado nos Estados Unidos o aparelho Kyocera 6035, da Palm, Inc., um dispositivo que combina um PDA com um telefone celular, sendo considerado um dos primeiros *smartphones* do mercado.
- Em 2003, a Research in Motion Limited (RIM) lançou o *smartphone* BlackBerry.
- Em 2003, a plataforma aberta Android foi lançada por Andy Rubin, um dos fundadores da empresa Android, Inc., que foi comprada pela Google em 2005.
- Em 2007, a Apple lançou o iPhone, um dos primeiros telefones celulares com interface baseada em tela sensível a múltiplos toques.

- Em 2010, a Apple lançou o iPad, um dispositivo portátil em formato de prancheta (*tablet*) que pode ser utilizado para acesso à Internet e visualização de conteúdos digitais, entre outras finalidades.
- Em 2012, o Facebook alcança 1 bilhão de usuários.
- Em 2015, a Apple lançou o Apple Watch, um dos primeiros *smartwatches*.
- Em 2016, a Universidade de Maryland construiu o primeiro computador quântico reprogramável.
- Em 2017, a DARPA (Defense Advanced Research Projects Agency) começou o desenvolvimento de um computador molecular.

Ambiente Computacional

- Computadores realizam tarefas complexas por meio de um número tipicamente grande de operações simples.
- Para gerenciar a complexidade das soluções, um ambiente computacional é organizado como uma hierarquia de camadas, em que cada uma é responsável por uma tarefa específica.

Programas de Aplicação
Compiladores / Interpretadores
Sistema Operacional
Hardware

- Como usuários, interagimos com os programas de aplicação.
- Nesta disciplina, iremos construir novos programas de aplicação.
- Para construir novos programas, uma forma seria escrever códigos binários diretamente executados por um computador (hardware).
- Uma maneira mais simples é escrever os programas em uma linguagem de programação com nível mais alto de abstração.

- Uma linguagem de programação é um conjunto de comandos que são mais "próximos" da linguagem humana do que os sinais digitais.
- Nesta disciplina, usaremos a *linguagem de programação Python* (versão 3.8.2 ou superior).
- *Compiladores* e *Interpretadores* são programas que convertem um código em uma linguagem de programação em instruções em linguagem de máquina.
- Exemplo:

```
for i in range(10):
  c = a + b
```

```
LOOP: ADD c, a, b
      ADD i, i, 1
      BNQ i, 10, LOOP
```

```
01000010 00110101 01010100 00110110
01100110 01110101 01010100 00110110
11110000 01110101 01010100 00110110
```

- Um interpretador traduz o código linha a linha, apenas quando aquela linha de código precisar ser executada.
- Já o compilador traduz o programa inteiro em código de máquina de uma só vez, gerando um código executável.
- Durante a tradução o compilador gera um relatório de erros, caso existam, enquanto o interpretador interrompe a tradução para código de máquina somente quando encontra o primeiro erro em tempo de execução.
- Os códigos executáveis gerados por compiladores são mais rápidos do que os códigos interpretados.
- Correções e alterações são mais simples de serem feitas em códigos interpretados, que não exigem ser compilados antes de serem executados.
- Python é uma linguagem interpretada.

- Um sistema operacional é um conjunto de programas cuja função principal é gerenciar os recursos do sistema (memória, processador, discos, etc.).
- Um sistema operacional deve permitir o uso eficiente e seguro do hardware pelos usuários.
- Exemplos de sistema operacional:
  - Windows
  - Linux
  - Mac OS
  - MS-DOS
  - Android
  - iOS

Introdução a Python

https://pt.slideshare.net/jhoonb/introduo-python-mdulo-1

O que é Python?

Python é uma linguagem de programação de **propósito geral**, de **alto nível**, **interpretada**, **multiparadigma**, de **tipagem dinâmica e forte**.

https://pt.slideshare.net/jhoonb/introduo-python-mdulo-1

Porque usar Python?

- Linguagem muito parecida com pseudo-código executável;
- O uso de indentação para marcar blocos; (controverso)
- Coletor de lixo para gerenciar automaticamente o uso da memória;
- Fácil de escrever e ler.
- Programar se torna divertido e não tedioso.
- Standard Library extremamente poderosa.
- Comunidade produtiva, amigável e ativa!

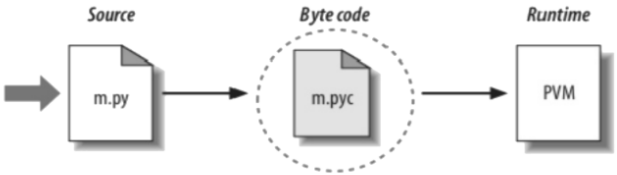
https://pt.slideshare.net/jhoonb/introduo-python-mdulo-1

Linguagem Interpretada.

Python é uma linguagem interpretada, isso quer dizer que o código fonte (.py) é lido linha a linha pelo programa interpretador, gera-se o bytecode pelo compilador (.pyc) e então executado pela VM de Python.

https://pt.slideshare.net/jhoonb/introduo-python-mdulo-1

Linguagem Interpretada.

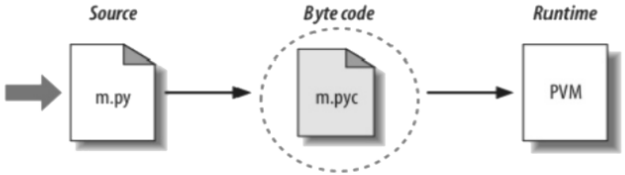


A linguagem é de altíssimo nível, mas ela também pode compilar seus programas para que a próxima vez que o executar não precise compilar novamente o programa.

.pyc: .py compilado
.pyo: .py otimizado e compilado

https://pt.slideshare.net/jhoonb/introduo-python-mdulo-1

Linguagem Interpretada.



Interpretado é mais lento que compilado...

logo Python, é lento? Ruim? Não presta?

Para a maioria dos nossos problemas ela é suficiente ou até melhor :)

Problema Prático

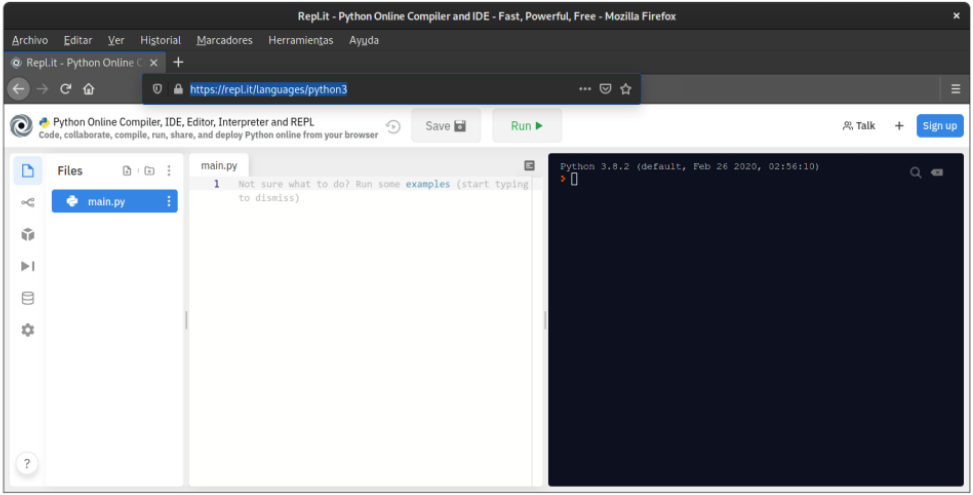
Implemente um programa que solicita a temperatura atual em graus Fahrenheit do usuário e exibe a temperatura em graus Celsius usando a fórmula

