



POLYTECHNIC
FROM PORTO
SCHOOL
HIGHER
MEDIA ARTS
AND DESIGN

P.PORTO

ALGORITHMY AND DATA STRUCTURES

MODULE II – CONTROL STRUCTURES

TECHNOLOGIES AND INFORMATION SYSTEMS FOR THE WEB

1. Decision Structures

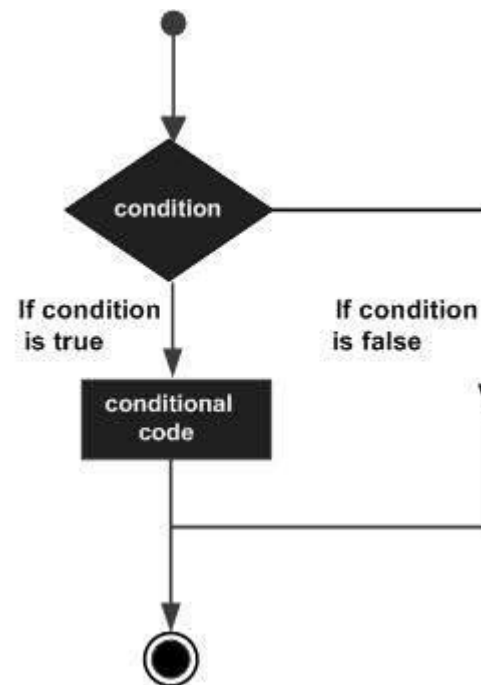
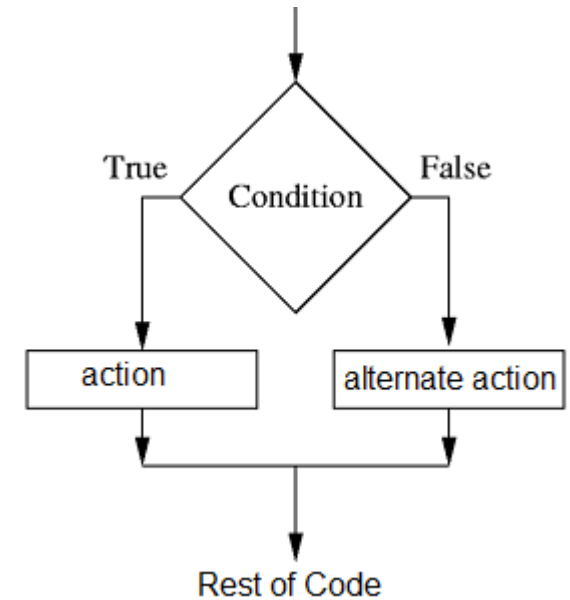
- ❑ if
- ❑ if-else
- ❑ if-elif
- ❑ match-case

2. Repetition Structures

- ❑ for
- ❑ while

3. Cycle breaks

- ❑ break
- ❑ continues



❖ Decision Structures

if

Evaluate conditions and allow executing actions (one or more instructions) based on the result of the condition evaluation

Specifies action to be performed when a certain condition is met (*if*)

```
FICHA 02 > Exemplo.py > ...  
1  
2 num1 = int(input("Primeiro Número:"))  
3 num2 = int(input("Segundo Numero:"))  
4  
5 # if  
6 if num1 > num2:  
7     print("o maior é {0}" .format(num1))  
8  
9  
10
```

C:\WINDOWS\py.exe
Primeiro Número:10
Segundo Numero:7
o maior é 10
-

❖ Decision Structures

if-else

Evaluate conditions and allow executing actions (one or more instructions) based on the result of the condition evaluation

- Specifies action to perform when the condition is true (*if*)
- Define action to take when the condition fails (*else*)

FICHA 02 > Exemplo.py > ...

```
1
2  num1 = int(input("Primeiro Número:"))
3  num2 = int(input("Segundo Numero:"))
4
5
6
7  if num1 > num2:
8      print("o maior é {0}" .format(num1))
9  else:
10     print("o maior é {0}" .format(num2))
11
12
```

C:\WINDOWS\py.exe

```
Primeiro Número:10
Segundo Numero:20
o maior é 20
```

❖ Decision Structures

if-elif-else

Evaluate conditions and allow executing actions (one or more instructions) based on the result of the condition evaluation

