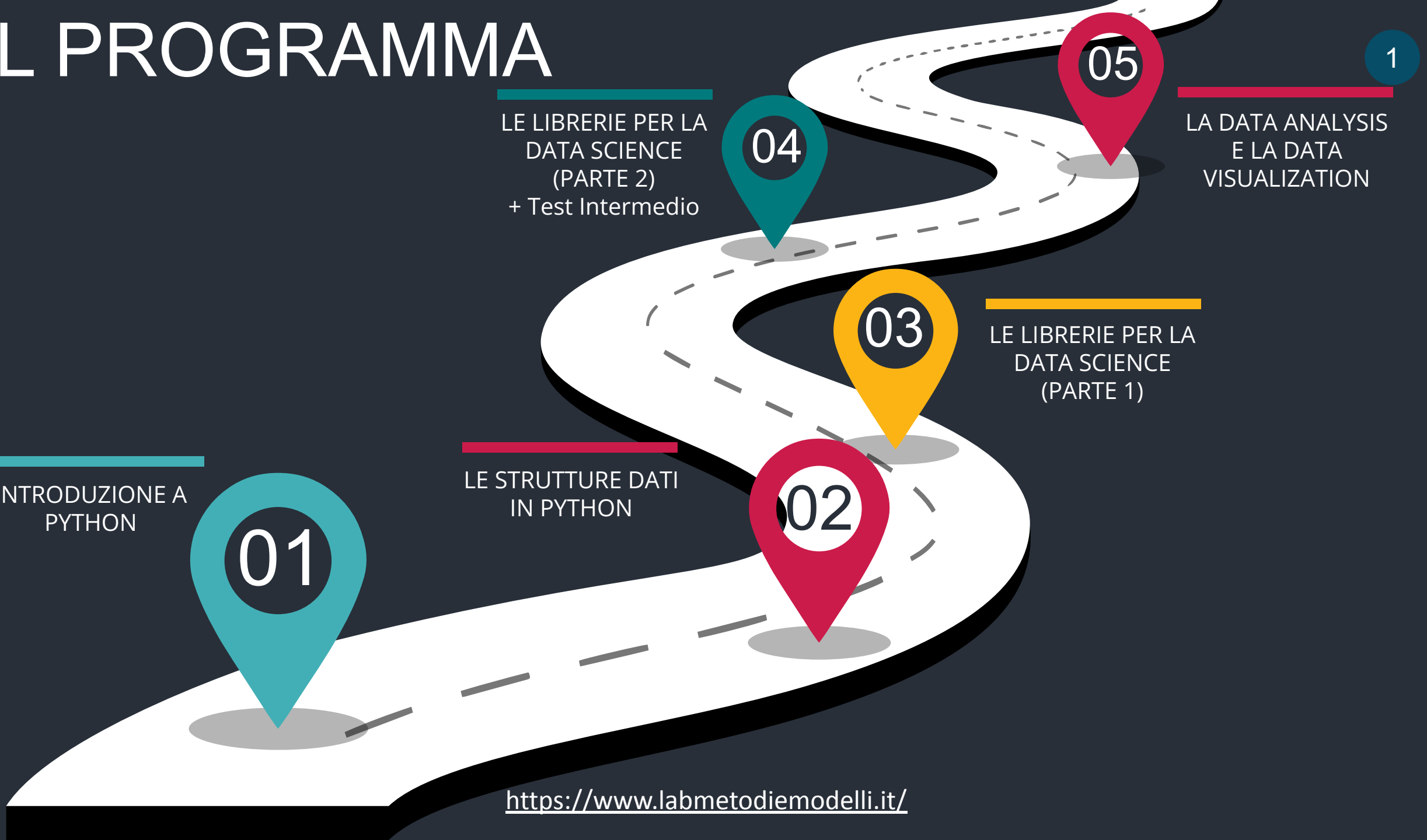
LABORATORIO DI:

