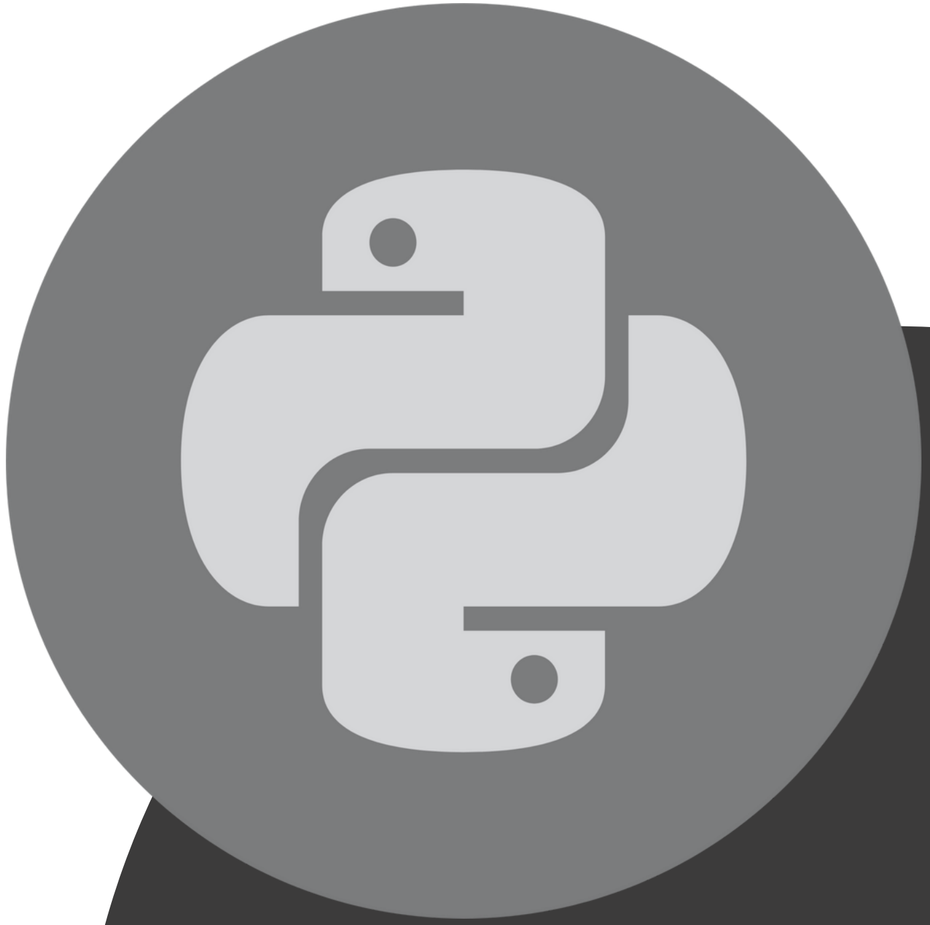
# PYTHON COURSE

## ENTRY LEVEL

**Basics of programming in Python 3.10**

This course will cover part of the arguments found in PCEP™ – Certified Entry-Level Python Programmer Certification



# OPERATIONS WITH LISTS

# OPERATIONS WITH LISTS

- Perform iterations on lists

Iterate directly **using elements**

```
my_list = ['Al', 'John', 'Jack']

for name in my_list:
    print(name) # prints 'Al', 'John', 'Jack' on three lines
```

Iterate **with positional indexing**

```
my_list = ['Al', 'John', 'Jack']

for i in range(len(my_list)):
    print(my_list[i]) # prints 'Al', 'John', 'Jack' on three lines
```

PYTHON

# OPERATIONS WITH LISTS

cast ranges into lists to create ordered numerical lists

```
my_range = range(10)

my_list = list(my_range)

for i in my_list:
    print(i, end=" ") # Prints 0 1 2 3 4 5 6 7 8 9
```

PYTHON

# OPERATIONS WITH LISTS

extract synthetical informations from lists

```
my_range = range(10)

my_list = list(my_range)

print(max(my_list)) # prints 9
print(min(my_list)) # prints 0
print(sum(my_list)) # prints 45
```