- Specifies action to perform when the condition is true (*if*)
- Defines new condition when the previous one fails (*elif*)
- Defines action to take when all previous conditions fail (*else*)

```
1
2 num1 = int(input("Primeiro Número:"))
3 num2 = int(input("Segundo Numero:"))
4
5
6 if num1 > num2:
7     print("o maior é {0}" .format(num1))
8 elif num1 == num2:
9     print("os números {0} são iguais" .format(num1))
10 else:
11     print("o maior é {0}" .format(num2))
12
13
```

C:\WINDOWS\py.exe

Primeiro Número:10
Segundo Numero:10
os números 10 são iguais

❖ Decision Structures

if-elif-else

Evaluate conditions and allow executing actions (one or more instructions) based on the result of the condition evaluation

- Specifies action to perform when the condition is true (*if*)
- Defines new condition when the previous one fails (*elif*)
- Defines action to take when all previous conditions fail (*else*)

```
1
2 num1 = int(input("Primeiro Número:"))
3 num2 = int(input("Segundo Numero:"))
4
5
6 if num1 > num2:
7     print("o maior é {0}" .format(num1))
8 elif num1 == num2:
9     print("os números {0} são iguais" .format(num1))
10 else:
11     print("o maior é {0}" .format(num2))
12
13
```

C:\WINDOWS\py.exe
Primeiro Número:15
Segundo Numero:25
o maior é 25

❖ Decision Structures

if-elif-else

Evaluate conditions and allow executing actions (one or more instructions) based on the result of the condition evaluation

- Specifies action to perform when the condition is true (*if*)
- Defines new condition when the previous one fails (*elif*)
- Defines action to take when all previous conditions fail (*else*)

```
num1 = int(input("Primeiro Número:"))
num2 = int(input("Segundo Numero:"))
num3 = int(input("Terceiro Numero:"))

if num1 > num2 and num1 > num3:
    print("o maior é {0}" .format(num1))
elif num2 > num1 and num2 > num3:
    print("o maior é {0}" .format(num2))
else:
    print("o maior é {0}" .format(num3))
```

C:\WINDOWS\py.exe

```
Primeiro Número:10
Segundo Numero:29
Terceiro Numero:25
o maior é 29
```

❖ Decision Structures

if-elif-else

Synthetic writing of conditional structures

```
FICHA 02 > Exemplo.py > ...  
1  
2 num1 = int(input("Primeiro Número:"))  
3 num2 = int(input("Segundo Numero:"))  
4  
5 if num1 > num2:  
6     print ("o maior é {0}" .format(num1))  
7  
8 if num1 > num2: print ("o maior é {0}" .format(num1))  
9  
10
```


❖ Decision Structures

if-elif-else

Synthetic writing of conditional structures

```
num1 = int(input("Primeiro Número:"))  
num2 = int(input("Segundo Numero:"))  
  
if num1 > num2:  
|   print ("o maior é {0}" .format(num1))  
else:  
|   print ("o maior é {0}" .format(num2))
```

```
print ("o maior é {0}" .format(num1)) if num1 > num2 else print ("o maior é {0}" .format(num2))
```

❖ Decision Structures

if-elif-else

Synthetic writing of conditional structures

```
num1 = int(input("Primeiro Número:"))
num2 = int(input("Segundo Numero:"))

if num1 > num2:
    print("o maior é {0}" .format(num1))
elif num1 == num2:
    print("os números {0} são iguais" .format(num1))
else:
    print("o maior é {0}" .format(num2))

print ("o maior é {0}" .format(num1)) if num1 > num2 else print("os números {0} são iguais" .format(num1)) if num1 == num2 else print
["o maior é {0}" .format(num2)]
```

input()

C:\WINDOWS\py.exe

Primeiro Número:10
Segundo Numero:25
o maior é 25

❖ Decision Structures

match-case

From version 3.10 upwards, Python has implemented a switch case feature called “structural pattern matching”.

You can implement this feature with the **match** and **case** keywords

```
match term:
    case pattern-1:
        action-1
    case pattern-2:
        action-2
    case pattern-3:
        action-3
    case _:
        action-default
```

Note that the **underscore** symbol is what you use to define a default case. It is executed if none of the previous options are checked!

