



Universidade Federal de Uberlândia  
FEELT – Faculdade de Engenharia Elétrica  
FEELT39015 - Sistemas Embarcados II



## **Semana 02: Linux - Ambiente de Programação**

Professor: Éder Alves de Moura

Luiza Custódio Freitas (12021EAU002)



## 1. Introdução

Este relatório contém uma revisão das ferramentas de compilação em linguagem C no ambiente Linux. O Kernel do Linux é desenvolvido em linguagem C e esta é a base para a criação das chamadas de sistema para o Kernel. Para isso, tem-se o GNU cc (gcc) e algumas de suas propriedades. Por fim, apresenta-se um nivelamento em programação na linguagem Python.

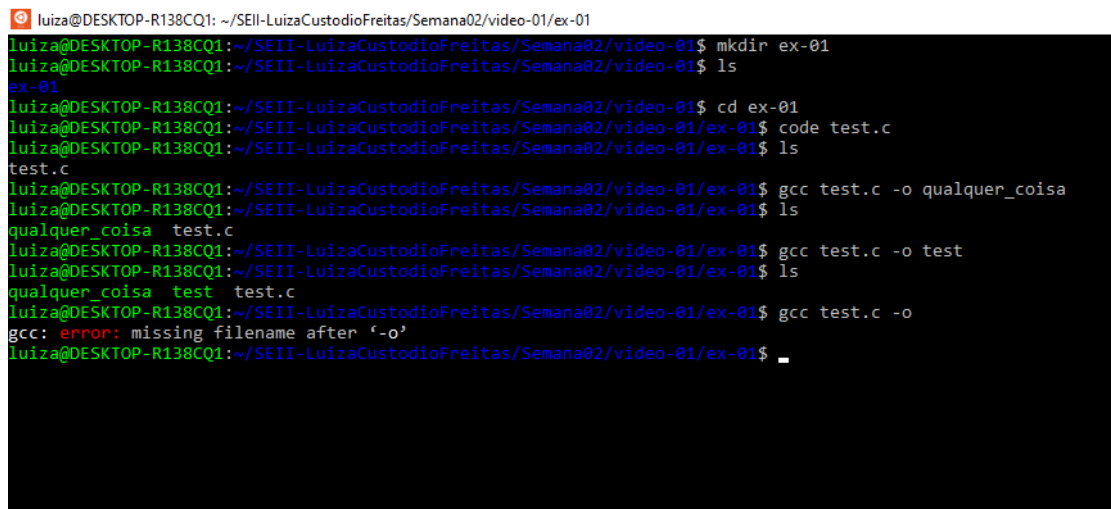
## 2. Compilação de códigos em C

Os vídeos sugeridos no roteiro e referenciados trazem o conteúdo de compilação de códigos na linguagem C utilizando o Linux. Os dois primeiros se tratam da compilação manual utilizando os comandos com gcc, já os três últimos fornecem uma alternativa de compilação automática com o Makefile.

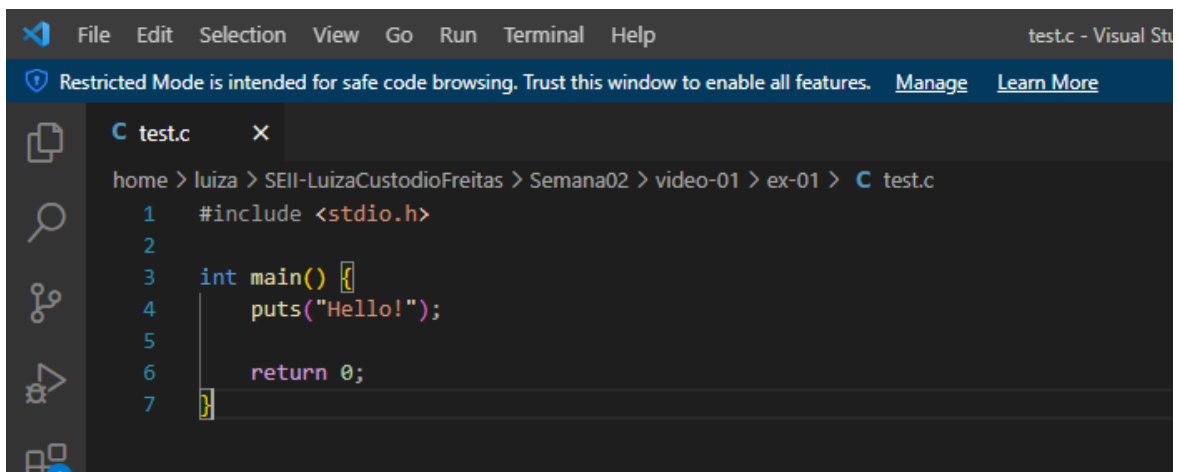
É importante lembrar que o código objeto é um binário gerado pelo compilador depois que já processou o código fonte. Ele costuma ficar em um arquivo para depois poder gerar o executável através do processo de linkedição.

### Vídeo 1:

Exemplo 1: este é um exemplo introdutório, nele cria-se uma pasta “ex-01”, onde é dado um código em “.c” de exemplo utilizando o Visual Studio Code (comando “code test.c”) e depois compilado utilizando o comando “gcc test.c -o” para criar um arquivo .o (código objeto). Um detalhe apresentado no vídeo é o arquivo .o pode ser nomeado em seguida, se não for nomeado será criado um arquivo com o mesmo nome do arquivo .c.

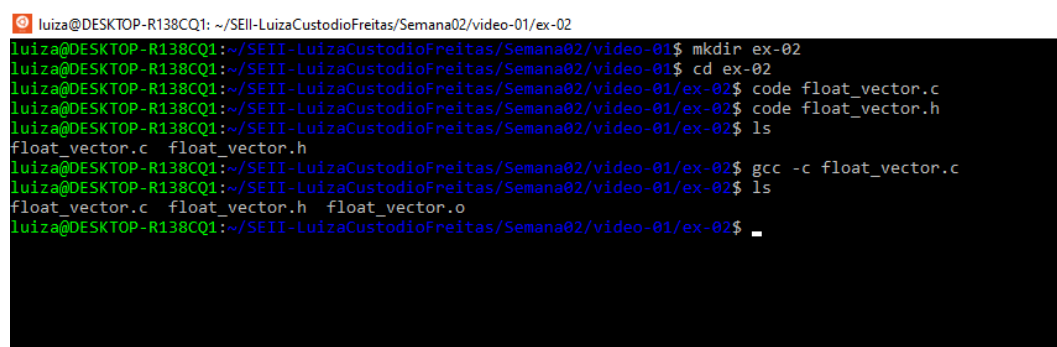


```
luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01$ mkdir ex-01
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01$ ls
ex-01
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01$ cd ex-01
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ code test.c
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ ls
test.c
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ gcc test.c -o qualquer_coisa
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ ls
qualquer_coisa  test.c
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ gcc test.c -o test
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ ls
qualquer_coisa  test  test.c
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ gcc test.c -o
gcc: error: missing filename after '-o'
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-01$ _
```

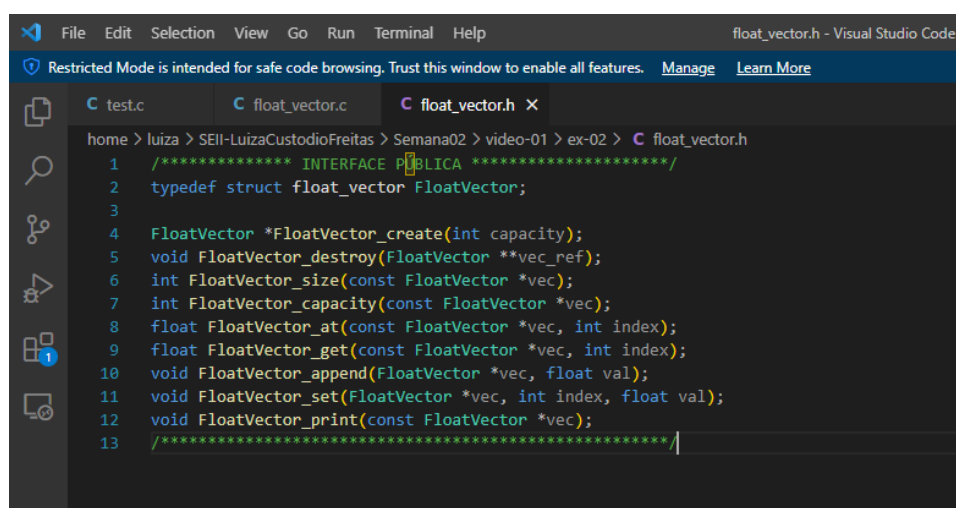


```
File Edit Selection View Go Run Terminal Help
test.c - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
C test.c
home > luiza > SEII-LuizaCustodioFreitas > Semana02 > video-01 > ex-01 > C test.c
1 #include <stdio.h>
2
3 int main() {
4     puts("Hello!");
5
6     return 0;
7 }
```

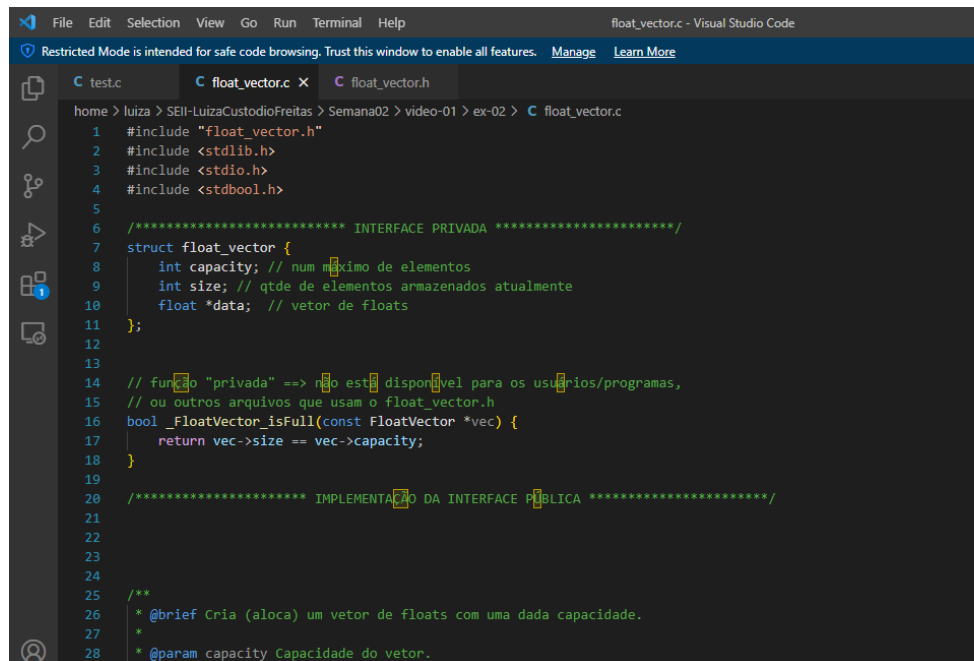
Exemplo 2: criada a pasta “ex-02”, é introduzido dois códigos no Visual Studio, sendo estes “float\_vector.c” e “float\_vector.h”, respectivamente o código principal e código de inclusões. Utilizando o comando manual “gcc -c float\_vector.c” no linux novamente, é compilado o arquivo “float\_vector.c” e tem-se o código objeto “float\_vector.o”



```
luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01$ mkdir ex-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01$ cd ex-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ code float_vector.c
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ code float_vector.h
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ ls
float_vector.c float_vector.h
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ gcc -c float_vector.c
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ ls
float_vector.c float_vector.h float_vector.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$
```