$$\texttt{celsius} = \frac{5}{9}(\texttt{fahrenheit} - 32)$$

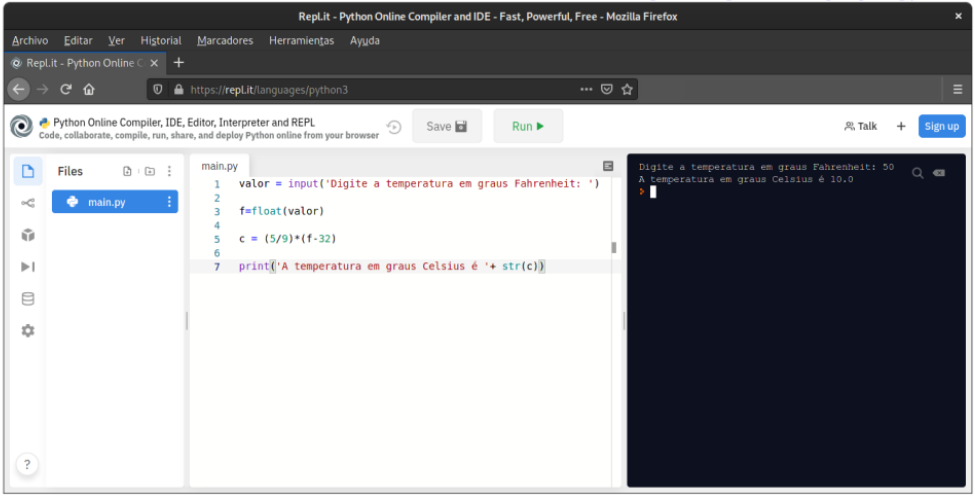
Seu programa deverá ser executado da seguinte forma:

```
>>>
Digite a temperatura em graus Fahrenheit: 50
A temperatura em graus Celsius é 10.0
```

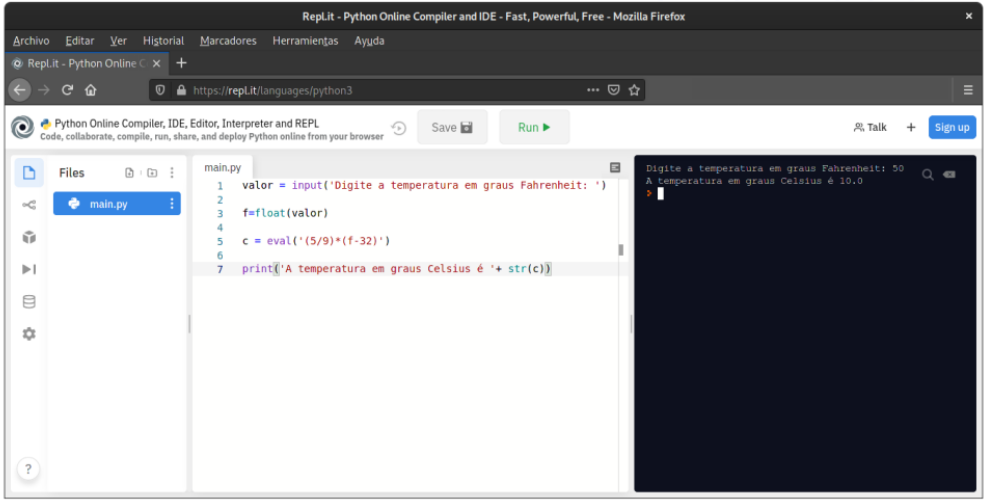
https://repl.it/languages/python3



https://repl.it/languages/python3



https://repl.it/languages/python3



Um programa em Python é um arquivo texto, contendo declarações e operações da linguagem. Este arquivo também é chamado de *código fonte*.

```
1 print("Hello World")
```

Você pode salvar este arquivo como hello.py.

Para executar um programa a partir do seu código fonte, você deve usar o seguinte comando em um terminal:

```
$ python hello.py
Hello World
```

Erros de execução ocorrem quando o comportamento do programa diverge do esperado.

```
1 print("Hello World)
```

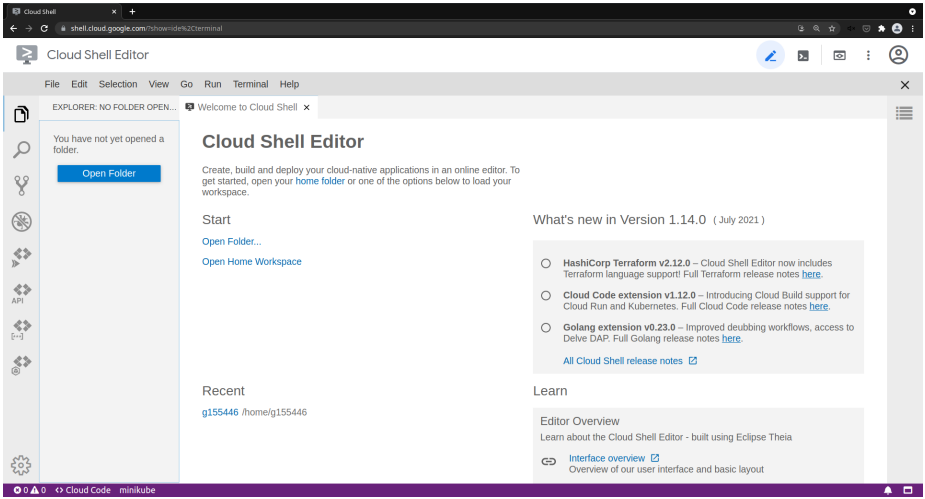
```
1 $ python hello.py
2   File "<stdin>", line 1
3     print("Hello World)
4                        ^
5 SyntaxError: EOL while scanning string literal
```

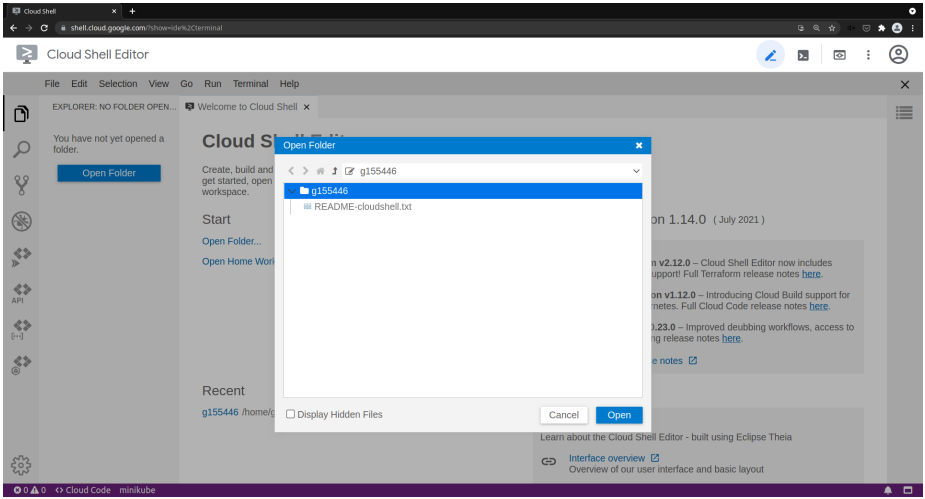
```
1  x = float(input("Qual o valor de x? "))
2  y = float(input("Qual o valor de y? "))
3
4  if (x == y):
5    print("Os dois valores são iguais: x = y =", x)
6  else:
7    if (x > y):
8      print("O maior valor é x =", x)
9    else:
10     print("O maior valor é y =", y)
```