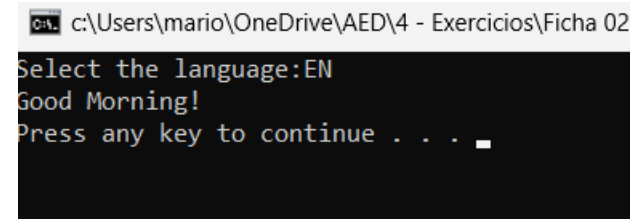
❖ Decision Structures

match-case

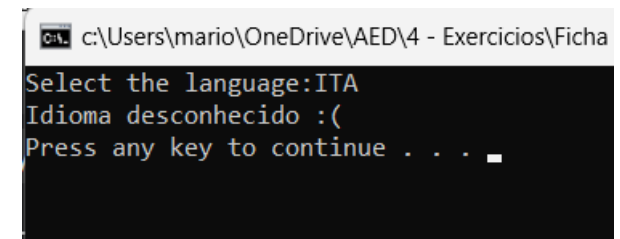
From version 3.10 upwards, Python has implemented a switch case feature called “structural pattern matching”.

You can implement this feature with the **match** and **case** keywords

```
lang = input("Select the language:")
match lang:
    case "PT":
        print("Bom dia!")
    case "EN":
        print("Good Morning!")
    case "FR":
        print("Bonjour!")
    case "GER":
        print("Guten Morgen!")
    case _:
        print("Idioma desconhecido :(")
```



A terminal window titled "c:\Users\mario\OneDrive\AED\4 - Exercicios\Ficha 02" showing the program's execution. The user has entered "EN" in response to the prompt "Select the language:". The program outputs "Good Morning!" and then "Press any key to continue . . .".



A terminal window titled "c:\Users\mario\OneDrive\AED\4 - Exercicios\Ficha" showing the program's execution. The user has entered "ITA" in response to the prompt "Select the language:". The program outputs "Idioma desconhecido :(" and then "Press any key to continue . . .".

We use the **match – case structure** when we have a “closed list” of valid options, to test!!

❖ Repetition | Iterative Structures *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

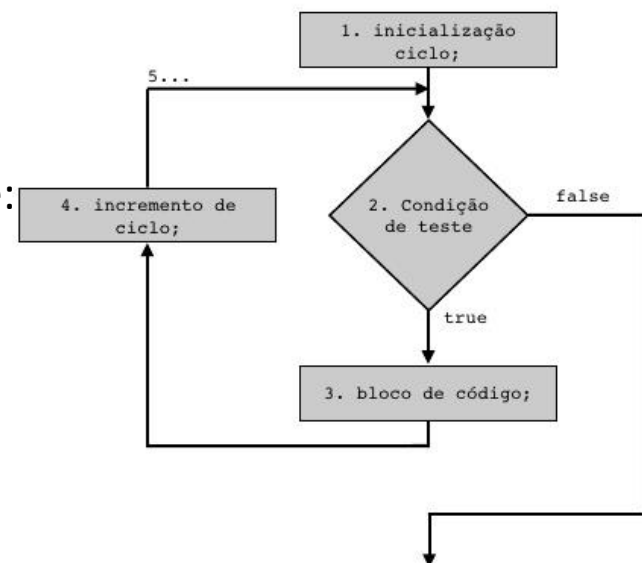
The iteration (repetition) of a cycle *for* can be associated with:

☐ Numeric values (*range*)

☐ Text (strings)

☐ Sequences values
(perform repeat to each value of a sequence:
an array, list, tuple, etc.)

for: como funciona em fluxograma



❖ Repetition Structures | Iterative *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

range: allows you to specify the number of times the cycle repeats

Variable
accountant
of the cycle

Repeat the cycle **10 times**. Starts with $i=0$ and ends when $i=9$

```
Exemplos.py > ...  
1 # Exemplos de estruturas repetitivas (iterativas) for  
2  
3 for i in range (10):  
4     print(i)  
5  
6  
7  
8  
9
```