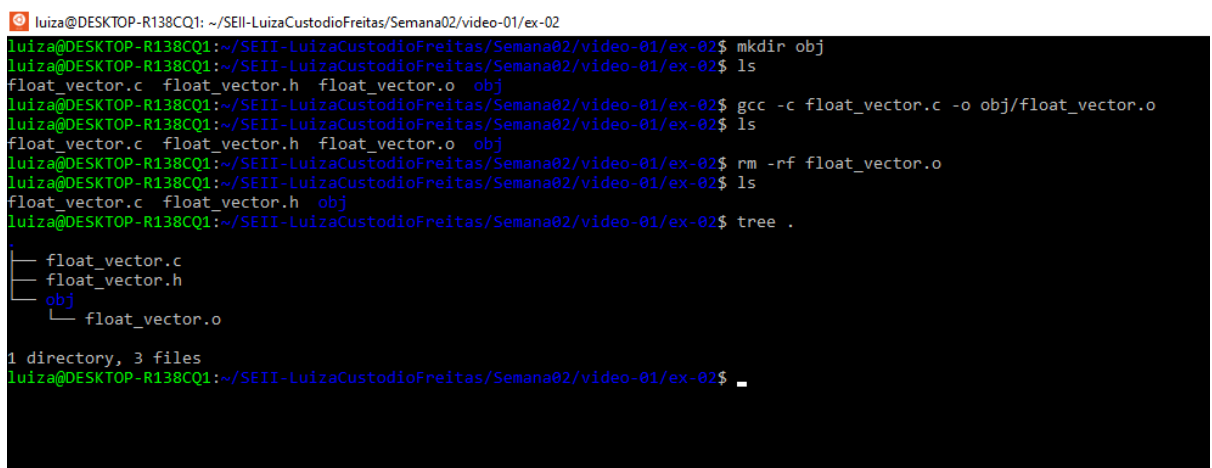


```
File Edit Selection View Go Run Terminal Help
float_vector.h - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
C test.c C float_vector.c C float_vector.h
home > luiza > SEII-LuizaCustodioFreitas > Semana02 > video-01 > ex-02 > C float_vector.h
1 /****** INTERFACE PÚBLICA *****/
2 typedef struct float_vector FloatVector;
3
4 FloatVector *FloatVector_create(int capacity);
5 void FloatVector_destroy(FloatVector **vec_ref);
6 int FloatVector_size(const FloatVector *vec);
7 int FloatVector_capacity(const FloatVector *vec);
8 float FloatVector_at(const FloatVector *vec, int index);
9 float FloatVector_get(const FloatVector *vec, int index);
10 void FloatVector_append(FloatVector *vec, float val);
11 void FloatVector_set(FloatVector *vec, int index, float val);
12 void FloatVector_print(const FloatVector *vec);
13 /*****/
```



```
1 #include "float_vector.h"
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <stdbool.h>
5
6 /***** INTERFACE PRIVADA *****/
7 struct float_vector {
8     int capacity; // num máximo de elementos
9     int size; // qtde de elementos armazenados atualmente
10    float *data; // vetor de floats
11 };
12
13
14 // função "privada" ==> não está disponível para os usuários/programas,
15 // ou outros arquivos que usam o float_vector.h
16 bool _FloatVector_isFull(const FloatVector *vec) {
17     return vec->size == vec->capacity;
18 }
19
20 /***** IMPLEMENTAÇÃO DA INTERFACE PÚBLICA *****/
21
22
23
24
25 /**
26  * @brief Cria (aloca) um vetor de floats com uma dada capacidade.
27  *
28  * @param capacity Capacidade do vetor.
```

Exemplo 3: agora cria-se o diretório “obj” para criar e depositar os arquivos objetos por meio do comando manual “gcc -c float\_vector.c -o obj/float\_vector.o”. Logo em seguida, é possível utilizar o comando “tree .” que visualiza a disposição de diretórios e seus arquivos em um formato mais visual e simples.



```
luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ mkdir obj
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ ls
float_vector.c float_vector.h float_vector.o obj
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ gcc -c float_vector.c -o obj/float_vector.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ ls
float_vector.c float_vector.h float_vector.o obj
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ rm -rf float_vector.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ ls
float_vector.c float_vector.h obj
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ tree .
.
├── float_vector.c
├── float_vector.h
└── obj
    └── float_vector.o

1 directory, 3 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$
```

Exemplo 4: com o mesmo intuito do último exemplo, cria-se agora uma pasta “include” para os códigos de inclusão e deposita-se o “float\_vector.h” com o comando de mover “mv float\_vector.h include/”. Agora, de uma maneira mais organizada, pode-se criar o código objeto e colocar os arquivos em suas respectivas pastas utilizando apenas o comando “gcc -c float\_vector.c -I include/ -o obj/float\_vector.o”.

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ mkdir include
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ ls
float_vector.c  float_vector.h  include  obj
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ mv float_vector.h include/
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ ls
float_vector.c  include  obj
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ tree .
.
├── float_vector.c
├── include
│   └── float_vector.h
└── obj
    └── float_vector.o

2 directories, 3 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ gcc -c float_vector.c -I include/ -o obj/float_vector.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ tree .
.
├── float_vector.c
├── include
│   └── float_vector.h
└── obj
    └── float_vector.o

2 directories, 3 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$

```

## Vídeo 2:

Exemplo 5: após criar um diretório tanto para o vídeo 2 quanto para o exemplo 5 e copiado os arquivos do último exemplo utilizando o comando “cp -R”, cria-se mais um diretório “src” de Source - Fonte, que correspondem aos códigos .c, move-se o arquivo “float\_vector.c” com “mv float\_vector.c src/” e agora o comando para compilação e organização dos diretórios fica “gcc -c ./src/float\_vector.c -I include/ -o ./obj/float\_vector.o”.

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01/ex-02$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-01$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02$ mkdir video-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02$ cd video-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ cp -R ../video-01/ex-02
cp: missing destination file operand after '../video-01/ex-02'
Try 'cp --help' for more information.
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ cp -R ../video-01/ex-02 ex-05
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ ls
ex-05
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ cd ex-05
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ ls
float_vector.c  include  obj
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ tree .
.
├── float_vector.c
├── include
│   └── float_vector.h
└── obj
    └── float_vector.o

2 directories, 3 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ rm -rf obj/float_vector.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ tree .
.
├── float_vector.c
├── include
│   └── float_vector.h
└── obj

2 directories, 2 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$

```

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ mkdir src
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ mv float_vector.c src/
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ tree .
.
├── include
│   └── float_vector.h
├── obj
├── src
│   └── float_vector.c
3 directories, 2 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ gcc -c ./src/float_vector.c -I ./include -o ./obj/float_vector.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$ tree .
.
├── include
│   └── float_vector.h
├── obj
│   └── float_vector.o
├── src
│   └── float_vector.c
3 directories, 3 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-05$

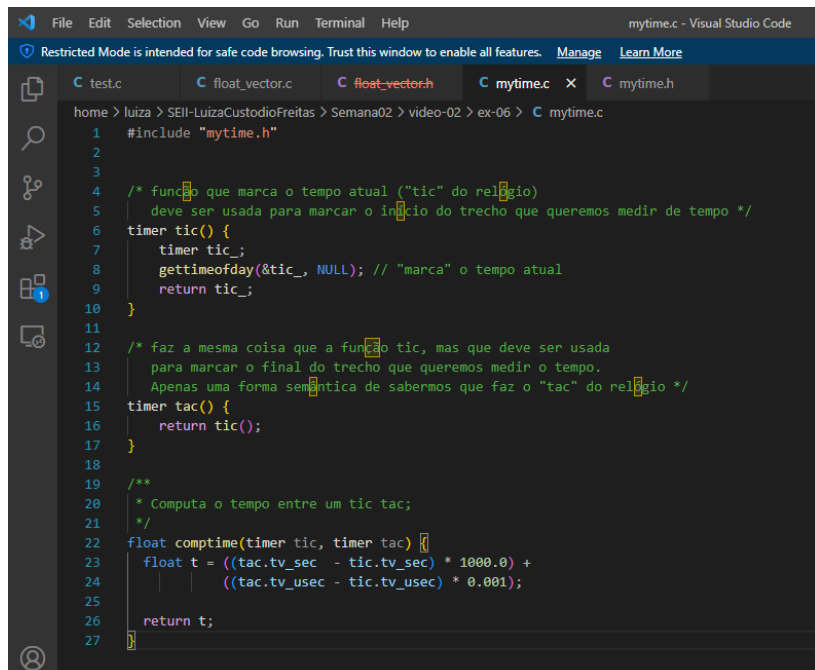
```

Exemplo 6: nesse exemplo são introduzidos novos códigos fontes, inclusões e sempre implementados em suas respectivas pastas, para isso cada vez mais o comando de compilação aumenta para considerar tudo que está sendo pedido e todos os códigos utilizados. Além disso, são criadas mais dois diretórios, para colocar as aplicações e os binários (código executável).

