GUIA DE LABORATÓRIO 2.2
CONTROLO DA EXECUÇÃO, *SCRIPTING* E BIBLIOTECA PADRÃO

EXERCÍCIOS DE REVISÃO

1. ~17.

2.
```
str1 = "ALBERTO"
str2 = "XALBERTO"[1:]
```

3. `is` ▫ Compara IDs (se dois objectos são o mesmo objecto, ie, se têm o mesmo ID)
`==` ▫ Compara o conteúdo

4. Imutável: Objecto que não podes er alterado/modificado
`str, tuple, frozenset, namedtuple`

5.

5.1
```
b, c, d = True, False, False
```

5.2
```
x, y, z = True, False, True
```

5.3
```
c = 'F'
```

5.4
```
m = 30.5    # 2.5 + 2 + 1 + 25
```

6.

6.1
```
vals[2:-2]            = [24]
```

6.2
```
vals[:-len(vals)]     = []
```

6.3
```
vals[:-len(vals) + 1] = [22]
```

6.4
```
vals[1:], vals[0] = vals[:-1], vals[-1]     vals = [26, 22, 23, 24, 25]
```

6.5
```
vals[:-1], vals[-1] = vals[1:], vals[0]     vals = [23, 24, 25, 26, 22]
```

7.
```
vals[-1:] = [vals[-1], *vals2]
```
Strings e tuplos são imutáveis.

8.

8.1 e 8.2 ▫ Porque o tuplo é uma estrutura de dados imutável

8.4 ▫ Porque o segundo elemento do tuplo é uma string, e as strings também são imutáveis

9.

`x=3,5/2  # isto é um tuplo` `print(x)`	`(3, 2.5)`
`codigo = {'A': 19, 'B': 20}` `print(list(codigo))`	`['A', 'B']`
`codigo = {'A': 19, 'B': 20}` `print(codigo['B'], codigo.get('C'), codigo.get('C', 21))`	`20 None 21`
`codigo = {'A': 19, 'B': 20}` `d = dict(a=list(codigo.values())[0],` `  b=list(codigo.values())[-1])` `print(d)`	`{'a': 19, 'b': 20}` ou `{'a': 20, 'b': 19}`
`processos = {'ls': 192, 'grep': 321, 'init': 1}` `print('ls' in processos, 321 in processos)` `print((192 in processos)*2)` `processos.update(ls=292, mkfs=19)` `print(list(processos.items()))`	`True False` `0` `[('ls', 292), ('grep', 321),` ` ('init', 1), ('mkfs', 19)]`
`pessoa = {'nome': 'alberto', 'idade': 23, 'altura': 190}` `print(pessoa['nome'].upper())` `print(pessoa.get(pessoa['idade'], 'altura'))`	`ALBERTO` `altura`
`x = 'ABC-DEF-GHI--JKL'.split('-')` `print("|{:^5}|".format(x[2:5][2]).replace(chr(32), '.'))`	`['ABC', 'DEF', 'GHI', '', 'JKL']` `x[2:5] = ['GHI', '', 'JKL']` `x[2:5][2] = 'JKL'` `' JKL '.replace(' ', '.')` `'.JKL.'` `---` `R: |.JKL.|`
`dados = [1993, "MANEL[1, 7, 8, 2]I", {'m': 12, 'n': 14}]` `dados[0] += 17` `print(str(dados[0])[2])` `print(eval(dados[1][5:-1])[-3:-1])`	`1` `[7, 8]`
`formulas = [[1, -1, 2], [3, 2, 2]]` `m = formulas[:]` `m[0] = [9, 9, 9]` `print(m, '--', formulas[0])` `m = formulas[:]` `m[0][0] = 9` `print(m, '--', formulas[0])`	`[[9, 9, 9], [3, 2, 2]] -- [1, -1, 2]` `[[9, -1, 2], [3, 2, 2]] -- [9, -1, 2]`

`vals = [1, 2, 3]` `print(vals[-1:])` `vals[-1:] = [4, 5]` `print(vals)`	`[3]` `[1, 2, 4, 5]`
`t = (1, 2, 3)` `print(t[-1:])` `vals = [10, 20, 30]` `vals[-2:] = t` `print(vals)`	`(3,)` `[10, 1, 2, 3]`
`txt = ''` `nums = [10, 11]` `if nums:` `  print("Um")` `  if not txt:` `  print("Dois")` `  txt = 'abc'` `  nums = []` `  else:` `  print("Três")` `  txt = 'xey'` `  nums[-1:] = [12]` `txt = txt.replace('a', '').replace('e', '')` `print("Quatro" if len(nums) else "Cinco")` `print(txt if len(nums) == 0 else nums)`	`Um` `Dois` `Cinco` `bc`

10. Possui 4 partes: INICIALIZAÇÃO, CONDIÇÃO, BLOCO DE INSTRUÇÕES e ACTUALIZAÇÃO. Todas as partes são opcionais.

11.

PYTHON:	***LINGUAGENS DERIVADAS DE C (C++ neste caso):***
`for p in range(0, 101, 2):` `  print(p)`	`for (int p = 0; p <= 100; p += 2) {` `  cout << p << endl;` `}`

12.

`i = 0` `while i < 10:` `  print(i)` `  i += 2`	`for i in range(0, 10, 2):` `  print(i)`

`for k in range(10, 2, -2):` `  print(k)`	*`k = 10`* *`while k > 2:`* *`  print(k)`* *`  k -= 2`*
`nums = (10, 2, 7, 9, 6, 5, 8)` `i = 0` `while i < len(nums):` `  print(nums[i])` `  i += 2`	*`nums = (10, 2, 7, 9, 6, 5, 8)`* *`for i in range(0, len(nums), 2):`* *`  print(nums[i])`* *`---`* *`for n in nums[::2]:`* *`  print(n)`*