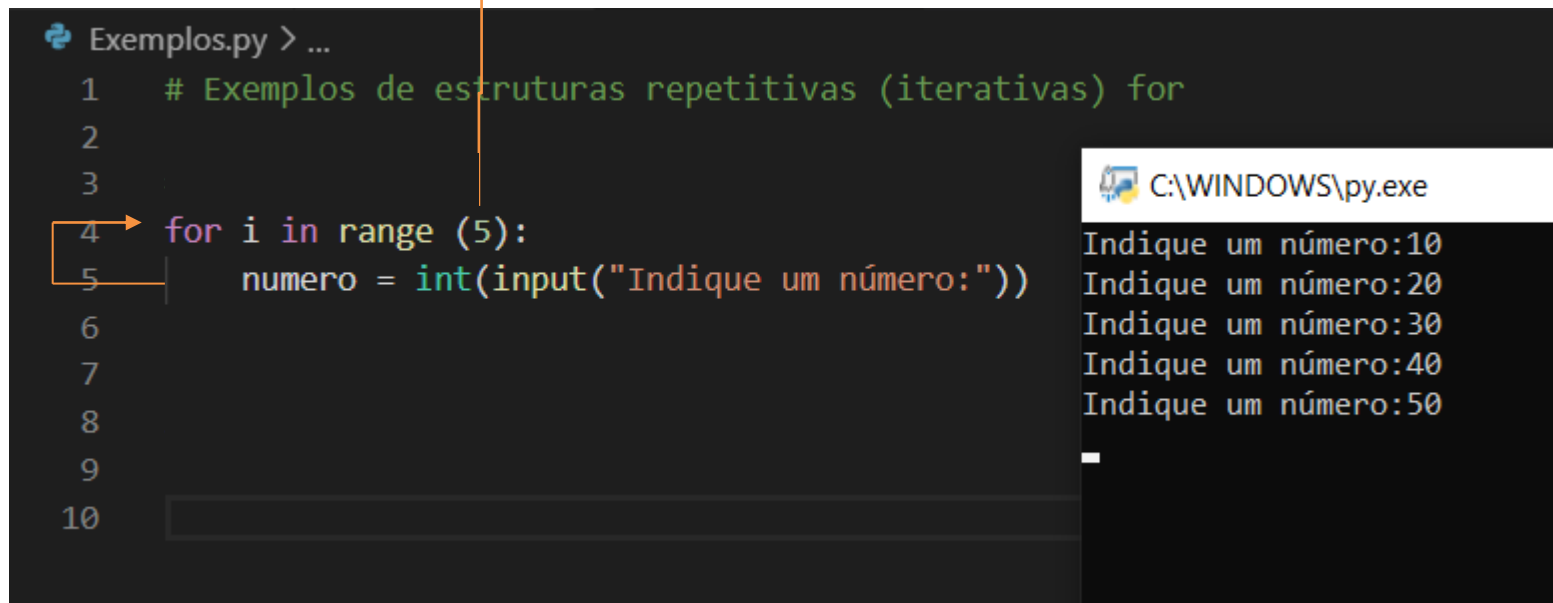
C:\WINDOWS\py.exe
0
1
2
3
4
5
6
7
8
9
—

❖ Repetition Structures | Iterative *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

range: allows you to specify the number of times the cycle repeats

Repeat the cycle **5 times**. Starts with $i = 0$ and ends when $i=4$



The image shows a Python script in a file named 'Exemplos.py' and its execution output in a command prompt window.

```
Exemplos.py > ...  
1  # Exemplos de estruturas repetitivas (iterativas) for  
2  
3  
4  for i in range (5):  
5      numero = int(input("Indique um número:"))  
6  
7  
8  
9  
10
```

The output window shows the program running five times, prompting the user to enter a number. The numbers entered are 10, 20, 30, 40, and 50.

```
C:\WINDOWS\py.exe  
Indique um número:10  
Indique um número:20  
Indique um número:30  
Indique um número:40  
Indique um número:50  
_
```