```

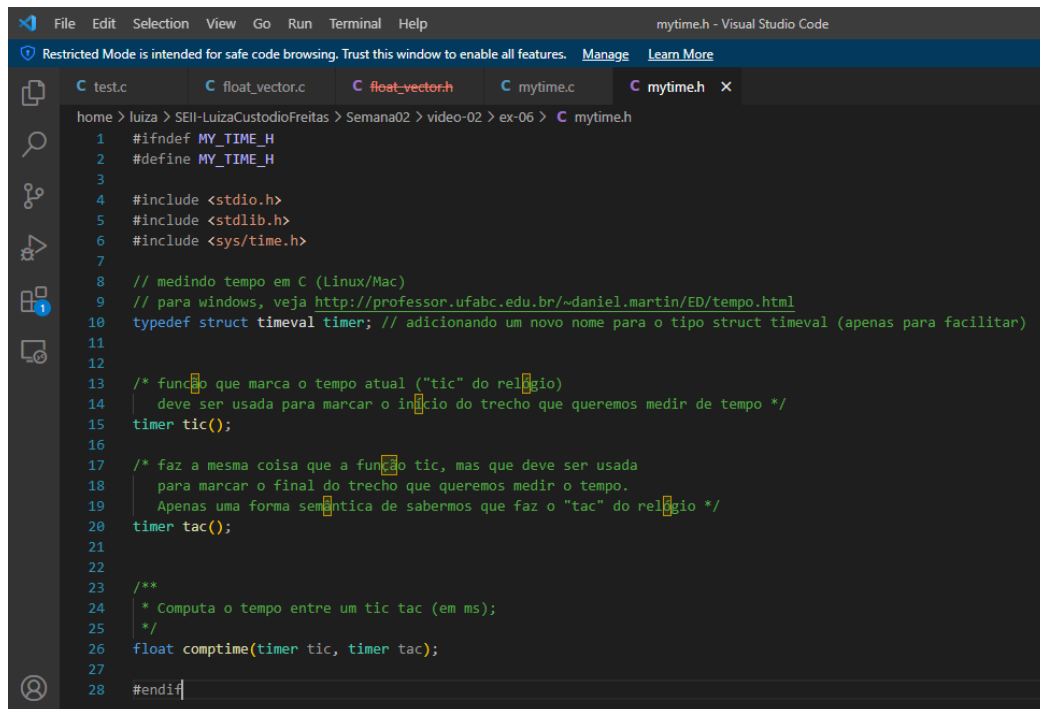
luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ cp -R ex-05 ex-06
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ ls
ex-05  ex-06
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ cd ex-06
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ ls
include  obj  src
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ rm obj/*luiza@DESKTOP-
.
├── include
│   └── float_vector.h
├── obj
├── src
│   └── float_vector.c
3 directories, 2 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ code mytime
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ ls
include  obj  src
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ tree .
.
├── include
│   └── float_vector.h
├── obj
├── src
│   └── float_vector.c
3 directories, 2 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ code mytime.c
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ code mytime.h
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ ls
include  mytime.c  mytime.h  obj  src
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ tree .
.
├── include
│   └── float_vector.h
├── mytime.c
├── mytime.h
├── obj
├── src
│   └── float_vector.c
3 directories, 4 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$

```



The screenshot shows the Visual Studio Code editor with the file `mytime.c` open. The editor has a dark theme and a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The status bar at the bottom shows the file path: `home > luiza > SEIL-LuizaCustodioFreitas > Semana02 > video-02 > ex-06 > C mytime.c`. The code in `mytime.c` is as follows:

```
1 #include "mytime.h"
2
3
4 /* função que marca o tempo atual ("tic" do relógio)
5  deve ser usada para marcar o início do trecho que queremos medir de tempo */
6 timer tic() {
7     timer tic_;
8     gettimeofday(&tic_, NULL); // "marca" o tempo atual
9     return tic_;
10 }
11
12 /* faz a mesma coisa que a função tic, mas que deve ser usada
13  para marcar o final do trecho que queremos medir o tempo.
14  Apenas uma forma semântica de sabermos que faz o "tac" do relógio */
15 timer tac() {
16     return tic();
17 }
18
19 /**
20  * Computa o tempo entre um tic tac;
21  */
22 float comptime(timer tic, timer tac) {
23     float t = ((tac.tv_sec - tic.tv_sec) * 1000.0) +
24              ((tac.tv_usec - tic.tv_usec) * 0.001);
25
26     return t;
27 }
```



The screenshot shows the Visual Studio Code editor with the file `mytime.h` open. The editor has a dark theme and a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The status bar at the bottom shows the file path: `home > luiza > SEIL-LuizaCustodioFreitas > Semana02 > video-02 > ex-06 > C mytime.h`. The code in `mytime.h` is as follows:

```
1 #ifndef MY_TIME_H
2 #define MY_TIME_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <sys/time.h>
7
8 // medindo tempo em C (Linux/Mac)
9 // para windows, veja http://professor.ufabc.edu.br/~daniel.martin/ED/tempo.html
10 typedef struct timeval timer; // adicionando um novo nome para o tipo struct timeval (apenas para facilitar)
11
12
13 /* função que marca o tempo atual ("tic" do relógio)
14  deve ser usada para marcar o início do trecho que queremos medir de tempo */
15 timer tic();
16
17 /* faz a mesma coisa que a função tic, mas que deve ser usada
18  para marcar o final do trecho que queremos medir o tempo.
19  Apenas uma forma semântica de sabermos que faz o "tac" do relógio */
20 timer tac();
21
22
23 /**
24  * Computa o tempo entre um tic tac (em ms);
25  */
26 float comptime(timer tic, timer tac);
27
28 #endif
```



```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ tree .
.
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
├── src
│   ├── float_vector.c
│   └── mytime.c
3 directories, 4 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ gcc -c ./src/float_vector.c -I ./include -o ./obj/float_vector.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ gcc -c ./src/mytime.c -I ./include -o ./obj/mytime.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ tree .
.
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
├── src
│   ├── float_vector.c
│   └── mytime.c
3 directories, 6 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ code app.c
Updating VS Code Server to version 6c3e3dba23e8fadc360aed75ce363ba185c49794
Removing previous installation...
Installing VS Code Server for x64 (6c3e3dba23e8fadc360aed75ce363ba185c49794)
Downloading: 100%
Unpacking: 100%
Unpacked 1797 files and folders to /home/luiza/.vscode-server/bin/6c3e3dba23e8fadc360aed75ce363ba185c49794.
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ ls
app.c  include  obj  src
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ cat app.c
#include "float_vector.h"
#include <stdio.h>

int main() {
    FloatVector *vec = FloatVector_create(2);
    FloatVector_print(vec);

```

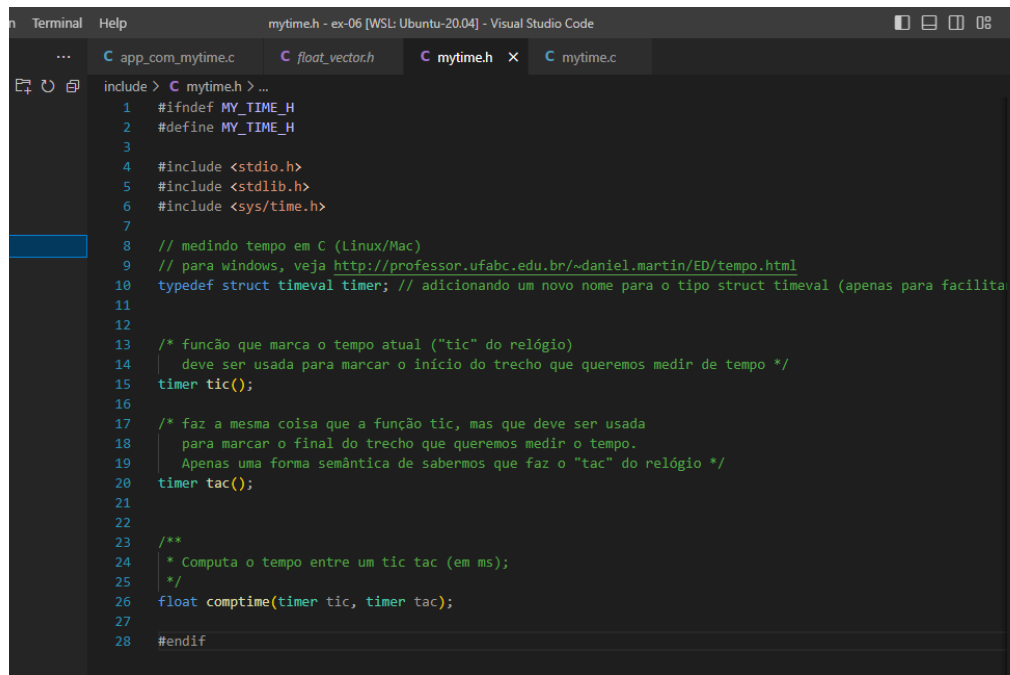
```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ mkdir apps bin
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ tree .
.
├── app.c
├── apps
├── bin
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
├── src
│   ├── float_vector.c
│   └── mytime.c
5 directories, 7 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ mv app.c apps/
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ tree .
.
├── apps
│   └── app.c
├── bin
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
├── src
│   ├── float_vector.c
│   └── mytime.c
5 directories, 7 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ gcc ./apps/app.c ./obj/float_vector.o -I ./include -o ./bin/app
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ ./bin/app
-----
Size: 0
Capacity: 2
-----