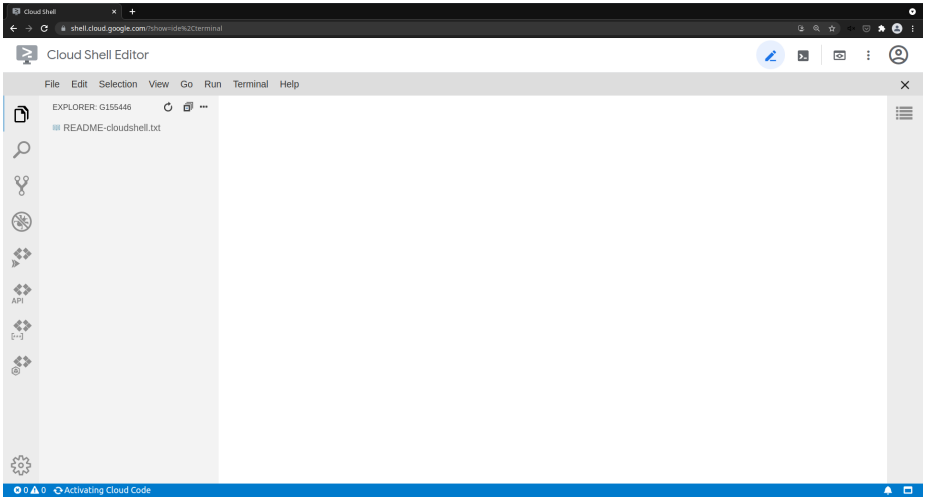
Primeira Aula de Laboratório

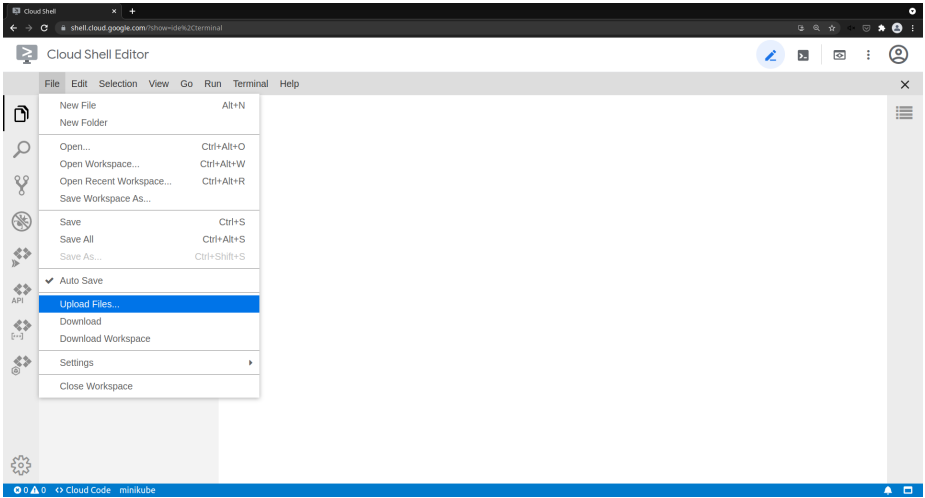
1. Acesse a página do SuSy:
   `https://susy.ic.unicamp.br:9999/mc102`
2. Clique na atividade prática “01 Aritmética com Inteiros”.
3. Na página da atividade prática clique em “Enunciado”.
4. Leia com cuidado todo o enunciado da atividade prática.
5. Na página da atividade prática clique em “Arquivos auxiliares” e realize o download do código base da atividade (lab01.py).

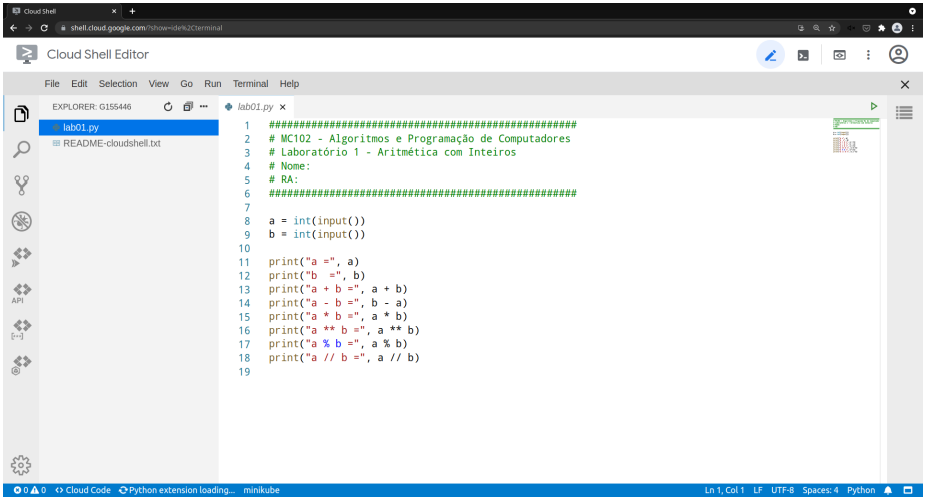
1. Acesse a página do Google Cloud Shell
   `https://shell.cloud.google.com`
2. Selecione a opção *Open Folder* que aparece à direita.
3. Uma janela mostrando uma pasta com o seu nome de usuário será exibida. Basta clicar em *Open* para abri-la como um *workspace* (área de trabalho).
4. Faça upload do arquivo base (lab01.py) no Google Cloud Shell.
5. Importante: verifique se o nome do arquivo é lab01.py (e não tem outra extensão, por exemplo, lab01.py.txt). Se o arquivo foi salvo com outro nome ou extensão, renomei para lab01.py (isso pode ser feito no próprio Google Cloud Shell).
6. No código, preencha o seu nome e RA nas linhas indicadas.