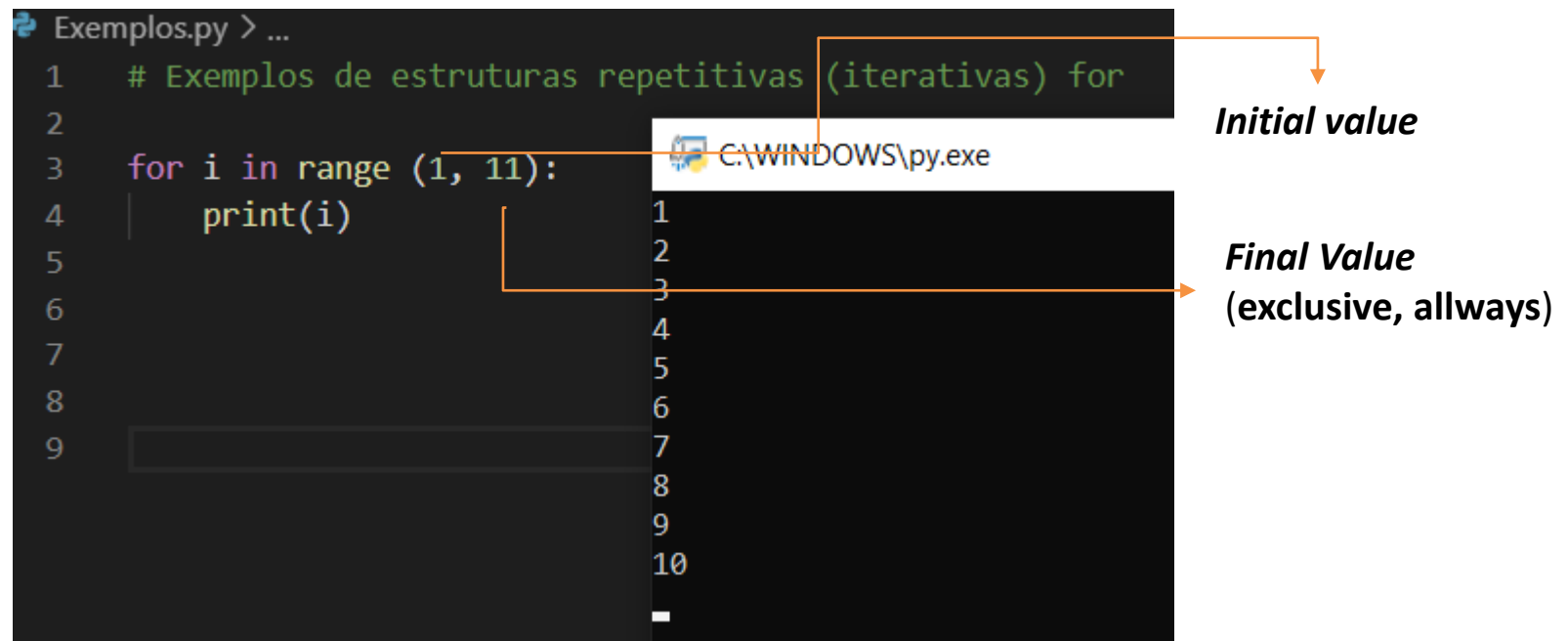
An orange arrow points from the text 'Repeat the cycle 5 times' to the '5' in the 'range(5)' function call in the code.

❖ Repetition Structures | Iterative *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

range: allows you to specify the number of times the cycle repeats

Repeat the cycle **10 times**. Starts with $i=1$ and ends in **10**



The screenshot shows a Python script named 'Exemplos.py' being executed. The script contains a `for` loop that iterates from 1 to 10, printing each value. The output of the script is displayed in a separate window, showing the numbers 1 through 10. Annotations with arrows point to the initial value (1) and the final value (10) in the range function, explaining that the final value is exclusive.

```
Exemplos.py > ...  
1  # Exemplos de estruturas repetitivas (iterativas) for  
2  
3  for i in range (1, 11):  
4      print(i)  
5  
6  
7  
8  
9
```

Initial value

Final Value
(exclusive, allways)