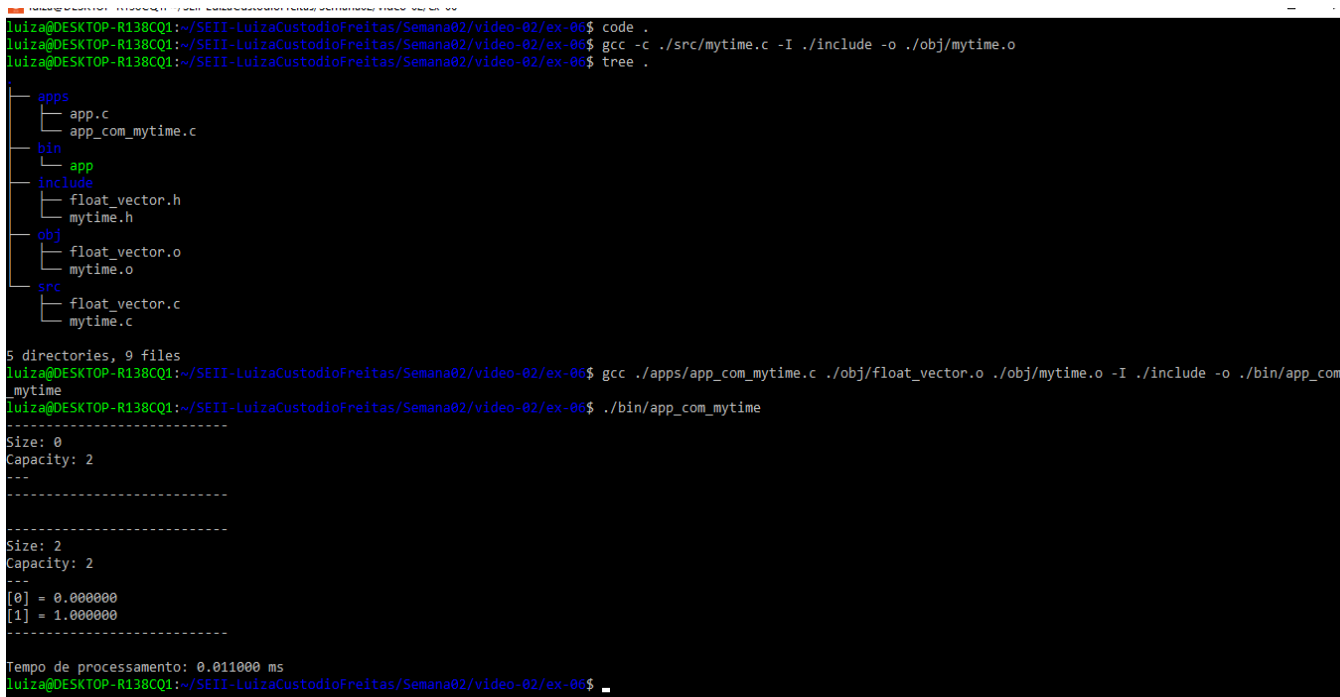
```

```
Help mytime.c - ex-06 [WSL: Ubuntu-20.04] - Visual Studio Code
C app_com_mytime.c C mytime.h C mytime.c X
src > C mytime.c > comptime(timer, timer)
1 #include "mytime.h"
2
3
4 /* função que marca o tempo atual ("tic" do relógio)
5 | deve ser usada para marcar o início do trecho que queremos medir de tempo */
6 timer tic() {
7     timer tic_;
8     gettimeofday(&tic_, NULL); // "marca" o tempo atual
9     return tic_;
10 }
11
12 /* faz a mesma coisa que a função tic, mas que deve ser usada
13 | para marcar o final do trecho que queremos medir o tempo.
14 | Apenas uma forma semântica de sabermos que faz o "tac" do relógio */
15 timer tac() {
16     return tic();
17 }
18
19 /**
20 * Comuta o tempo entre um tic tac;
21 */
22 float comptime(timer tic, timer tac) {
23     float t = ((tac.tv_sec - tic.tv_sec) * 1000.0) +
24             ((tac.tv_usec - tic.tv_usec) * 0.001);
25     return t;
26 }
27 }
```

```
terminal Help app_com_mytime.c - ex-06 [WSL: Ubuntu-20.04] - Visual Studio Code
... C app_com_mytime.c X C mytime.h C mytime.c
apps > C app_com_mytime.c > main0
1 #include "float_vector.h"
2 #include "mytime.h"
3 #include <stdio.h>
4
5 int main() {
6     FloatVector *vec = FloatVector_create(2);
7     FloatVector_print(vec);
8
9     timer t1 = tic();
10    FloatVector_append(vec, 0.0);
11    FloatVector_append(vec, 1.0);
12    FloatVector_print(vec);
13    timer t2 = tac();
14    printf("Tempo de processamento: %f ms\n", comptime(t1, t2));
15
16    FloatVector_destroy(&vec);
17
18    return 0;
19 }
```



```
mytime.h - ex-06 [WSL: Ubuntu-20.04] - Visual Studio Code
... C app_com_mytime.c C float_vector.h C mytime.h X C mytime.c
include > C mytime.h > ...
1  #ifndef MY_TIME_H
2  #define MY_TIME_H
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <sys/time.h>
7
8  // medindo tempo em C (Linux/Mac)
9  // para windows, veja http://professor.ufabc.edu.br/~daniel.martin/ED/tempo.html
10 typedef struct timeval timer; // adicionando um novo nome para o tipo struct timeval (apenas para facilitar)
11
12
13 /* função que marca o tempo atual ("tic" do relógio)
14    deve ser usada para marcar o início do trecho que queremos medir de tempo */
15 timer tic();
16
17 /* faz a mesma coisa que a função tic, mas que deve ser usada
18    para marcar o final do trecho que queremos medir o tempo.
19    Apenas uma forma semântica de sabermos que faz o "tac" do relógio */
20 timer tac();
21
22
23 /**
24  * Computa o tempo entre um tic tac (em ms);
25  */
26 float comptime(timer tic, timer tac);
27
28 #endif
```



```
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ code .
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ gcc -c ./src/mytime.c -I ./include -o ./obj/mytime.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ tree .
.
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
│   └── app
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
└── src
    ├── float_vector.c
    └── mytime.c

5 directories, 9 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ gcc ./apps/app_com_mytime.c ./obj/float_vector.o ./obj/mytime.o -I ./include -o ./bin/app_com_mytime
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ ./bin/app_com_mytime
-----
Size: 0
Capacity: 2
-----
-----
Size: 2
Capacity: 2
-----
[0] = 0.000000
[1] = 1.000000
-----

Tempo de processamento: 0.011000 ms
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$
```

### Vídeo 3:

Makefile 1: pode-se perceber que o comando de compilação da imagem acima se tornou extenso, e com a implementação de cada código esse comando tende a aumentar. Por isso existe o Makefile, uma forma automática de comandos disponível no Visual Studio onde se programa comandos a fim de facilitar a compilação dentro do Linux.

Nas imagens a seguir tem-se a criação das pastas para esse vídeo e a cópia do

último exemplo sendo realizada conforme já visto. Logo, tem-se um comando automatizado no Makefile, onde o “make” realiza a programação feita em “all” e o “hello” é um comando personalizado pelo usuário.

```
luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02/ex-06$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-02$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02$ mkdir video-03
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02$ cd video-03
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03$ mkdir makefile-01
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03$ cd makefile-01luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ cp -R ../../video-02/ex-06/* .
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ lsapps bin include obj src
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ tree .
.
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
│   ├── app
│   └── app_com_mytime
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
└── src
    ├── float_vector.c
    └── mytime.c

5 directories, 10 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ rm bin/* obj/*
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ tree .
.
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
├── src
│   ├── float_vector.c
│   └── mytime.c
└──
```

```
luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ code Makefile
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ make
make: *** No targets specified and no makefile found. Stop.
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ make
echo "Oi oi oi"
Oi oi oi
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ make hello
echo "Hello!"
Hello!
tree .
.
├── Makefile
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
├── src
│   ├── float_vector.c
│   └── mytime.c
└──
```

## Vídeo 4:

Makefile 2: neste exemplo especifica-se os comandos do gcc utilizados no linux de forma automática dentro do Makefile. Com isso, facilita a remoção de arquivos objetos com o “clean” e a compilação dos arquivos fontes com “make”, além de também colocar todos os arquivos em seus respectivos diretórios.

```
luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03/makefile-01$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-03$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02$ mkdir video-04
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02$ cd video-04
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04$ mkdir makefile-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04$ cd makefile-02luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$ tree .
.
├── Makefile
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
└── src
    ├── float_vector.c
    └── mytime.c

5 directories, 7 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$
```

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$ make
gcc -c ./src/float_vector.c -I ./include -o ./obj/float_vector.o
gcc -c ./src/mytime.c -I ./include -o ./obj/mytime.o
gcc ./apps/app.c ./obj/*.o -I ./include -o ./bin/app
gcc ./apps/app_com_mytime.c ./obj/*.o -I ./include -o ./bin/app_com_mytime
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$ tree .
.
├── Makefile
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
│   ├── app
│   └── app_com_mytime
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
└── src
    ├── float_vector.c
    └── mytime.c

5 directories, 11 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$ make run
./bin/app
-----
Size: 0
Capacity: 2
-----
Size: 2
Capacity: 2
[0] = 0.000000
[1] = 1.000000
-----
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$ make clean
rm ./bin/* ./obj/*
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$

```

### Makefile 3:

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-02$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04$ cp -R makefile-02 makefile-03
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04$ cd makefile-03luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$

```

```

terminal  Help  Makefile - ex-06 [WSL: Ubuntu-20.04] - Visual Studio Code
...  _mytime.c  float_vector.h  mytime.h  Makefile - /../makefile-01  Makefile - /../makefile-02
home > luiza > SEII-LuizaCustodioFreitas > Semana02 > video-04 > makefile-03 > Makefile
1  APPS = ./apps
2  BIN = ./bin
3  INCLUDE = ./include
4  OBJ = ./obj
5  SRC = ./src
6
7
8  all: libed myapps
9
10 libed:
11     gcc -c $(SRC)/float_vector.c -I $(INCLUDE) -o $(OBJ)/float_vector.o
12     gcc -c $(SRC)/mytime.c -I $(INCLUDE) -o $(OBJ)/mytime.o
13
14 myapps:
15     gcc $(APPS)/app.c $(OBJ)/*.o -I $(INCLUDE) -o $(BIN)/app
16     gcc $(APPS)/app_com_mytime.c $(OBJ)/*.o -I $(INCLUDE) -o $(BIN)/app_com_mytime
17
18 run:
19     ./bin/app
20
21 clean:
22     rm ./bin/* ./obj/*

```

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03
Makefile
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ make clean
rm ./bin/* ./obj/*
rm: cannot remove './bin/*': No such file or directory
rm: cannot remove './obj/*': No such file or directory
make: *** [Makefile:22: clean] Error 1
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ make libed
gcc -c ./src/float_vector.c -I ./include -o ./obj/float_vector.o
gcc -c ./src/mytime.c -I ./include -o ./obj/mytime.o
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ make myapps
gcc ./apps/app.c ./obj/*.o -I ./include -o ./bin/app
gcc ./apps/app_com_mytime.c ./obj/*.o -I ./include -o ./bin/app_com_mytime
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ tree .
.
├── Makefile
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
│   ├── app
│   └── app_com_mytime
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
└── src
    ├── float_vector.c
    └── mytime.c

5 directories, 11 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$

```

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ make clean
rm ./bin/* ./obj/*
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ tree .
.
├── Makefile
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
├── src
│   ├── float_vector.c
│   └── mytime.c
└──