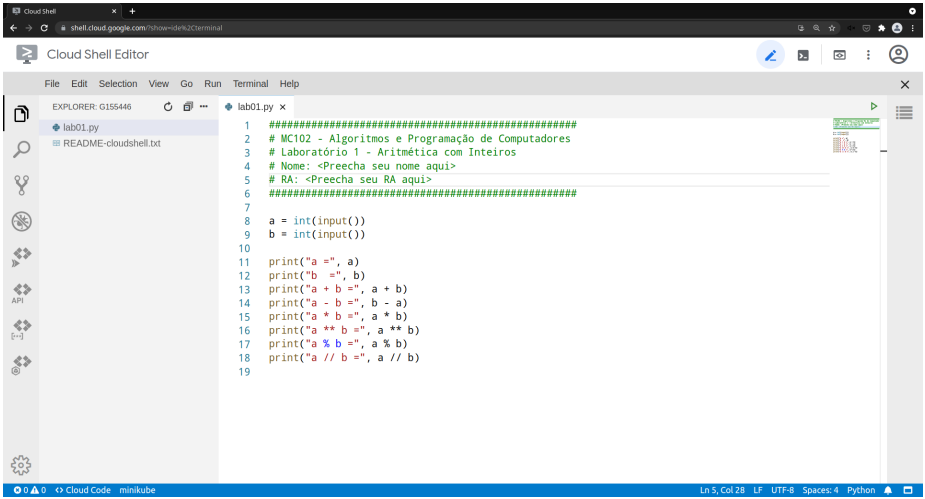

Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: NO FOLDER OPEN...
Welcome to Cloud Shell
You have not yet opened a folder.
Open Folder
Cloud Shell Editor
Create, build and deploy your cloud-native applications in an online editor. To get started, open your home folder or one of the options below to load your workspace.
Start
Open Folder...
Open Home Workspace
What's new in Version 1.14.0 (July 2021)
HashiCorp Terraform v2.12.0 – Cloud Shell Editor now includes Terraform language support! Full Terraform release notes here.
Cloud Code extension v1.12.0 – Introducing Cloud Build support for Cloud Run and Kubernetes. Full Cloud Code release notes here.
Golang extension v0.23.0 – Improved debbing workflows, access to Delve DAP. Full Golang release notes here.
All Cloud Shell release notes
Recent
g155446 /home/g155446
Learn
Editor Overview
Learn about the Cloud Shell Editor - built using Eclipse Theia
Interface overview
Overview of our user interface and basic layout
0 0 Cloud Code minikube


Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: NO FOLDER OPEN...
Welcome to Cloud Shell
You have not yet opened a folder.
Open Folder
Cloud Shell Editor
Create, build and get started, open workspace.
Start
Open Folder...
Open Home Work
Recent
g155446 /home/g
Open Folder
g155446
g155446
README-cloudshell.txt
Display Hidden Files
Cancel
Open
on 1.14.0 (July 2021)
n v2.12.0 – Cloud Shell Editor now includes support! Full Terraform release notes here.
on v1.12.0 – Introducing Cloud Build support for notes. Full Cloud Code release notes here.
0.23.0 – Improved debbing workflows, access to ng release notes here.
e notes
Learn about the Cloud Shell Editor - built using Eclipse Theia
Interface overview
Overview of our user interface and basic layout
0 0 Cloud Code minikube

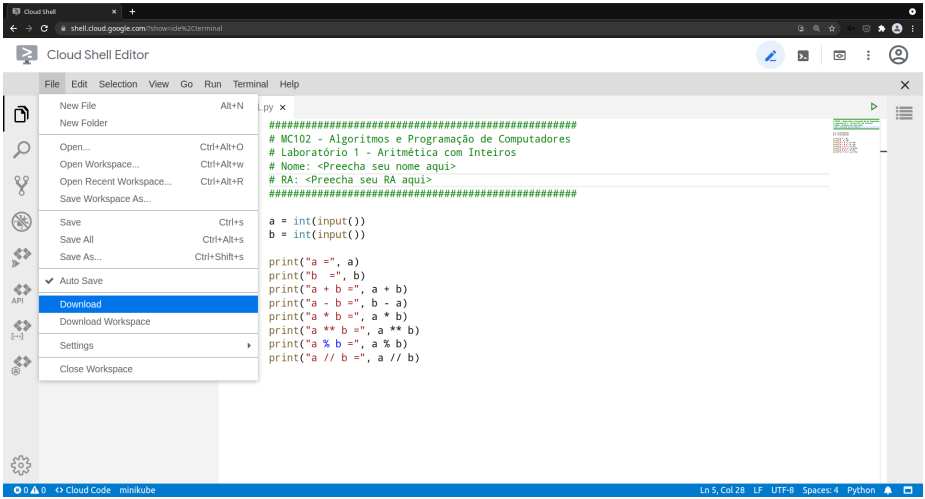

Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: G155446
README-cloudshell.txt
API
0 0 Activating Cloud Code

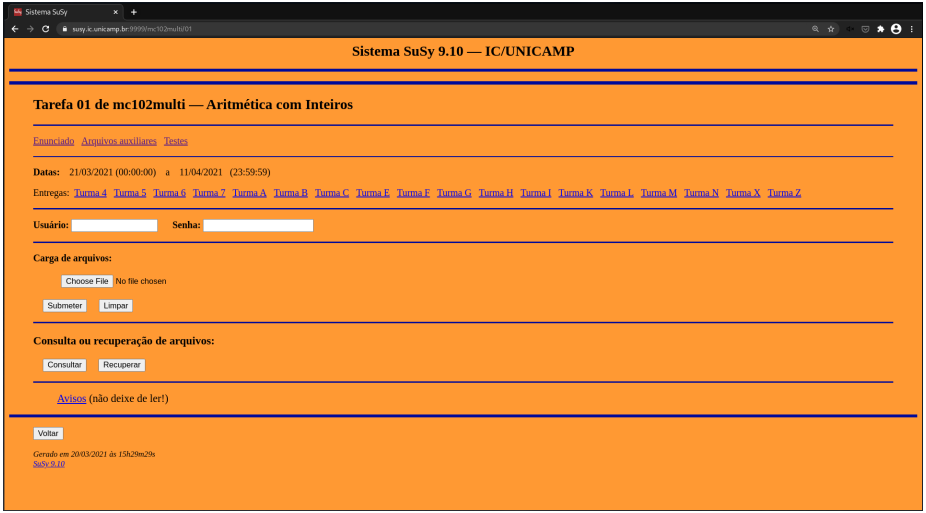

Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
New File Alt+N
New Folder
Open... Ctrl+Alt+O
Open Workspace... Ctrl+Alt+W
Open Recent Workspace... Ctrl+Alt+R
Save Workspace As...
Save Ctrl+S
Save All Ctrl+Alt+S
Save As... Ctrl+Shift+S
Auto Save
Upload Files...
Download
Download Workspace
Settings
Close Workspace
API
0 0 Cloud Code minikube


Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: G155446
lab01.py
README-cloudshell.txt
lab01.py
1 ##
2 # MC102 - Algoritmos e Programação de Computadores
3 # Laboratório 1 - Aritmética com Inteiros
4 # Nome:
5 # RA:
6 ##
7
8 a = int(input())
9 b = int(input())
10
11 print("a =", a)
12 print("b =", b)
13 print("a + b =", a + b)
14 print("a - b =", b - a)
15 print("a * b =", a * b)
16 print("a ** b =", a ** b)
17 print("a % b =", a % b)
18 print("a // b =", a // b)
19
0 0 Cloud Code Python extension loading... minikube
Ln 1, Col 1 LF UTF-8 Spaces: 4 Python


Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: G155446
lab01.py
README-cloudshell.txt
lab01.py
1 ##
2 # MC102 - Algoritmos e Programação de Computadores
3 # Laboratório 1 - Aritmética com Inteiros
4 # Nome: <Preecha seu nome aqui>
5 # RA: <Preecha seu RA aqui>
6 ##
7
8 a = int(input())
9 b = int(input())
10
11 print("a =", a)
12 print("b =", b)
13 print("a + b =", a + b)
14 print("a - b =", b - a)
15 print("a * b =", a * b)
16 print("a ** b =", a ** b)
17 print("a % b =", a % b)
18 print("a // b =", a // b)
19
API
0 0 Cloud Code minikube
Ln 5, Col 28 LF UTF-8 Spaces: 4 Python

1. Baixe o arquivo lab01.py do Google Cloud Shell para submeter no SuSy.
2. Na página da atividade prática no SuSy, para os campos de "Usuário" e "Senha", informe seu RA (apenas os números) e sua senha da DAC, respectivamente.
3. Na seção "Carga de arquivos:" clique em "Choose File" e selecione o arquivo do código base que você acabou de realizar o download (lab01.py).
4. Em seguida, clique no botão "Submeter".


Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
New File Alt+N
New Folder
Open... Ctrl+Alt+O
Open Workspace... Ctrl+Alt+w
Open Recent Workspace... Ctrl+Alt+R
Save Workspace As...
Save Ctrl+s
Save All Ctrl+Alt+s
Save As... Ctrl+Shift+s
Auto Save
Download
Download Workspace
Settings
Close Workspace
API
# MC102 - Algoritmos e Programação de Computadores
# Laboratório 1 - Aritmética com Inteiros
# Nome: <Preecha seu nome aqui>
# RA: <Preecha seu RA aqui>
a = int(input())
b = int(input())
print("a =", a)
print("b =", b)
print("a + b =", a + b)
print("a - b =", b - a)
print("a * b =", a * b)
print("a ** b =", a ** b)
print("a % b =", a % b)
print("a // b =", a // b)
Cloud Code minikube
Ln 5, Col 28 LF UTF-8 Spaces: 4 Python


Sistema SuSy
susy.ic.unicamp.br:9999/mc102multi/01
Sistema SuSy 9.10 — IC/UNICAMP
Tarefa 01 de mc102multi — Aritmética com Inteiros
Enunciado Arquivos auxiliares Testes
Datas: 21/03/2021 (00:00:00) a 11/04/2021 (23:59:59)
Entregas: Turma 4 Turma 5 Turma 6 Turma 7 Turma A Turma B Turma C Turma E Turma F Turma G Turma H Turma I Turma K Turma L Turma M Turma N Turma X Turma Z
Usuário:
Senha:
Carga de arquivos:
Choose File
No file chosen
Submeter
Limpar
Consulta ou recuperação de arquivos:
Consultar
Recuperar
Avisos (não deixe de ler!)
Voltar
Gerado em 20/03/2021 às 15h29m29s
SuSy 9.10

Arquivos carregados:

```
lab01.py (lab01.py):  434 bytes

Total: 434 bytes
```

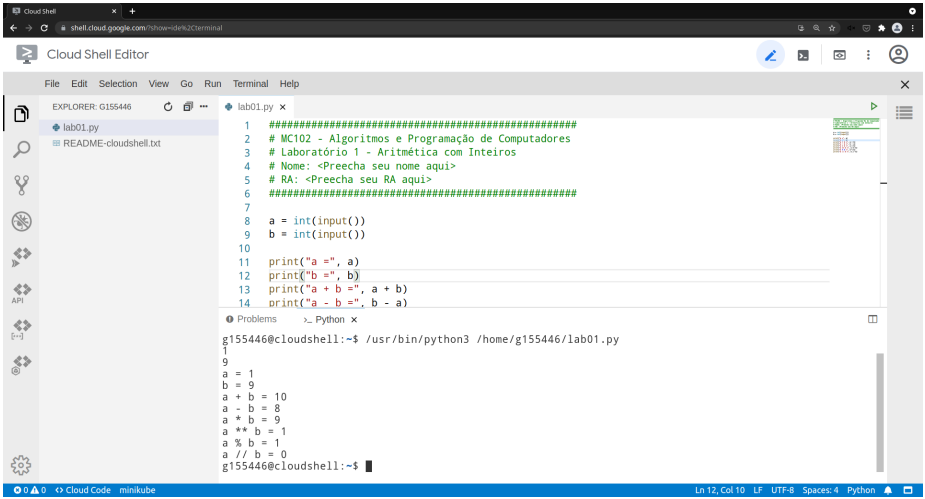
Fase de execução:

```
Teste 01: resultado incorreto

     b  = 9                                                    | b = 9
     a - b = 8                                                 | a - b = -8
     a % b = 1                                                 <
                                                               > a % b = 1


Teste 02: resultado incorreto

     b  = 8                                                    | b = 8
     a - b = 6                                                 | a - b = -6
     a % b = 2                                                 <
                                                               > a % b = 2


Teste 03: resultado incorreto

     b  = 3                                                    | b = 3
     a - b = -4                                                | a - b = 4
     a % b = 1                                                 <
                                                               > a % b = 1
```

1. Acesse a página da atividade desejada no SuSy.
2. Informe seu usuário e sua senha.
3. Clique em “Consultar”.
4. Será mostrado o relatório da sua última submissão.

1. Acesse a página da atividade desejada no SuSy.
2. Informe seu usuário e sua senha.
3. Clique em “Recuperar”.
4. Será mostrado um link para o seu último arquivo submetido.

1. No Google Cloud Shell, faça a primeira correção solicitada no enunciado (linha 12 do código).
2. Clique na seta verde no canto superior esquerdo para executar o programa.
3. Em seguida, digite no terminal (campo abaixo do código) as entradas para o seu programa conforme mostrado no enunciado da atividade.
4. Baixe o arquivo lab01.py e submeta no SuSy.
5. Sua submissão ainda deve gerar um relátorio com "resultado incorreto" para todos os casos de teste.


Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: G155446
lab01.py
README-cloudshell.txt
lab01.py
1 ##
2 # MC102 - Algoritmos e Programação de Computadores
3 # Laboratório 1 - Aritmética com Inteiros
4 # Nome: <Preecha seu nome aqui>
5 # RA: <Preecha seu RA aqui>
6 ##
7
8 a = int(input())
9 b = int(input())
10
11 print("a =", a)
12 print("b =", b)
13 print("a + b =", a + b)
14 print("a - b =", b - a)
Problems
Python
g155446@cloudshell:~$ /usr/bin/python3 /home/g155446/lab01.py
1
9
a = 1
b = 9
a + b = 10
a - b = 8
a * b = 9
a ** b = 1
a % b = 1
a // b = 0
g155446@cloudshell:~$
Cloud Code
minikube
Ln 12, Col 10
LF
UTF-8
Spaces: 4
Python

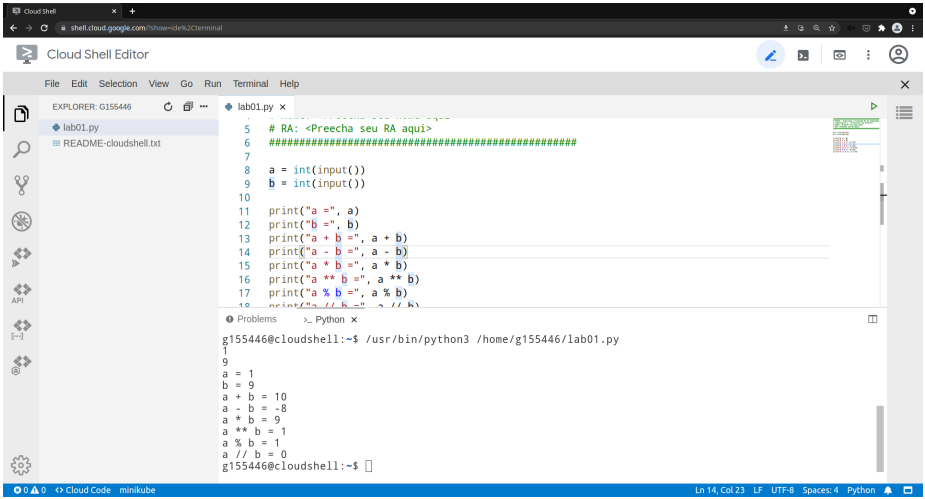
Arquivos carregados:

```
lab01.py (lab01.py):  477 bytes

Total: 477 bytes
```