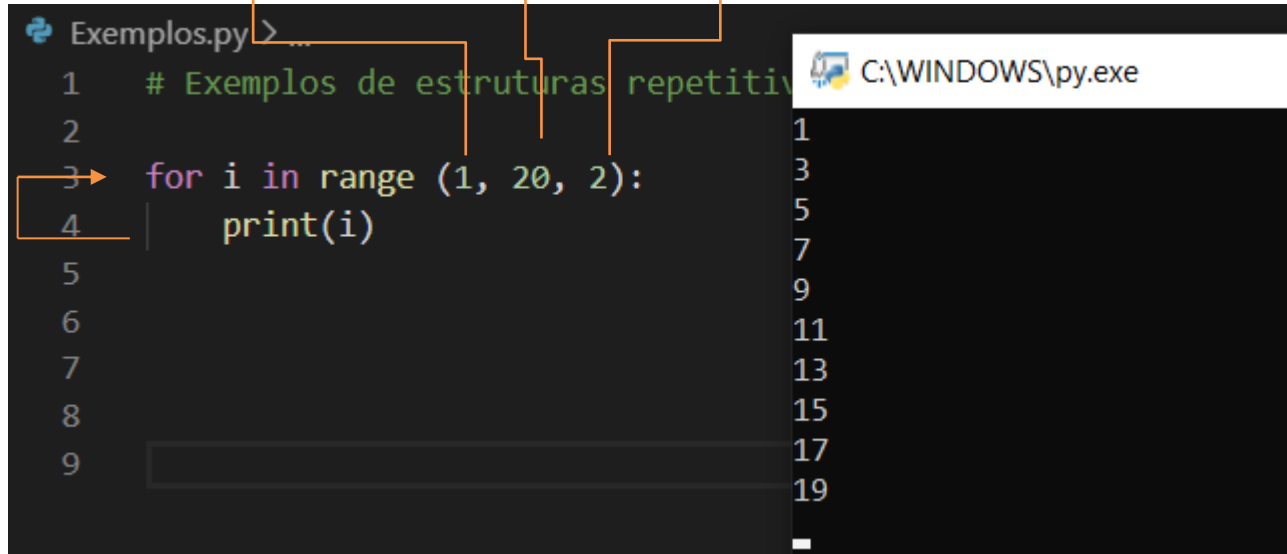
1
2
3
4
5
6
7
8
9
10
-

❖ Repetition Structures | Iterative *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

range: allows you to specify the number of times the cycle repeats

Initial value Final value (excluded) Step /increment from the variable **i** in each iteration



```
Exemplos.py > ...
1  # Exemplos de estruturas repetitivas
2
3  for i in range (1, 20, 2):
4      print(i)
5
6
7
8
9
```

The output of the program is displayed in a separate window titled 'C:\WINDOWS\py.exe':

```
1
3
5
7
9
11
13
15
17
19
```

The diagram illustrates the components of the `range` function in the code snippet `for i in range (1, 20, 2):`. Arrows point from the labels 'Initial value', 'Final value (excluded)', and 'Step /increment' to the values 1, 20, and 2 respectively in the code. A fourth arrow points from the variable `i` in the loop header to the text 'from the variable i in each iteration'.

❖ Repetition Structures | Iterative *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

range: allows you to specify the number of times the cycle repeats

Initial value Final value (excluded) Step /increment from the variable **i** in each iteration

```
Exemplos.py > ...  
1  # Exemplos de estruturas repeti  
2  
3  for i in range(0, 20, 2):  
4      print(i)  
5  
6  
7  
8  
9
```

C:\WINDOWS\py.exe
0
2
4
6
8
10
12
14
16
18

❖ Repetition Structures | Iterative *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

strings: we can iterate strings, as they consist of sequences of characters

```
Exemplos.py > ...  
1  # Exemplos de estruturas repetit  
2  
3  nome = "exemplo for"  
4  for caracter in nome:  
5      print(caracter)  
6  
7  
8  
9  
10
```