5 directories, 7 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ make
gcc -c ./src/float_vector.c -I ./include -o ./obj/float_vector.o
gcc -c ./src/mytime.c -I ./include -o ./obj/mytime.o
gcc ./apps/app.c ./obj/*.o -I ./include -o ./bin/app
gcc ./apps/app_com_mytime.c ./obj/*.o -I ./include -o ./bin/app_com_mytime
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ tree .
.
├── Makefile
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
│   ├── app
│   └── app_com_mytime
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
└── src
    ├── float_vector.c
    └── mytime.c

5 directories, 11 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$

```

## Vídeo 5:

Makefile 4: neste último exemplo é possível aprender algumas formas de generalizar os arquivos, para que por exemplo toda vez que um código for introduzido no programa, não seja necessário adicioná-lo nos comando automáticos do Makefile.

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04/makefile-03$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-04$ cd ..
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02$ mkdir video-05
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05$ cd video-05
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05$ mkdir makefile-04
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05$ cd makefile-04luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04$ cp -R ../../video-04/makefile-03/* .
cp: -r not specified; omitting directory '../../video-04/makefile-03/apps'
cp: -r not specified; omitting directory '../../video-04/makefile-03/bin'
cp: -r not specified; omitting directory '../../video-04/makefile-03/include'
cp: -r not specified; omitting directory '../../video-04/makefile-03/obj'
cp: -r not specified; omitting directory '../../video-04/makefile-03/src'
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04$ tree .
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04$ tree .
.
├── Makefile
├── apps
│   ├── app.c
│   └── app_com_mytime.c
├── bin
│   ├── app
│   └── app_com_mytime
├── include
│   ├── float_vector.h
│   └── mytime.h
├── obj
│   ├── float_vector.o
│   └── mytime.o
└── src
    ├── float_vector.c
    └── mytime.c

5 directories, 11 files
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04$

```

```

Help      Makefile - ex-06 [WSL: Ubuntu-20.04] - Visual Studio Code
mytime.c  C float_vector.h  C mytime.h  Makefile - ../../makefile-01  Makefile - ../../makefile-02
home > luiza > SEII-LuizaCustodioFreitas > Semana02 > video-04 > makefile-03 > Makefile
1  APPS = ./apps
2  BIN = ./bin
3  INCLUDE = ./include
4  OBJ = ./obj
5  SRC = ./src
6
7
8  all: libed myapps
9
10 libed: \
11     float_vector.o \
12     mytime.o
13
14 myapps:
15     gcc $(APPS)/app.c $(OBJ)/*.o -I $(INCLUDE) -o $(BIN)/app
16     gcc $(APPS)/app_com_mytime.c $(OBJ)/*.o -I $(INCLUDE) -o $(BIN)/app_com_mytime
17
18
19 %.o: $(SRC)/%.c $(INCLUDE)/%.h
20     gcc -c $< -I $(INCLUDE) -o $(OBJ)/%o
21
22 run:
23     ./bin/app
24
25 clean:
26     rm -rf ./bin/* ./obj/*

```

```

luiza@DESKTOP-R138CQ1: ~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04$ make
gcc -c ./src/float_vector.c -I ./include -o ./obj/float_vector.o
gcc -c ./src/mytime.c -I ./include -o ./obj/mytime.o
gcc ./apps/app.c ./obj/*.o -I ./include -o ./bin/app
gcc ./apps/app_com_mytime.c ./obj/*.o -I ./include -o ./bin/app_com_mytime
luiza@DESKTOP-R138CQ1:~/SEII-LuizaCustodioFreitas/Semana02/video-05/makefile-04$

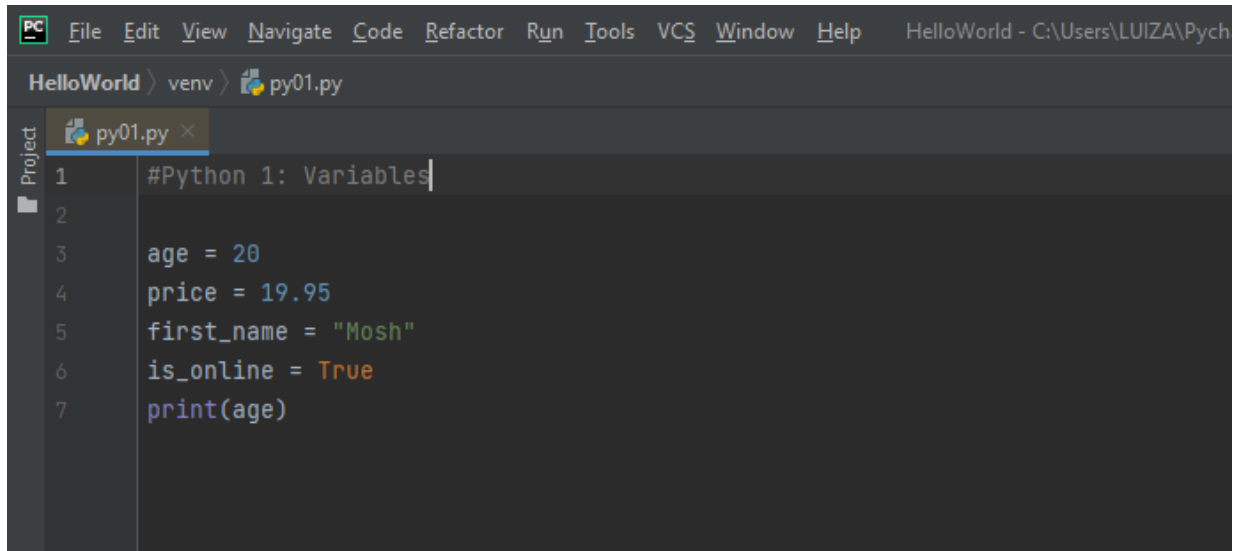
```

### 3. Python

Essa atividade consistiu em um curso introdutório de Python. As seguintes temáticas foram abordadas:

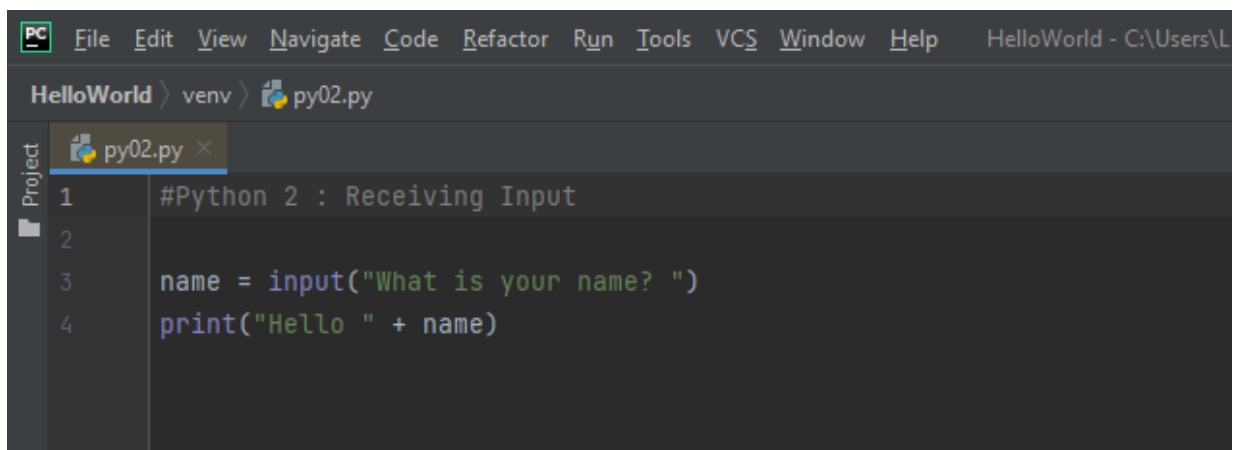


**Variables:** especifica-se 4 tipos de variáveis neste exemplo, sendo elas, int, float, str e bool.

A screenshot of an IDE window titled 'HelloWorld - C:\Users\LUIZA\Pyth'. The file explorer on the left shows a project named 'HelloWorld' with a subdirectory 'venv' containing a file 'py01.py'. The main editor area shows the code for 'py01.py' with the following content:

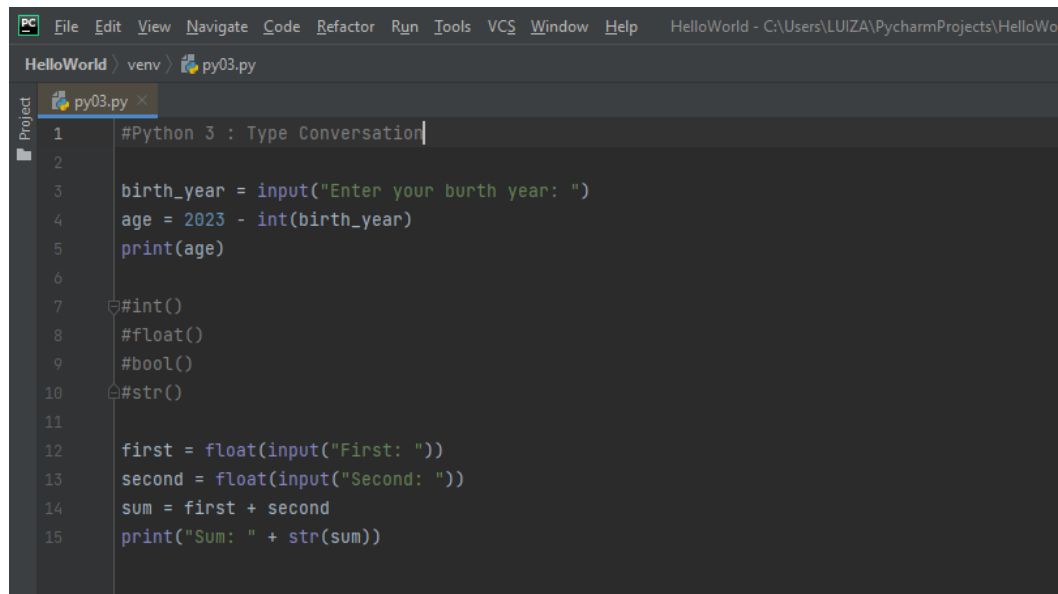
```
1 #Python 1: Variables
2
3 age = 20
4 price = 19.95
5 first_name = "Mosh"
6 is_online = True
7 print(age)
```

**Receiving Input:** neste exemplo é explicado como interagir com o usuário, por meio de respostas deste, por meio do “input()”.