Fase de execução:

```
Teste 01: resultado incorreto

     a - b = 8                                          | a - b = -8
     a % b = 1                                          <
                                                        > a % b = 1

Teste 02: resultado incorreto

     a - b = 6                                          | a - b = -6
     a % b = 2                                          <
                                                        > a % b = 2

Teste 03: resultado incorreto

     a - b = -4                                         | a - b = 4
     a % b = 1                                          <
                                                        > a % b = 1

Teste 04: resultado incorreto

     a - b = -5                                         | a - b = 5
```

1. No Google Cloud Shell, faça a segunda correção solicitada no enunciado (linha 14 do código).
2. Em seguida, teste novamente seu programa com as entradas fornecidas na atividade.
3. Baixe o arquivo lab01.py e submeta no SuSy.
4. Sua submissão ainda deve gerar um relatório com "resultado incorreto" para todos os casos de teste.


Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: G155446
lab01.py
README-cloudshell.txt
lab01.py
RA: <Preecha seu RA aqui>
##
a = int(input())
b = int(input())
print("a =", a)
print("b =", b)
print("a + b =", a + b)
print("a - b =", a - b)
print("a * b =", a * b)
print("a ** b =", a ** b)
print("a % b =", a % b)
Problems
Python
g155446@cloudshell:~$ /usr/bin/python3 /home/g155446/lab01.py
1
9
a = 1
b = 9
a + b = 10
a - b = -8
a * b = 9
a ** b = 1
a % b = 1
a // b = 0
g155446@cloudshell:~$
Cloud Code
minikube
Ln 14, Col 23
LF
UTF-8
Spaces: 4
Python

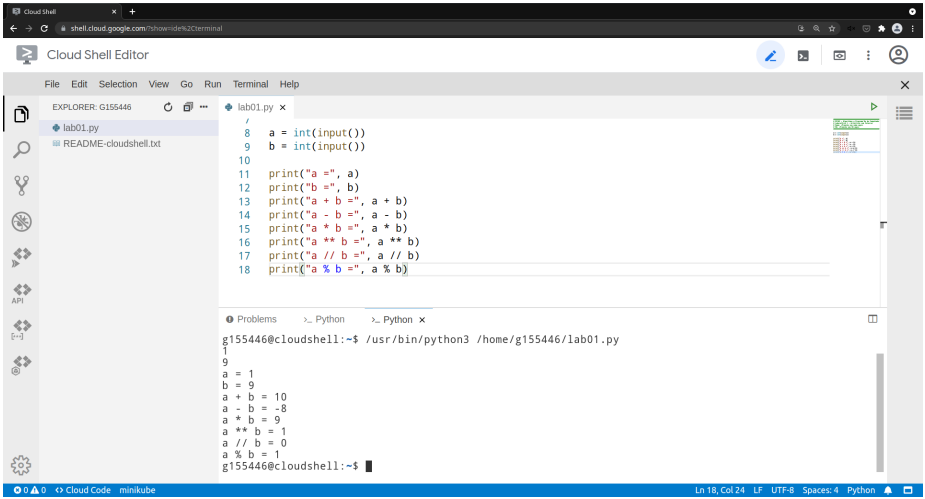
Arquivos carregados:

```
lab01.py (lab01.py):  477 bytes

Total: 477 bytes
```

Fase de execução:

```
Teste 01: resultado incorreto

      a % b = 1                                                 <
                                                                > a % b = 1

Teste 02: resultado incorreto

      a % b = 2                                                 <
                                                                > a % b = 2

Teste 03: resultado incorreto

      a % b = 1                                                 <
                                                                > a % b = 1

Teste 04: resultado incorreto

      a % b = 5                                                 <
                                                                > a % b = 5

Teste 05: resultado incorreto
```

1. Voltando ao Google Cloud Shell, faça a terceira correção solicitada no enunciado (linhas 17 e 18 do código).
2. Teste novamente seu programa com as entradas fornecidas na atividade.
3. Baixe o arquivo lab01.py e submeta no SuSy.
4. Nessa submissão o relatório gerado deve indicar "resultado correto" para todos os casos de teste.


Cloud Shell
shell.cloud.google.com/?show=ide%2Cterminal
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: G155446
lab01.py
README-cloudshell.txt
lab01.py
8 a = int(input())
9 b = int(input())
10
11 print("a =", a)
12 print("b =", b)
13 print("a + b =", a + b)
14 print("a - b =", a - b)
15 print("a * b =", a * b)
16 print("a ** b =", a ** b)
17 print("a // b =", a // b)
18 print("a % b =", a % b)
Problems
Python
Python
g155446@cloudshell:~$ /usr/bin/python3 /home/g155446/lab01.py
1
9
a = 1
b = 9
a + b = 10
a - b = -8
a * b = 9
a ** b = 1
a // b = 0
a % b = 1
g155446@cloudshell:~$
0 0 Cloud Code minikube
Ln 18, Col 24 LF UTF-8 Spaces: 4 Python

Arquivos carregados:

```
lab01.py (lab01.py):  476 bytes

Total: 476 bytes
```

Fase de execução:

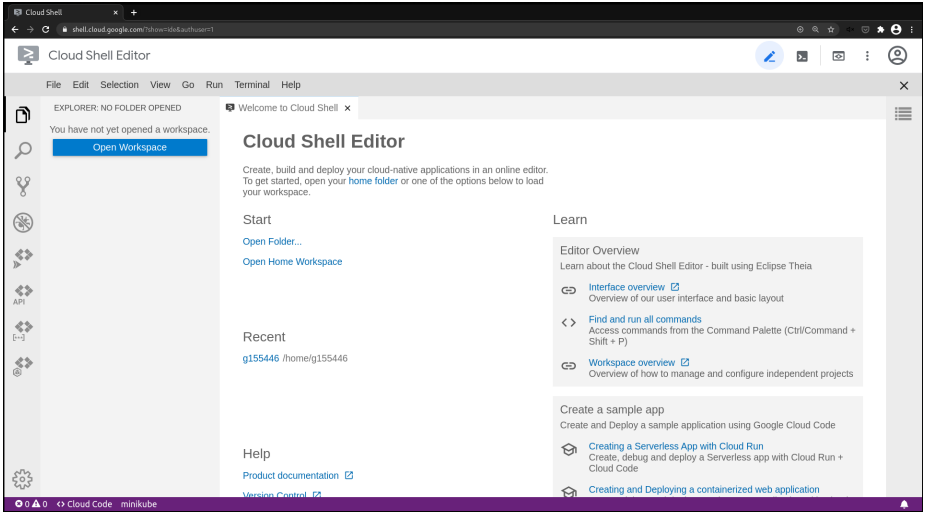
```
Teste 01: resultado correto
Teste 02: resultado correto
Teste 03: resultado correto
Teste 04: resultado correto
Teste 05: resultado correto
Teste 06: resultado correto
Teste 07: resultado correto
Teste 08: resultado correto
Teste 09: resultado correto
Teste 10: resultado correto
Teste 11: resultado correto
Teste 12: resultado correto
Teste 13: resultado correto
Teste 14: resultado correto
Teste 15: resultado correto
Teste 16: resultado correto
Teste 17: resultado correto
Teste 18: resultado correto
Teste 19: resultado correto
Teste 20: resultado correto
```