C:\WINDOWS\py.exe
e
x
e
m
p
l
o

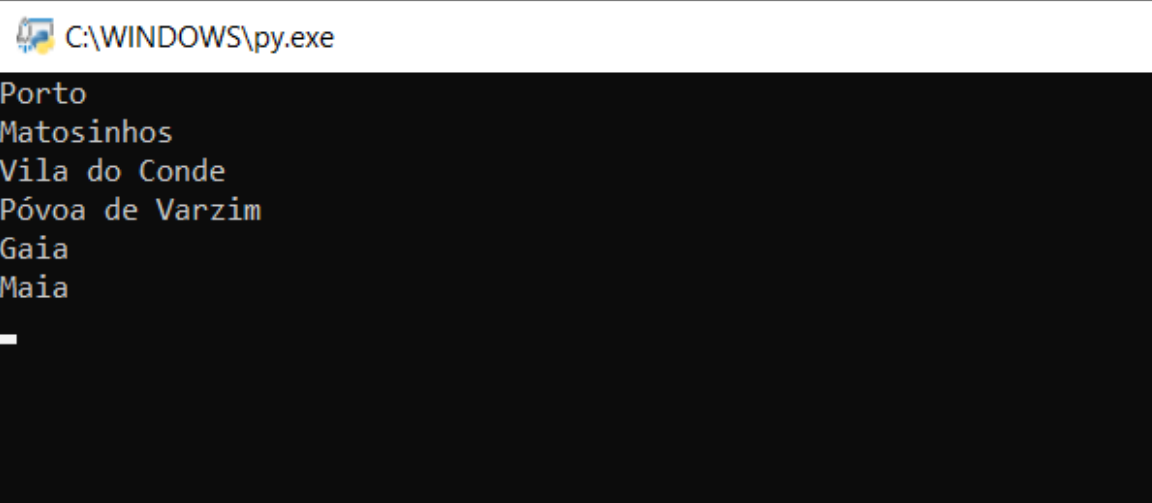
f
o
r

❖ Repetition Structures | Iterative *for*

Allows you to implement a cycle to execute a set of instructions repeatedly

Sequences: we can iterate structures that represent sequences of data

```
Exemplos.py > ...  
1  # Exemplos de estruturas repetitivas (iterativas) for  
2  
3  cidades = ["Porto", "Matosinhos", "Vila do Conde", "Póvoa de Varzim", "Gaia", "Maia"]  
4  for nome in cidades:  
5      print(nome)  
6  
7  
8  
9  
10
```



The screenshot shows a Python script being executed. The script defines a list of cities and iterates over it, printing each city name. The output of the script is displayed in a separate window titled 'C:\WINDOWS\py.exe'.

Output:

```
Porto  
Matosinhos  
Vila do Conde  
Póvoa de Varzim  
Gaia  
Maia  
-
```

Note: we will come back to these cases later when we use data structures of this type

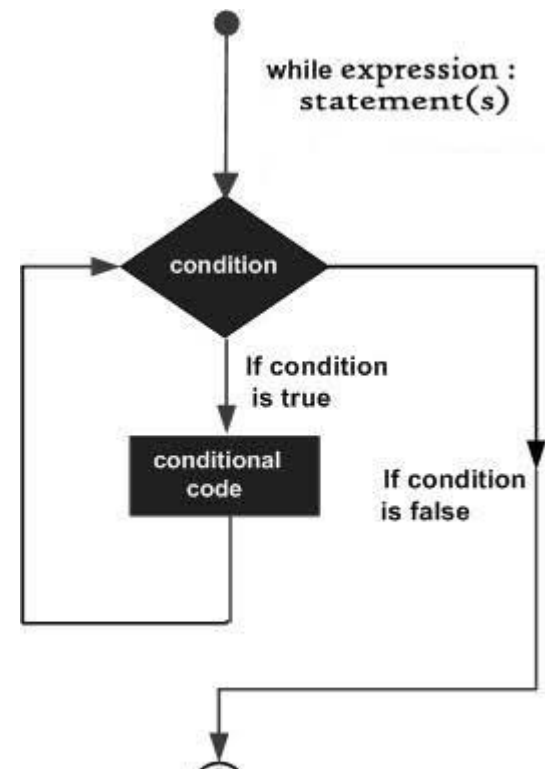
❖ Repetition Structures | Iterative *while*

Allows you to implement a cycle to execute a set of instructions repeatedly, but as long as a given condition is true

The condition is tested repeatedly before starting each iteration of the cycle

When the condition turns out to be false, execution continues to the line of code immediately following the end of the `while` cycle

```
while expression:  
    statement(s)
```



❖ Repetition Structures | Iterative *while*

Allows you to implement a cycle to execute a set of instructions repeatedly, but as long as a given condition is true

The condition is tested repeatedly before starting each iteration of the cycle

```
Exemplos.py > ...  
1  # Exemplos de estruturas repetitivas (iterativas) while  
2  
3  numero = 1  
4  while numero <10 :  
5      print(numero)  
6      numero+=1  
7  
8  
9  
10  
11  
12
```

Test again if
the condition
Is true / valid

C:\WINDOWS\py.exe

1
2
3
4
5
6
7
8
9

❖ Repetition Structures | Iterative *while*

Allows you to implement a cycle to execute a set of instructions repeatedly, but as long as a given condition is true

The condition is tested repeatedly before starting each iteration of the cycle

```
Exemplos.py > ...  
1  # Exemplos de estruturas repetitivas (iterativas) while  
2  
3  
4  numero = int(input("Indique uma valor entre 0 e 20:"))  
5  → while numero < 0 or numero > 20:  
6      print("o número inserido não é válido! \n")  
7      numero = int(input("Indique uma valor entre 0 e 20:"))  
8  
9  
10  
11  
12
```

C:\WINDOWS\py.exe

Indique uma valor entre 0 e 20:30
o número inserido não é válido!