METODI E MODELLI MATEMATICI IN PYTHON

A CURA DI: ANTONIO MIRARCHI & GIUSEPPE TROTTA

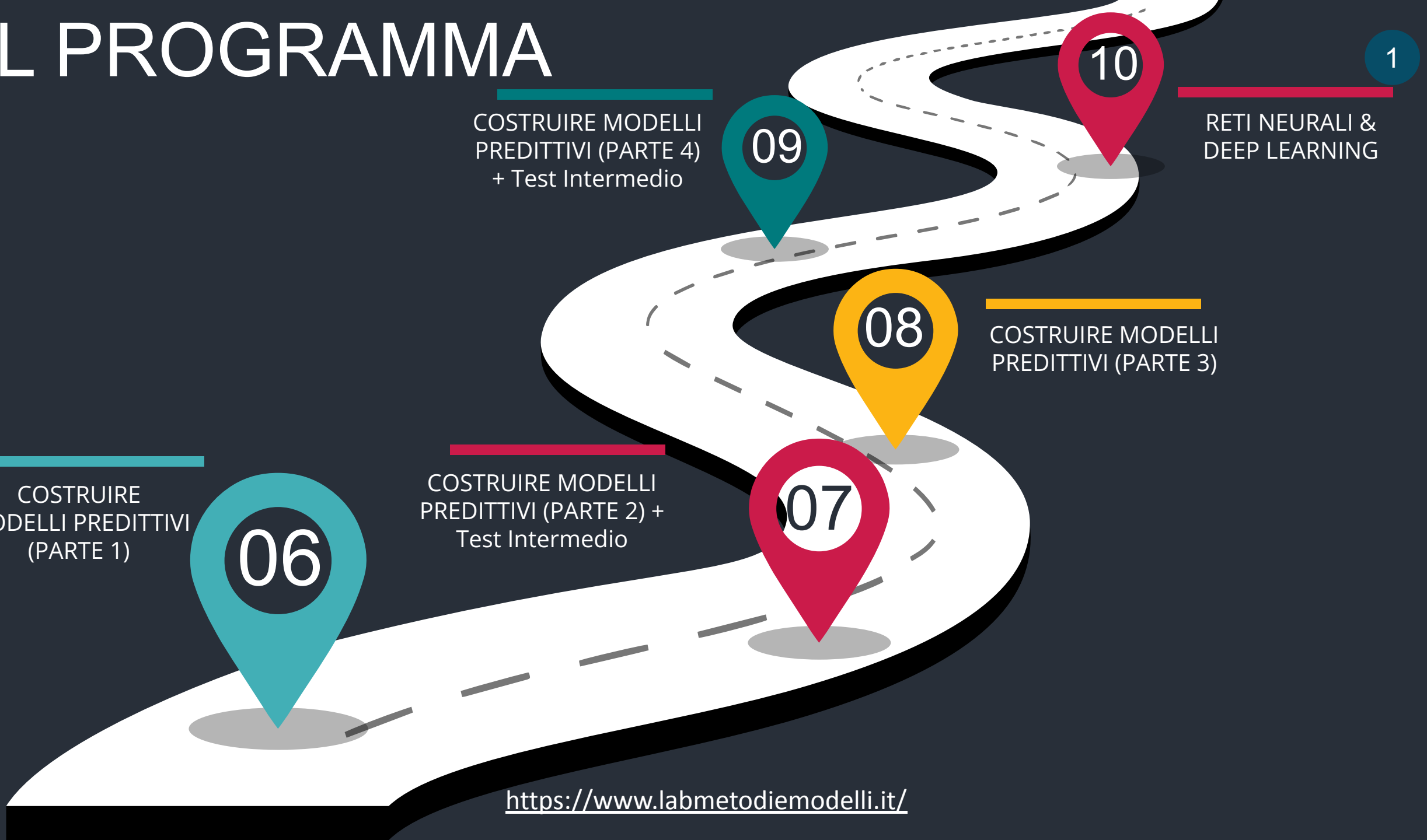
https://www.labmetodiemodelli.it/

IL PROGRAMMA



https://www.labmetodiemodelli.it/