- São permitidas no máximo 20 submissões no SuSy para cada atividade prática.
- Utilize o sistema SuSy com o seu RA (apenas números) e com a senha que você utiliza para fazer acesso ao sistema da DAC.
- Para avaliação, será considerado apenas o resultado da última submissão.
- Você deve seguir com cuidado as instruções de submissão descritas no enunciado.
- Não use o SuSy para testar o seu programa: sempre teste seu programa com os casos de testes abertos, antes de submeter o seu programa para avaliação no SuSy.
- Para mais informações, visite o site da disciplina: `https://ic.unicamp.br/~mc102`

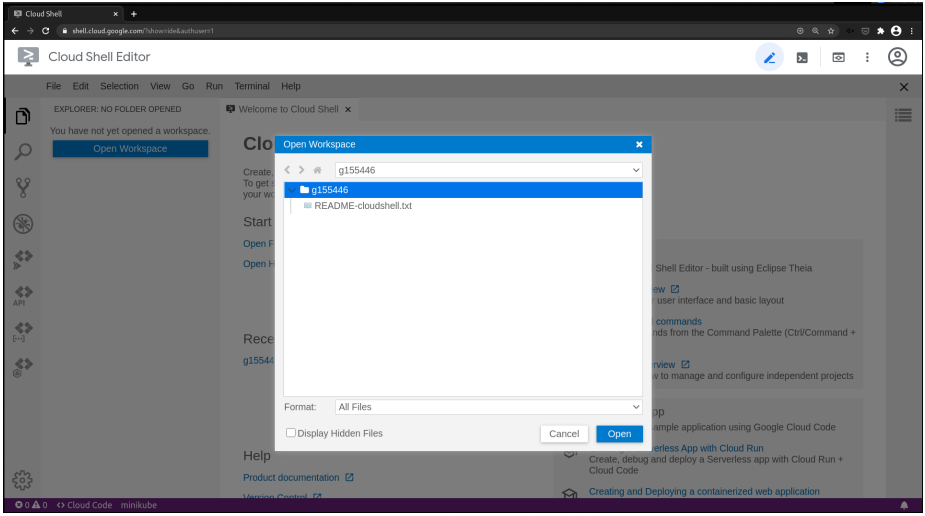
Google Cloud Shell

- Google Cloud Shell:
  - `https://shell.cloud.google.com/`

- Para acessar a ferramenta será necessária uma conta no Google.
- Todos os alunos têm acesso a uma conta do Google utilizando a senha da DAC:
  - https://www.ccuec.unicamp.br/ccuec/euquero/utilizar-e-mail-e-ferramentas-da-google
- Acesse a plataforma Google Cloud Shell com uma conta do Google:
  - https://shell.cloud.google.com
- Selecione a opção *Open Workspace* que aparece à direita.
- Uma janela mostrando uma pasta com o seu nome de usuário será exibida. Basta clicar em *Open* para abri-la como um *workspace* (área de trabalho).

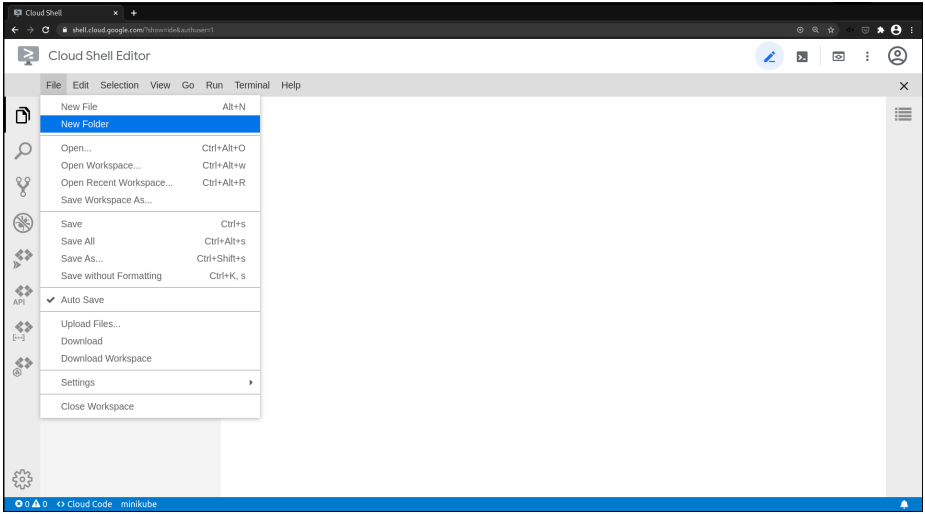


Tela inicial do Google Cloud Shell.

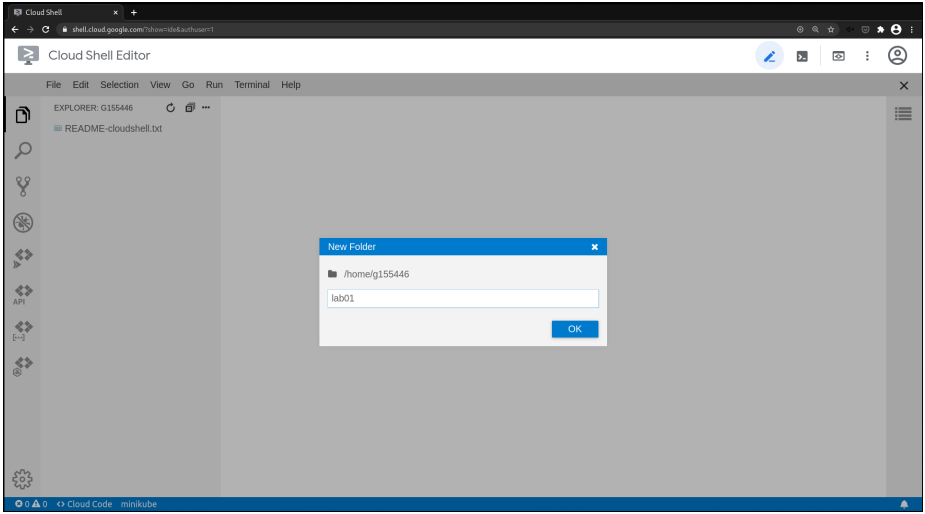


Janela para abertura de um novo *workspace*.