A screenshot of an IDE window titled 'HelloWorld - C:\Users\LUIZA\Pyth'. The file explorer on the left shows a project named 'HelloWorld' with a subdirectory 'venv' containing a file 'py02.py'. The main editor area shows the code for 'py02.py' with the following content:

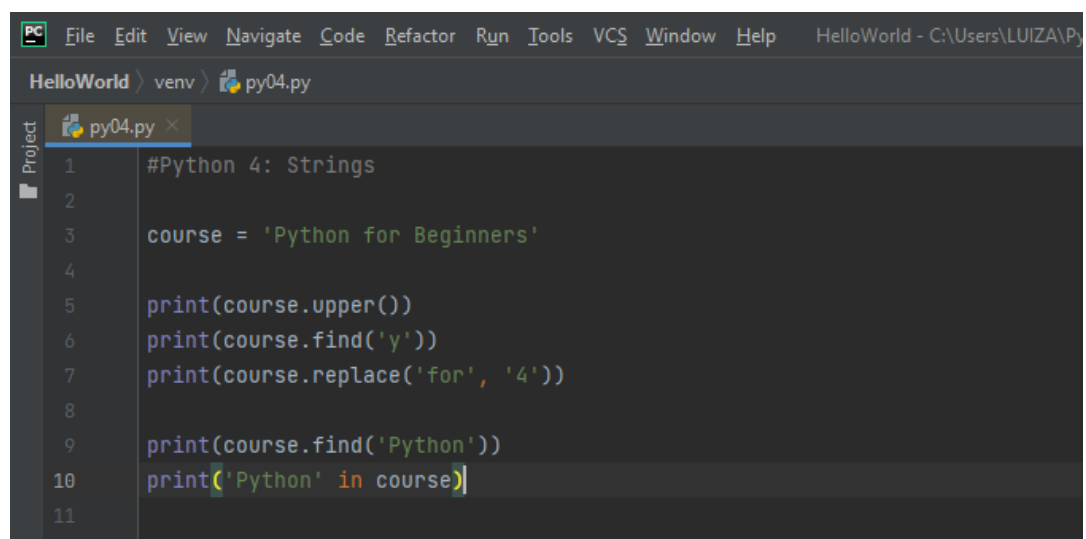
```
1 #Python 2 : Receiving Input
2
3 name = input("What is your name? ")
4 print("Hello " + name)
```

**Type Conversion:** é feito um exemplo de conversação do programa com o usuário, perguntando o ano de nascimento e devolvendo a idade como resposta. Ressalta-se a necessidade de se fazer somas ou subtrações com variáveis iguais.



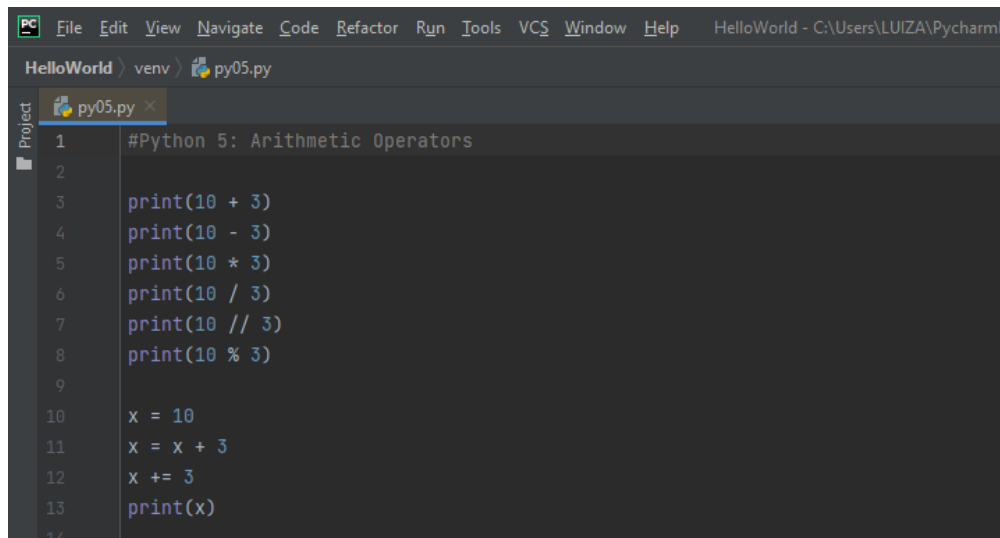
```
1 #Python 3 : Type Conversation
2
3 birth_year = input("Enter your burth year: ")
4 age = 2023 - int(birth_year)
5 print(age)
6
7 #int()
8 #float()
9 #bool()
10 #str()
11
12 first = float(input("First: "))
13 second = float(input("Second: "))
14 sum = first + second
15 print("Sum: " + str(sum))
```

**Strings:** apresenta uma string (palavra) e algumas funções utilizando “.função”. Como é possível observar abaixo onde “print(course.upper())” que devolve o texto em course de forma maiúscula.



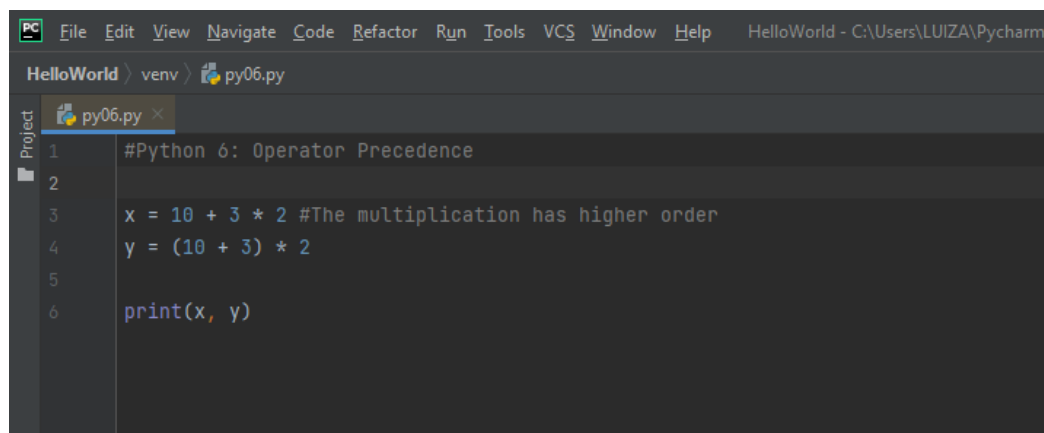
```
1 #Python 4: Strings
2
3 course = 'Python for Beginners'
4
5 print(course.upper())
6 print(course.find('y'))
7 print(course.replace('for', '4'))
8
9 print(course.find('Python'))
10 print('Python' in course)
11
```

**Arithmetic Operators:** demonstra as principais operações aritméticas.



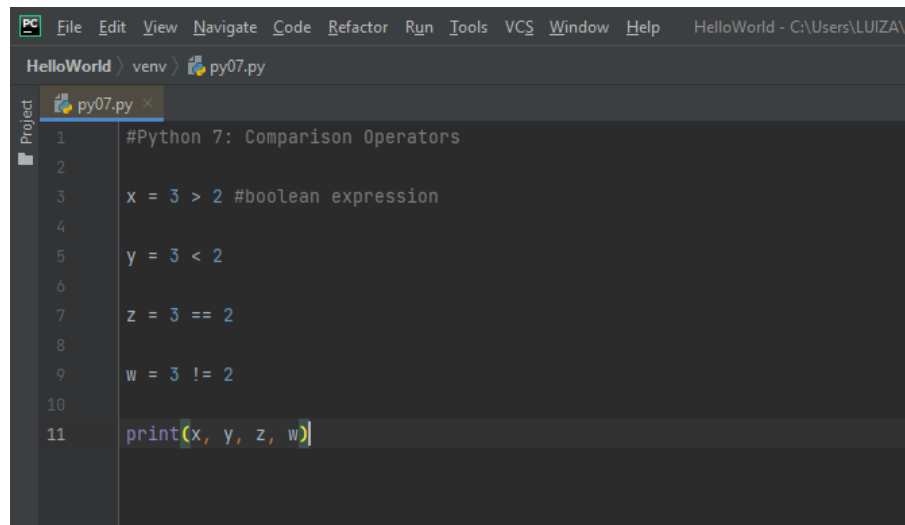
```
1 #Python 5: Arithmetic Operators
2
3 print(10 + 3)
4 print(10 - 3)
5 print(10 * 3)
6 print(10 / 3)
7 print(10 // 3)
8 print(10 % 3)
9
10 x = 10
11 x = x + 3
12 x += 3
13 print(x)
```

**Operator Precedence:** explica a questão de prioridade, onde divisão (/) e multiplicação (\*) são realizadas primeiro do que soma (+) e subtração (-), porém a utilização de parênteses é prioritária.



```
1 #Python 6: Operator Precedence
2
3 x = 10 + 3 * 2 #The multiplication has higher order
4 y = (10 + 3) * 2
5
6 print(x, y)
```

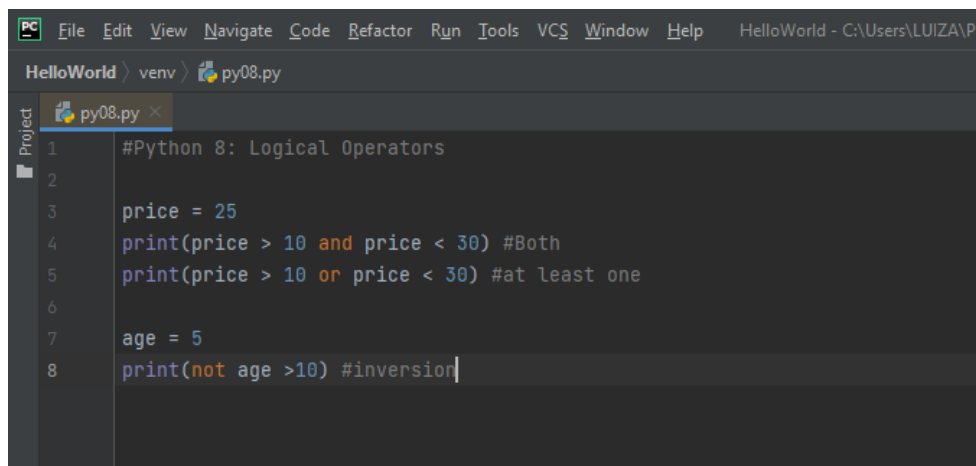
**Comparison Operators:** mostra os operadores de comparação, maior (>), menor (<), igual (==) e diferente (!=). O programa entende esses operadores como perguntas e devolve uma resposta booleana (True ou False).