IL PROGRAMMA



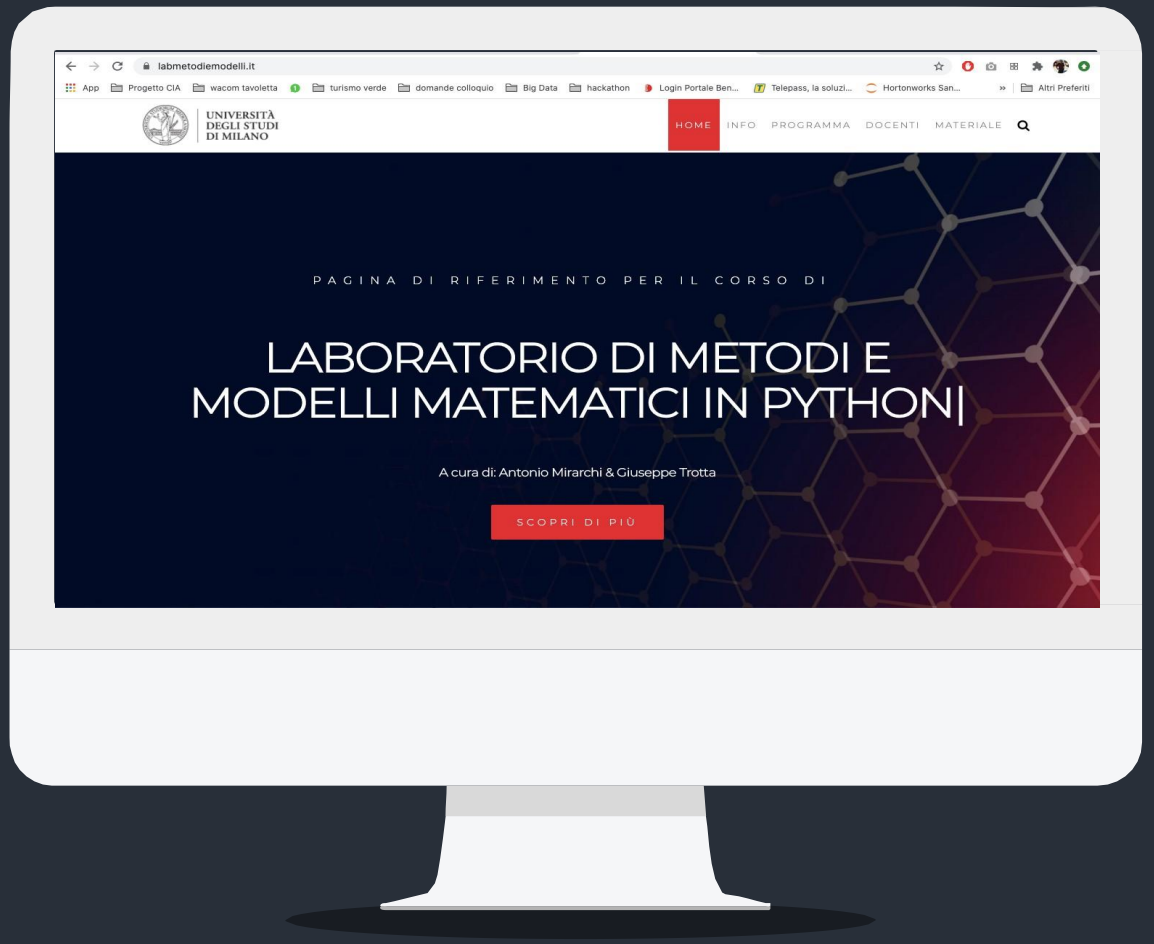
https://www.labmetodiemodelli.it/

Let’s Code!

https://www.labmetodiemodelli.it/

La Password per decomprimere I notebook presenti sul sito é:

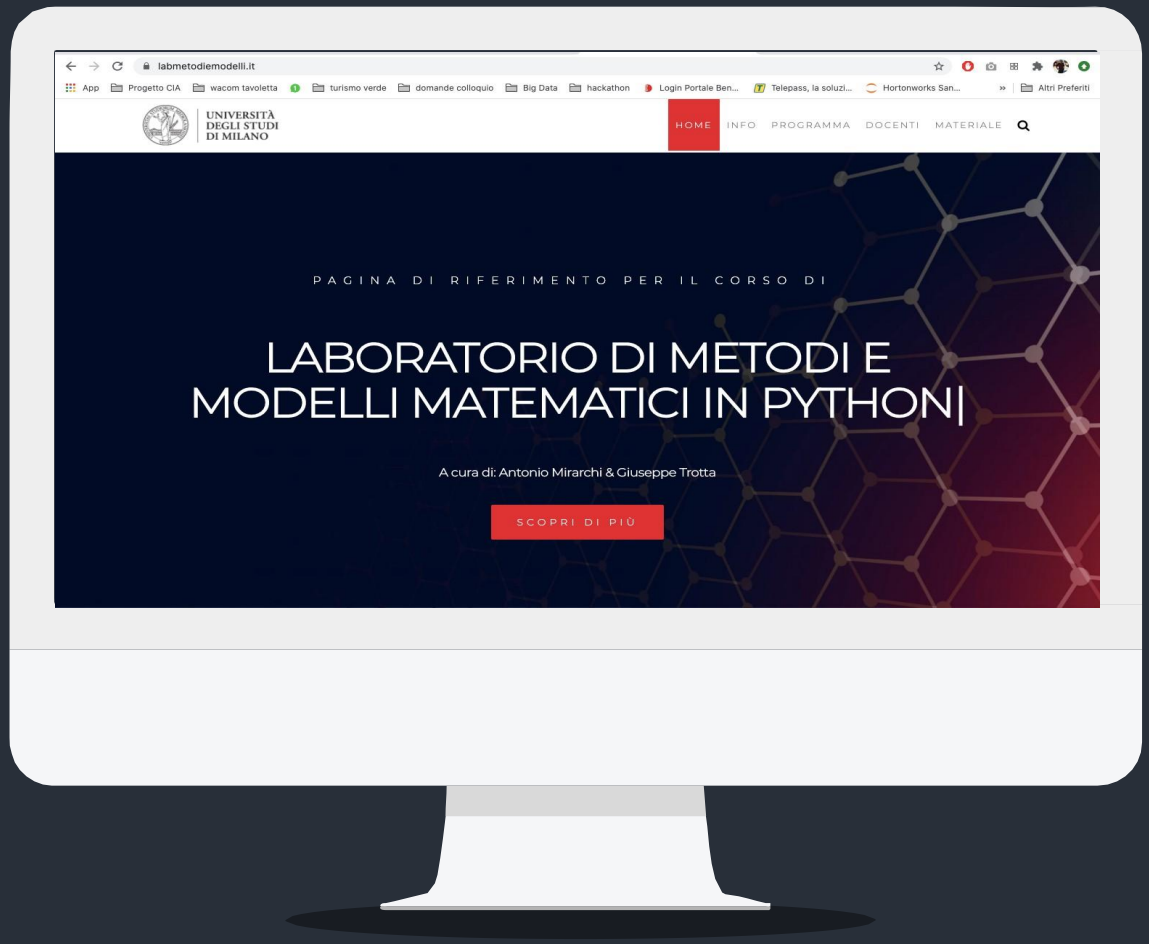
q@rant4!