Indique uma valor entre 0 e 20:-1
o número inserido não é válido!

Indique uma valor entre 0 e 20:18

❖ Repetition Structures | Iterative *while*

Allows you to implement a cycle to execute a set of instructions repeatedly, but as long as a given condition is true

The condition is tested repeatedly before starting each iteration of the cycle

```
Exemplos.py > ...
1  # Exemplos de estruturas repetitivas (iterativas) while
2
3
4  tabuada = int(input("Imprimir a tabuada dos: "))
5  numero = 1
6  while numero < 11 :
7      print(tabuada, "*", numero, "=", tabuada * numero )
8      numero+=1
9
10
11
12
13
```

C:\WINDOWS\py.exe

Imprimir a tabuada dos: 7

7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70

❖ Repetition Structures | Iterative *while*

Allows you to implement a cycle to execute a set of instructions repeatedly, but as long as a given condition is true

The condition is tested repeatedly before entering the cycle

```
Exemplos.py > ...
1  # Exemplos de estruturas repetitivas (iterativas) while
2  import random
3
4  numero = random.randint(0,10)
5  palpite = int(input("Indique o seu palpite:"))
6  while numero != palpite:
7      print("Não acertou, tente novamente :( \n")
8      palpite = int(input("Indique o seu palpite:"))
9
10 print("Parabéns, Acertou!!! :-)" )
11
12
13
14
15
16
```

C:\WINDOWS\py.exe

Indique o seu palpite:5
Não acertou, tente novamente :(

Indique o seu palpite:6
Não acertou, tente novamente :(

Indique o seu palpite:8
Não acertou, tente novamente :(

Indique o seu palpite:7
Não acertou, tente novamente :(

Indique o seu palpite:4
Parabéns, Acertou!!! :-)

❖ Repetition Structures | Iterative *while*

Allows you to implement a cycle to execute a set of instructions repeatedly, but as long as a given condition is true

The condition is tested repeatedly before entering the cycle

Exemplos.py > ...

```
1  # Exemplos de estruturas repetitivas (iterativas) while
2  import random
3
4  numero = random.randint(0,10) # gera numero aleatorio ente 0 e 10
5  palpite = int(input("Indique o seu palpite: "))
6  tentativas = 1                # para contar o nº de tentativas até acertar
7
8  while numero != palpite:
9      print("Não acertou, tente novamente :( \n")
10     palpite = int(input("Indique o seu palpite: "))
11     tentativas+=1
12
13 print("Parabéns, Acertou em {0} tentativas! :-)" .format(tentativas) )
14
15
16
17
18
19
```

C:\WINDOWS\py.exe

Indique o seu palpite: 5
Não acertou, tente novamente :(

Indique o seu palpite: 7
Não acertou, tente novamente :(

Indique o seu palpite: 3
Não acertou, tente novamente :(

Indique o seu palpite: 8
Parabéns, Acertou em 4 tentativas! :-)

❖ Repetition Structures | Iterative

Cycle breaks

Allows you to interrupt/break repetitive cycles

❑ break

Allows you to break the cycle and transfers the execution to the first instruction immediately after the cycle

❑ continues

Allows you to continue directly to the next iteration of a cycle without executing the following instructions of the current iteration

❖ Repetition Structures | Iterative

Cycle breaks

Allows you to interrupt/break repetitive cycles

break

Exemplo1.py > ...

```
1  # Exemplos de quebras de ciclos
2
3  tabuada = int(input("Imprimir a tabuada dos: "))
4  numero = 0
5  while numero < 10 :
6      numero+=1
7      if numero == 6:
8          break
9      print(tabuada, "*", numero, "=", tabuada * numero)
10
11
12
```

C:\WINDOWS\py.exe

```
Imprimir a tabuada dos: 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
```

❖ Repetition Structures | Iterative

Cycle breaks

Allows you to interrupt/break the current iteration

continues

```
Exemplo1.py > ...
1  # Exemplos de quebras de ciclos
2
3  tabuada = int(input("Imprimir a tabuada dos: "))
4  numero = 0
5  while numero < 10 :
6      numero+=1
7      if numero == 6:
8          continue
9      print(tabuada, "*", numero, "=", tabuada * numero)
10
11
12
```

```
C:\WINDOWS\py.exe
Imprimir a tabuada dos: 7
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```