The screenshot shows a code editor window titled 'HelloWorld - C:\Users\LUIZA\'. The editor is open to a file named 'py07.py'. The code in the editor is as follows:

```
1 #Python 7: Comparison Operators
2
3 x = 3 > 2 #boolean expression
4
5 y = 3 < 2
6
7 z = 3 == 2
8
9 w = 3 != 2
10
11 print(x, y, z, w)
```

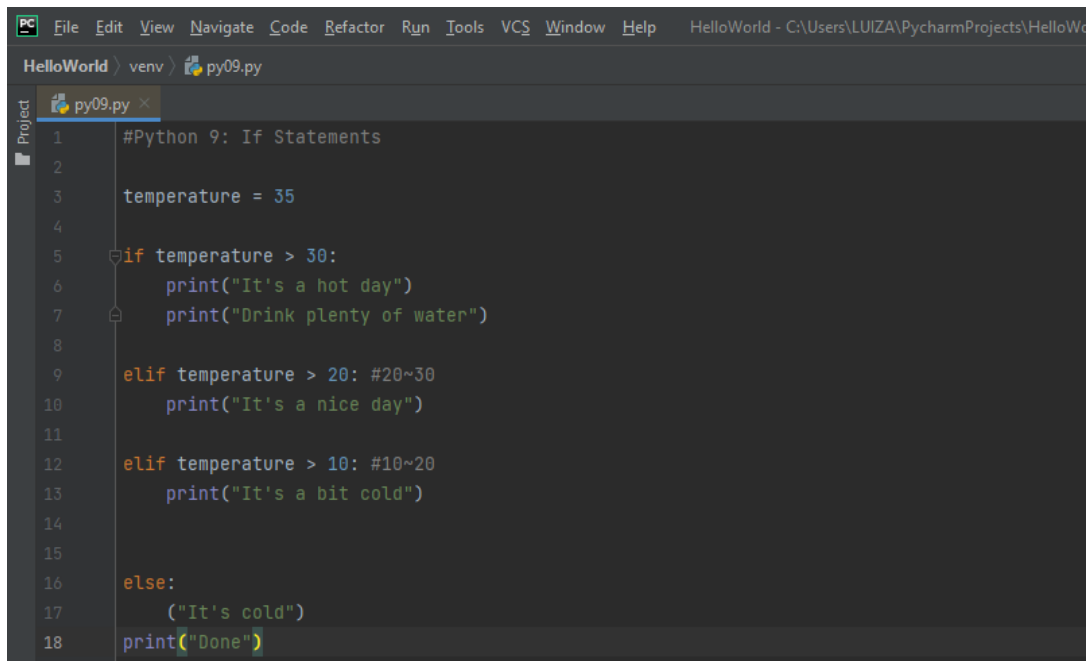
**Logical Operators:** exemplo onde tem-se os operadores lógicos and (todos tem que apresentar a característica desejada), or (pelo menos um tem que apresentar a característica desejada) e not (tem que apresentar o inverso da característica desejada).



The screenshot shows a code editor window titled 'HelloWorld - C:\Users\LUIZA\'. The editor is open to a file named 'py08.py'. The code in the editor is as follows:

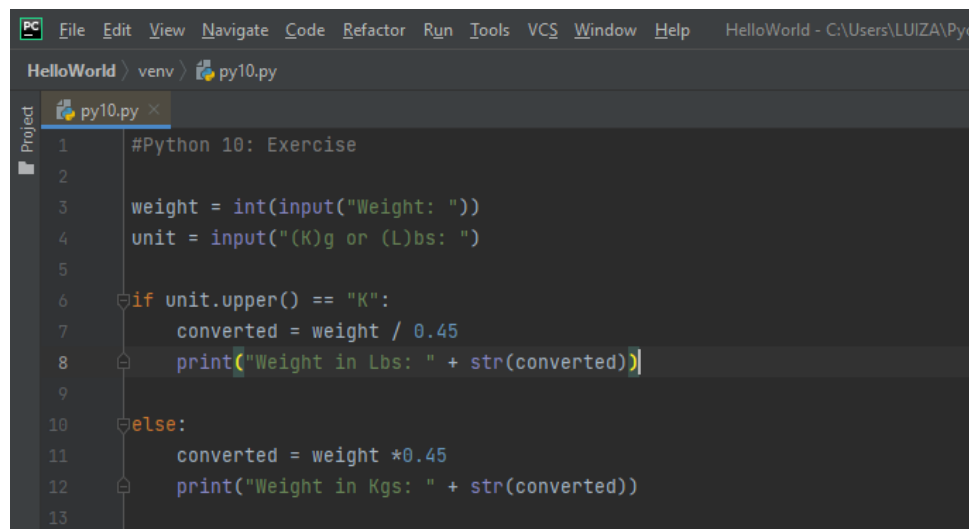
```
1 #Python 8: Logical Operators
2
3 price = 25
4 print(price > 10 and price < 30) #Both
5 print(price > 10 or price < 30) #at least one
6
7 age = 5
8 print(not age > 10) #inversion
```

**If Statements:** introduz o comando if (se) e elif (se não) para utilizar em lógicas onde respostas desejadas para perguntas específicas.



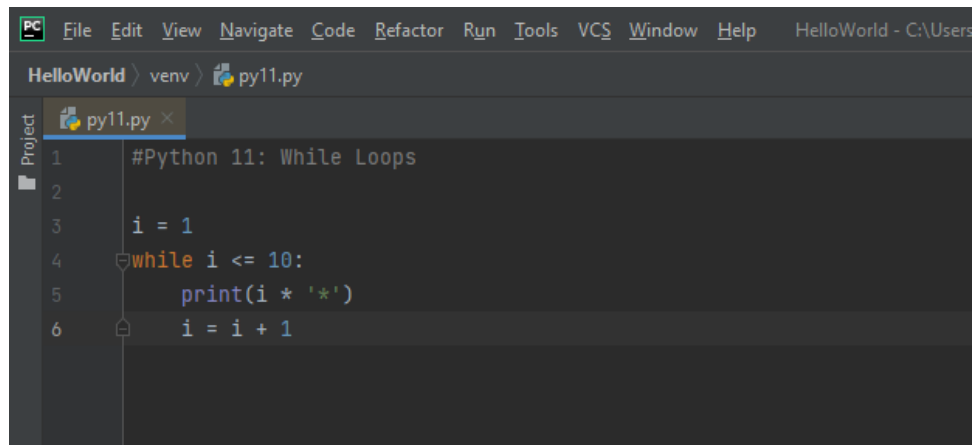
```
1 #Python 9: If Statements
2
3 temperature = 35
4
5 if temperature > 30:
6     print("It's a hot day")
7     print("Drink plenty of water")
8
9 elif temperature > 20: #20~30
10    print("It's a nice day")
11
12 elif temperature > 10: #10~20
13    print("It's a bit cold")
14
15
16 else:
17     ("It's cold")
18 print("Done")
```

**Exercise:** exercício de conversão de massa utilizando tudo que foi aprendido até o momento.



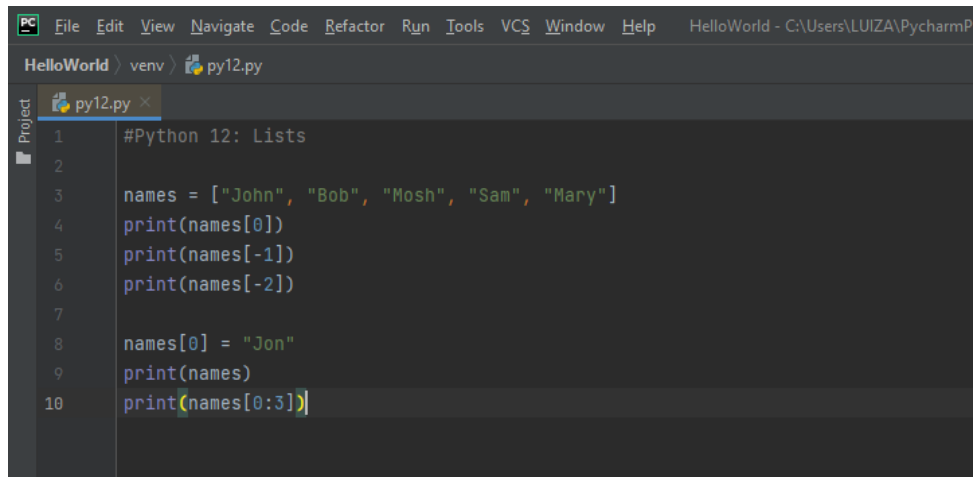
```
1 #Python 10: Exercise
2
3 weight = int(input("Weight: "))
4 unit = input("(K)g or (L)bs: ")
5
6 if unit.upper() == "K":
7     converted = weight / 0.45
8     print("Weight in Lbs: " + str(converted))
9
10 else:
11     converted = weight * 0.45
12     print("Weight in Kgs: " + str(converted))
13
```

**While Loops:** utilizando o comando while, enquanto uma condição for verdadeira a resposta será a programada em uma função.



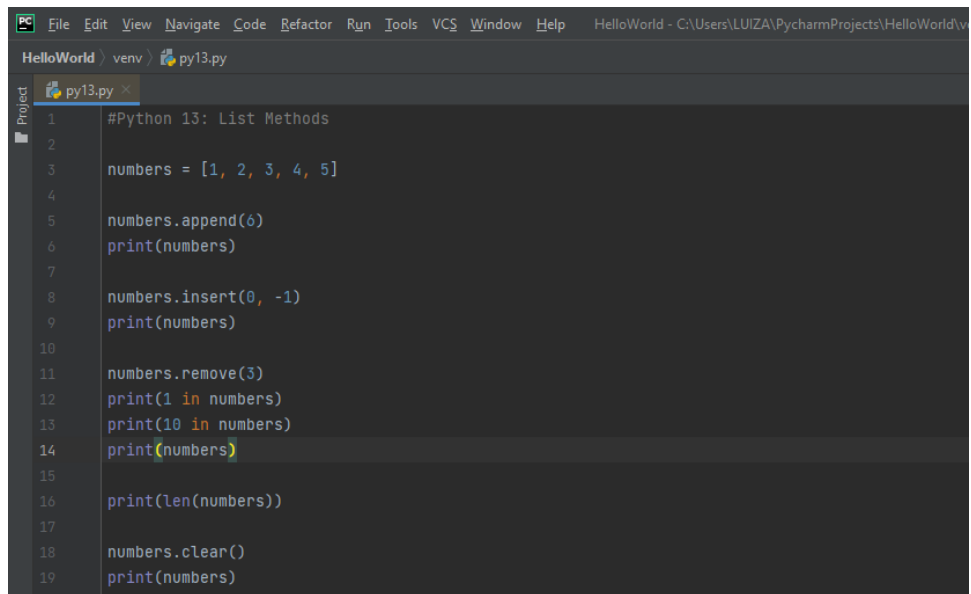
```
1 #Python 11: While Loops
2
3 i = 1
4 while i <= 10:
5     print(i * '*')
6     i = i + 1
```

**Lists:** utilizando [] é possível fazer listas de variáveis. Com [0] pega-se a variável da primeira posição, [-1] da última e [-2] da penúltima.