https://www.labmetodiemodelli.it/

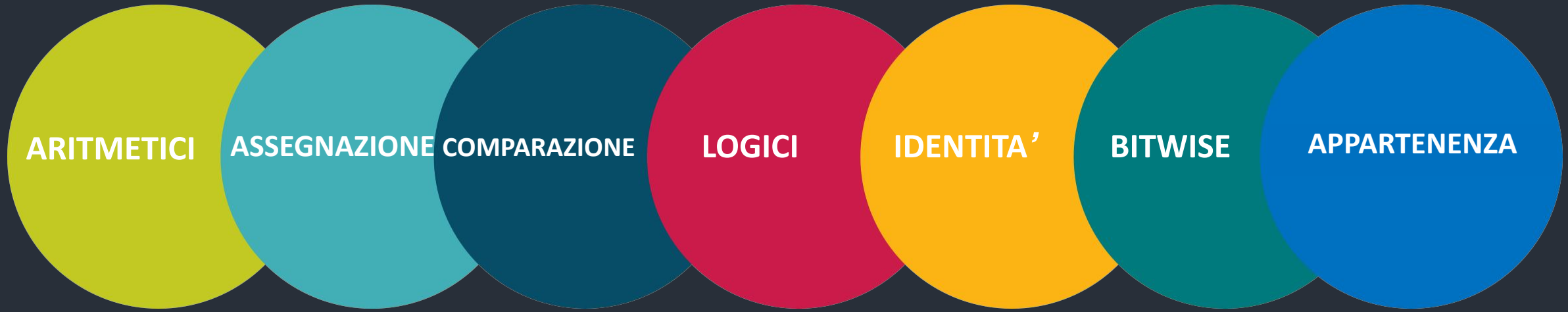
LEZIONE 2 – LE STRUTTURE DATI IN PYTHON E LE FUNZIONI

1. Gli Operatori
2. Flow Control (Conditional Statement)
3. Le Strutture Dati
4. Definizione di funzioni
5. ricorsione
6. gestione delle eccezioni



https://www.labmetodiemodelli.it/

1. Gli Operatori



1. Gli Operatori

ARITMETICI

Operatore	Descrizione	Esempio
+	Addizione	3 + 2 = 5
-	Sottrazione	10 - 4 = 6
*	moltiplicazione	4 * 3 = 12
/	divisione	20 / 2 = 10
%	Modulo (resto della divisione)	21 % 2 = 1
**	Potenza	3**2 = 9
//	Floor (taglia la parte dopo la virgola)	10.5//2 =5.0

1. Gli Operatori

ASSEGNAZIONE

Operatore	Descrizione	Esempio
=	assegnazione	X = 10 X= X + 5 15
+=	Incremento di	X=15 X += 5 20
-=	Decremento di	X=20 X -= 5 15
*=	Prodotto ed assegnazione	X = 15 X *= 3 45

1. Gli Operatori : x = 10 y = 5

COMPARAZIONE

Operatore	Descrizione	Esempio
>	maggiore	x > 11 False
<	minore	x < 20 True
==	uguale	x == y False
>=	maggiore o uguale	y >= 5 True
<=	minore o uguale	x <= 10 True
!=	diverso	x != y True

1. Gli Operatori: x=10

LOGICI

Operatore	Descrizione	Esempio
and	ritorna il valore True se tutte le condizioni indicate sono vere.	x > 9 and x < 11 True.
or	ritorna il valore True se almeno una delle condizioni indicate è vera.	x > 5 or x < 9 True.
not	inverte il risultato atteso: se fosse True diventerebbe False e viceversa.	not x > 5 False.

1. Gli Operatori:

Gli operatori di identità vengono usati per verificare se due operandi sono uguali(se si riferiscono allo stesso oggetto), ovvero se puntano alla stessa locazione di memoria.

Nota bene: il fatto che i due operandi abbiano lo stesso valore non implica che essi siano uguali.