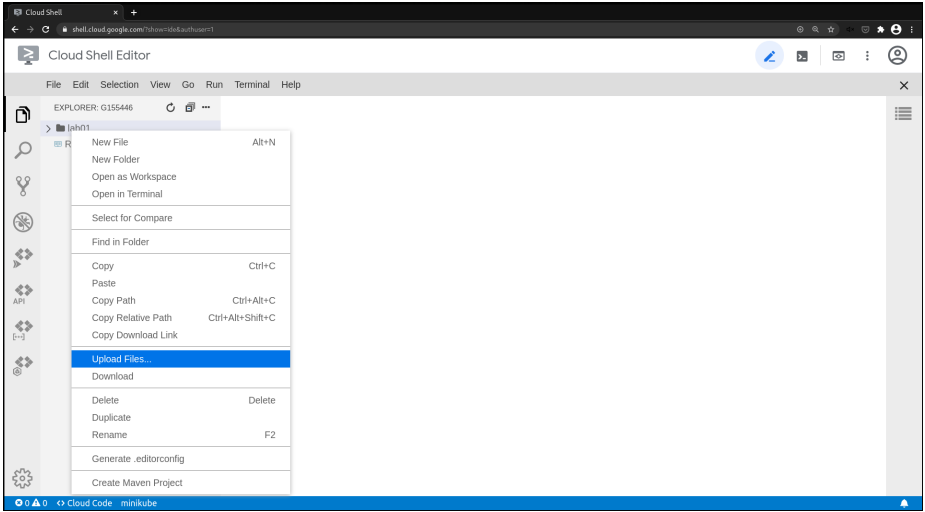
- Faça o download dos testes do SuSy e do código base: arquivos `auxXX.zip` e `labXX.py`, onde XX é o número do laboratório.
- Descompacte os arquivos de testes.
- Copie os arquivos testador.py e `labXX.py` para o mesmo diretório que você descompactou os arquivos de testes.
- No Google Cloud Shell, selecione a opção *New Folder* no menu *File* e crie uma pasta como o nome `labXX`.
- Clicando como o botão direito na pasta criada, selecione a opção *Upload Files* e envie todos os arquivos de teste, o código base e o arquivo testador.py.



Criando uma nova pasta.

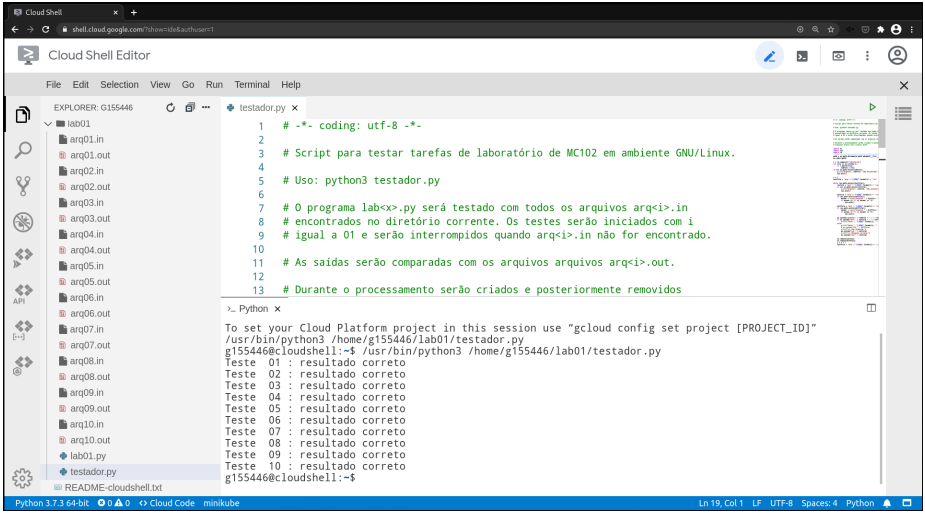


Criando uma nova pasta.



Selecione a opção *Upload File* e envie todos os arquivos de teste, o código base e o arquivo testador.py.

- Complete o arquivo `labXX.py` com a sua solução para o laboratório.
- Para testar manualmente o programa basta clicar na seta verde no canto superior esquerdo com o arquivo `labXX.py` aberto.
- Para testar automaticamente o programa, com todos os casos de teste, basta clicar na seta verde no canto superior esquerdo com o arquivo `testador.py` aberto.
- Caso a seta verde não apareça é necessario editar o arquivo aberto (adicionar e remover um espaço é o suficiente).
- Para enviar o código para o SuSy, basta baixar o arquivo `labXX.py` e submeter.


Cloud Shell
shell.cloud.google.com/?show=ide&authuser=1
Cloud Shell Editor
File Edit Selection View Go Run Terminal Help
EXPLORER: G155446
lab01
arq01.in
arq01.out
arq02.in
arq02.out
arq03.in
arq03.out
arq04.in
arq04.out
arq05.in
arq05.out
arq06.in
arq06.out
arq07.in
arq07.out
arq08.in
arq08.out
arq09.in
arq09.out
arq10.in
arq10.out
lab01.py
testador.py
README-cloudshell.txt
testador.py
1 # -*- coding: utf-8 -*-
2
3 # Script para testar tarefas de laboratório de MC102 em ambiente GNU/Linux.
4
5 # Uso: python3 testador.py
6
7 # O programa lab<x>.py será testado com todos os arquivos arq<i>.in
8 # encontrados no diretório corrente. Os testes serão iniciados com i
9 # igual a 01 e serão interrompidos quando arq<i>.in não for encontrado.
10
11 # As saídas serão comparadas com os arquivos arquivos arq<i>.out.
12
13 # Durante o processamento serão criados e posteriormente removidos
Python
To set your Cloud Platform project in this session use "gcloud config set project [PROJECT_ID]"
/usr/bin/python3 /home/g155446/lab01/testador.py
g155446@cloudshell:~$ /usr/bin/python3 /home/g155446/lab01/testador.py
Teste 01 : resultado correto
Teste 02 : resultado correto
Teste 03 : resultado correto
Teste 04 : resultado correto
Teste 05 : resultado correto
Teste 06 : resultado correto
Teste 07 : resultado correto
Teste 08 : resultado correto
Teste 09 : resultado correto
Teste 10 : resultado correto
g155446@cloudshell:~$
Python 3.7.3 64-bit
Cloud Code
minikube
Ln 19, Col 1
LF
UTF-8
Spaces: 4
Python

Perguntas ....

Referências

- Zanoni Dias, MC102, Algoritmos e Programação de Computadores, IC/UNICAMP, 2021. https://ic.unicamp.br/~mc102/

  Aula Introdutória [slides] [vídeo]
  Primeira Aula de Laboratório [slides] [vídeo]
  Python Básico: Tipos, Variáveis, Operadores, Entrada e Saída [slides] [vídeo]
  Comandos Condicionais [slides] [vídeo]
  Comandos de Repetição [slides] [vídeo]
  Listas e Tuplas [slides] [vídeo]
  Strings [slides] [vídeo]
  Dicionários [slides] [vídeo]
  Funções [slides] [vídeo]
  Objetos Multidimensionais [slides] [vídeo]
  Algoritmos de Ordenação [slides] [vídeo]
  Algoritmos de Busca [slides] [vídeo]
  Recursão [slides] [vídeo]
  Algoritmos de Ordenação Recursivos [slides] [vídeo]
  Arquivos [slides] [vídeo]
  Expressões Regulares [slides] [vídeo]
  Execução de Testes no Google Cloud Shell [slides] [vídeo]
  Numpy [slides] [vídeo]
  Pandas [slides] [vídeo]

- Panda - Cursos de Computação em Python (IME -USP) https://panda.ime.usp.br/

  Como Pensar Como um Cientista da Computação https://panda.ime.usp.br/pensepy/static/pensepy/
  Aulas de Introdução à Computação em Python https://panda.ime.usp.br/aulasPython/static/aulasPython/

- Fabio Kon, Introdução à Ciência da Computação com Python http://bit.ly/FabioKon/
- Socratica, Python Programming Tutorials http://bit.ly/SocraticaPython/
- Google - online editor for cloud-native applications (Python programming) https://shell.cloud.google.com/
- w3schools - Python Tutorial https://www.w3schools.com/python/
- Outros, citados nos Slides.