***min***(), ***max***() e ***sum***() are **built-in functions in python**

PYTHON

# OPERATIONS WITH LISTS

List slicing, as for Strings, you can slice lists the same ways

```
separation = 5

my_list = list(range(10))

left_part = my_list[:separation]
right_part = my_list[separation:]

print(left_part) # [0, 1, 2, 3, 4]
print(right_part) # [5, 6, 7, 8, 9]
```

PYTHON

# LISTS

create lists as a result of operations on elements

```
my_list = list(range(10))

new_list =[]

for num in my_list:
    new_list.append(num ** 2)

print(new_list)
```

use .append() method to dynamically insert new elements inside new_list

PYTHON

# LISTS

use the ***map***() built-in function to perform a function or operation on every element of the list

```
new_list = [0, 1, 2, 3]

types_list = list(map(type, new_list))

print(types_list)
```

the map function, performs the operation passed as first argument on every element in the list passed as second argument

PYTHON

# LISTS: LIST COMPREHENSION

a special Python syntax used to perform operations on lists with less code lines

**standard** syntax

```
my_list = list(range(10))

new_list =[]

for num in my_list:
    new_list.append(num ** 2)

print(new_list)
```

with **list comprehension**

```
my_list = list(range(10))

new_list = [num ** 2 for num in my_list]

print(new_list)
```

PYTHON

# LISTS: CLONING AND COPYING

**List are mutable objects**, so you need to pay attention when performing operations.
**Copying a list will copy the same reference**, so **modifying the copy, will modify the original object also**

In this code, the list **my_list is copied**, so **removing the element from the copy will remove it from the original list also**

```
my_list = list(range(5))

print(my_list)

new_list = my_list
new_list.pop()

print(my_list)
```

PYTHON

# LISTS

To avoid potential problem, you can **clone the list using slicing**, this will create a new different object and operations on it will not modify original object

```
my_list = list(range(5))

print(my_list)

new_list = my_list[:]
new_list.pop()

print(my_list)
```

in this code, the original object is not mutated, cause the object has been cloned as the result of a slicing operation

# NESTED LISTS: MATRIX

- You can **nest multiple lists** to **create matrices**

```
list1 = [1, 2, 3]
list2 = [0, 2, 0]
list3 = [0, 0, 0]

matrix = [list1, list2, list3]

print(matrix) # [[1, 2, 3], [0, 2, 0], [0, 0, 0]]
```

PYTHON

# NESTED LISTS: MATRIX

you can iterate through matrices' elements using loops

```
list1 = [1, 2, 3]
list2 = [0, 2, 0]
list3 = [0, 0, 0]

matrix = [list1, list2, list3]

for i in range(len(matrix)):
    for j in range(len(matrix[i])):
        print(f"element at row {i}, column {j} is {matrix[i][j]}")
```

nesting for loops, you can access single elements in a matrix

PYTHON

# NESTED LISTS: TENSORS

You can also create Tensors by nesting lists (lists of matrices, also known as cubes)

```
matrix1 = [[1, 2, 3], [1, 0, 5], [0, 0, 0]]
matrix2 = [[4, 5, 6], [0, 0, 3], [1, 2, 0]]
matrix3 = [[1, 5, 3], [0, 2, 0], [0, 2, 3]]

tensor = [matrix1, matrix2, matrix3]
```

PYTHON

# NESTED LISTS: TENSORS

you can iterate through Tensors' elements using triple nested loops (one for every dimension)

```
matrice1 = [[0, 1, 2], [3, 4, 5], [6, 7, 8]]
matrice2 = [[9, 10, 11], [12, 13, 14], [15, 16, 17]]
matrice3 = [[18, 19, 20], [21, 22, 23], [24, 25, 26]]


tensore = [matrice1, matrice2, matrice3]

for i in range(len(tensore)):
    for j in range(len(tensore[i])):
        for k in range(len(tensore[i][j])):
            print(f"elemento al matrice {i}, row {j}, colonna{k}, is {tensore[i][j][k]}")
```

So, **when nested**, loops **resolves from the most inner to the outer one**
(The outer for iteration is updated when the inner for iterations are completed)

# EXERCISES