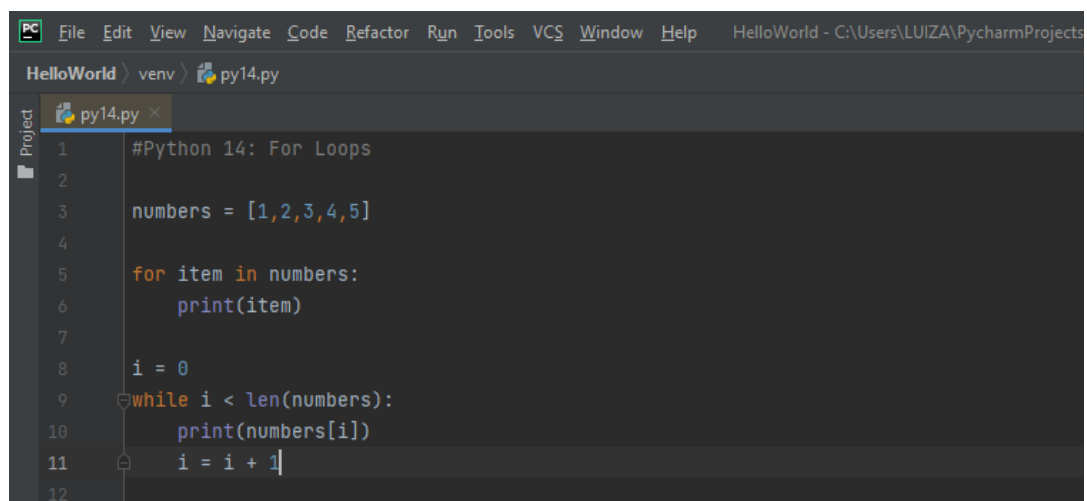
```
1 #Python 12: Lists
2
3 names = ["John", "Bob", "Mosh", "Sam", "Mary"]
4 print(names[0])
5 print(names[-1])
6 print(names[-2])
7
8 names[0] = "Jon"
9 print(names)
10 print(names[0:3])
```

**List Methods:** neste exemplo é utilizadas algumas funções para utilização em listas, por exemplo o “numbers.remove()” que remove da lista a variável que está especificada no parênteses.



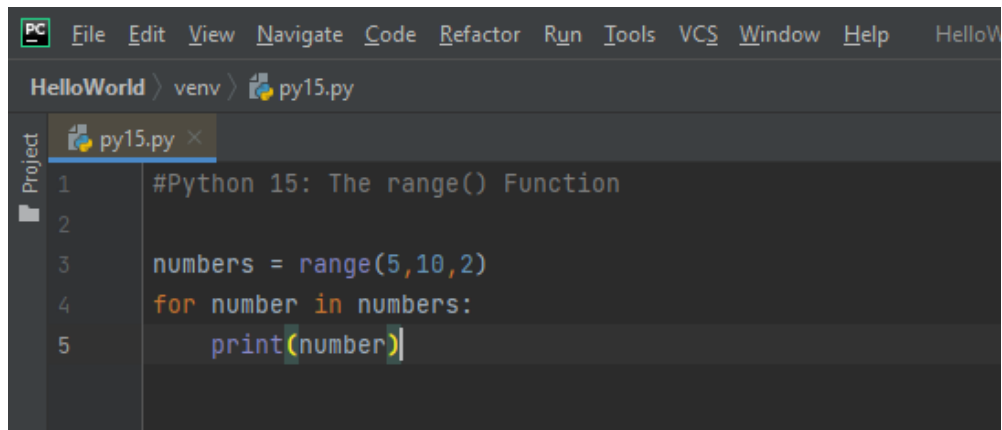
```
1 #Python 13: List Methods
2
3 numbers = [1, 2, 3, 4, 5]
4
5 numbers.append(6)
6 print(numbers)
7
8 numbers.insert(0, -1)
9 print(numbers)
10
11 numbers.remove(3)
12 print(1 in numbers)
13 print(10 in numbers)
14 print(numbers)
15
16 print(len(numbers))
17
18 numbers.clear()
19 print(numbers)
```

**For Loops:** o “for” é um loop utilizado para dizer “para cada” como no exemplo feito em que “para cada” item na lista “numbers” terá uma função (print(item)).



```
1 #Python 14: For Loops
2
3 numbers = [1,2,3,4,5]
4
5 for item in numbers:
6     print(item)
7
8 i = 0
9 while i < len(numbers):
10     print(numbers[i])
11     i = i + 1
12
```

**The range():** utilizado quando é necessário uma lista consideravelmente extensa, onde “range(5,10,2)” é impresso uma lista de 5 a 10 porém com um salto de 2, ou seja, a resposta seria: 5,7,9.

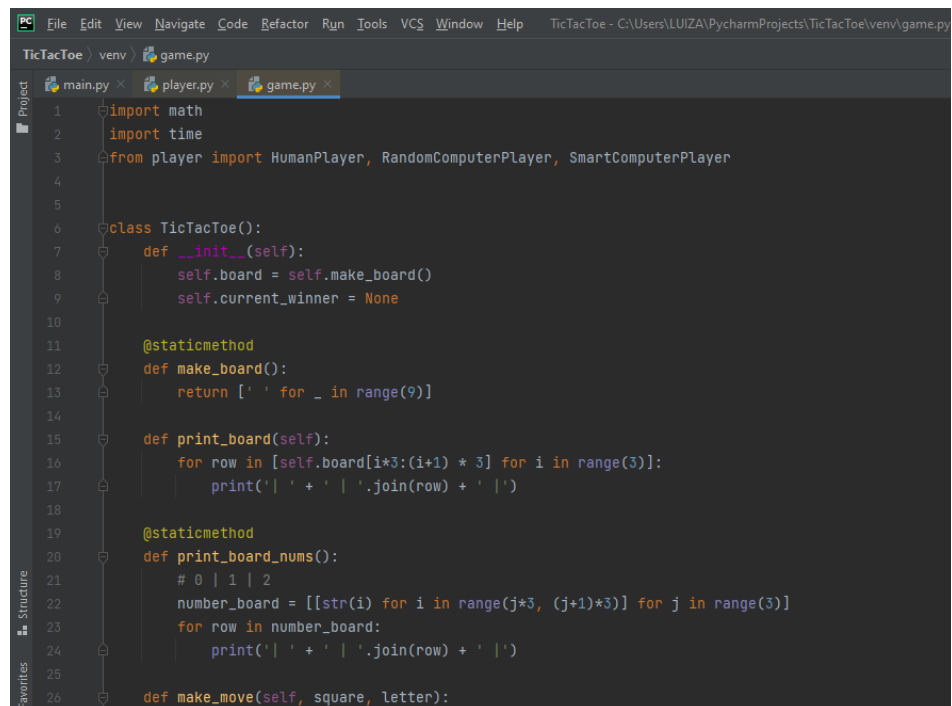


```
1 #Python 15: The range() Function
2
3 numbers = range(5,10,2)
4 for number in numbers:
5     print(number)
```

#### 4. Projetos utilizando Python

Dois projetos foram disponibilizados para serem aprendidos, sendo estes Tic Tac Toe e Sudoku. Seguem prévias dos códigos postados no GitHub:

##### Tic Tac Toe:



```
1 import math
2 import time
3 from player import HumanPlayer, RandomComputerPlayer, SmartComputerPlayer
4
5
6 class TicTacToe():
7     def __init__(self):
8         self.board = self.make_board()
9         self.current_winner = None
10
11     @staticmethod
12     def make_board():
13         return [' ' for _ in range(9)]
14
15     def print_board(self):
16         for row in [self.board[i*3:(i+1) * 3] for i in range(3)]:
17             print('| ' + ' | '.join(row) + ' | ')
18
19     @staticmethod
20     def print_board_nums():
21         # 0 | 1 | 2
22         number_board = [[str(i) for i in range(j*3, (j+1)*3)] for j in range(3)]
23         for row in number_board:
24             print('| ' + ' | '.join(row) + ' | ')
25
26     def make_move(self, square, letter):
```

##### Sudoku:



```
Sudoku - C:\Users\LUIZA\PycharmProjects\Sudoku\venv\sud
Sudoku venv sudoku.py
main.py x sudoku.py x sud_empty.py x
22 # that number must not be repeated in the row, column, or 3x3 square that it appears in
23
24 # let's start with the row
25 row_vals = puzzle[row]
26 if guess in row_vals:
27     return False # if we've repeated, then our guess is not valid!
28
29 # now the column
30 col_vals = []
31 # for i in range(9):
32     col_vals.append(puzzle[i][col])
33 col_vals = [puzzle[i][col] for i in range(9)]
34 if guess in col_vals:
35     return False
36
37 # and then the square
38 row_start = (row // 3) * 3 # 10 // 3 = 3, 5 // 3 = 1, 1 // 3 = 0
39 col_start = (col // 3) * 3
40
41 for r in range(row_start, row_start + 3):
42     for c in range(col_start, col_start + 3):
43         if puzzle[r][c] == guess:
44             return False
45
46 return True
```

## **Referências**

Compilando Códigos C: do Zero ao Jedi:

<https://www.youtube.com/watch?v=hrPxxwKtedCc&list=PL3ZslI15yo2pCf0WpZmV-ga02kMPxKH3p&index=1>

Python for Beginners - Learn Python in 1 Hour:

<https://www.youtube.com/watch?v=kqtD5dpn9C8>

12 Beginner Python Projects - Coding Course:

<https://www.youtube.com/watch?v=8ext9G7xspg>