IDENTITA'

Operatore	Descrizione	Esempio
is	ritorna il valore True se i due operandi sono uguali.	x is True
is not	ritorna il valore True se i due operandi non sono uguali(non si riferiscono allo stesso oggetto).	x is not False

1. Gli Operatori: x=4 (0000 0100) y = 2 (0000 0010)

BITWISE

Operatore	Descrizione	Esempio
&	assegna 1 se entrambi sono 1	x & y = 0 (0000 0000)
|	assegna 1 se almeno 1 è 1	x | y = 6 (0000 0110)
^	xor binario assegna 1 se solo 1 dei bit a confronto è 1	x ^ y = 6 (0000 0110)
~	Bitwise not inverte i valori binari	~x = -5 (1111 1011)
<<	sposta i bit a sinistra in base al numero indicato	x << 3 = 32 (0010 0000)
>>	sposta i bit a destra in base al numero indicato	x >> 2 = 1 (0000 0001)

1. Gli Operatori: x = "ciao"

APPARTENENZA

Operatore	Descrizione	Esempio
in	ritorna il valore True se l'operando di sinistra è presente nella sequenza indicata	"a" in x True.
not in	ritorna il valore True se l'operando di sinistra non è presente nella sequenza indicata	"o" not in x False.

Basic Data Types

1. Numbers
2. Booleans
3. Strings

1. Numbers

Integers and floats work as you would expect from other languages
Note that unlike many languages, Python does not have unary increment (x++) or decrement (x--) operators.

```
x = 3
print(type(x)) # Prints "<class 'int'>"
print(x) # Prints "3"
print(x + 1) # Addition; prints "4"
print(x - 1) # Subtraction; prints "2"
print(x * 2) # Multiplication; prints "6"
print(x ** 2) # Exponentiation; prints "9"
x += 1 print(x) # Prints "4"
x *= 2 print(x) # Prints "8"
y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # Prints "2.5 3.5 5.0 6.25"
```

2. Booleans

Python implements all of the usual operators for Boolean logic, but uses English words rather than symbols (&&, ||, etc.):

```
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"
print(t or f) # Logical OR; prints "True"
print(not t) # Logical NOT; prints "False"
print(t != f) # Logical XOR; prints "True"
```

3. Strings

Python has great support for strings:
String objects have a bunch of useful methods;

```
hello = 'hello' # String literals can use single quotes
world = "world" # or double quotes; it does not matter.
print(hello) # Prints "hello"
print(len(hello)) # String length; prints "5”
hw = hello + ' ' + world # String concatenation
print(hw) # prints "hello world"
hw12 = '%s %s %d' % (hello, world, 12) # printf style string formatting
print(hw12) # prints "hello world 12"
```

https://docs.python.org/3.5/library/stdtypes.html#string-methods

Let’s Code!

https://www.labmetodiemodelli.it/

Syntax

1. If/else Statement
2. For Loops
3. While Loops
4. Function

https://www.labmetodiemodelli.it/

1. If/else Statement

The if/else statement executes a block of code if a specified condition is true. If condition is not met, another block of code can be executed

```
var = 10

if (var >= 5):
    print('valore maggiore o uguale a 5')
elif (var < 0):
    print('valore negativo')
elif (var == 0):
    print('valore uguale a zero')
else:
    print('valore minore di 5')
```

2. For loops

Loops through a block of code a number of times

```
# One parameter
for i in range(10):
    print(i)
```

3. While loops

Loops through a block of code while a specified condition is met

```
var = 10

while(var <= 20):
        print('var minore o uguale a 20')
        var+=2
```

4. Function

A function is a block of code designed to perform a particular task.

```
def somma_due_numeri(a,b):
    somma = a+b
    return somma

somma_due_numeri(4,4)
```

4. Function – Esercizio 1

Using function somma_due_numeri, create a function somma_tre_numeri that sums three input numbers and returns the result.

#TestCase
Somma_tre_numeri(2,4,12)
18

4. Function – Esercizio 1 – Soluzione


```
#TestCase
Somma_tre_numeri(2,4,12)
18
```

```
def somma_tre_numeri(a,b,c):
    d = somma_due_numeri(a,b)
    somma_total = somma_due_numeri(d,c)
    return somma_total

somma_tre_numeri(2,4,12)
```

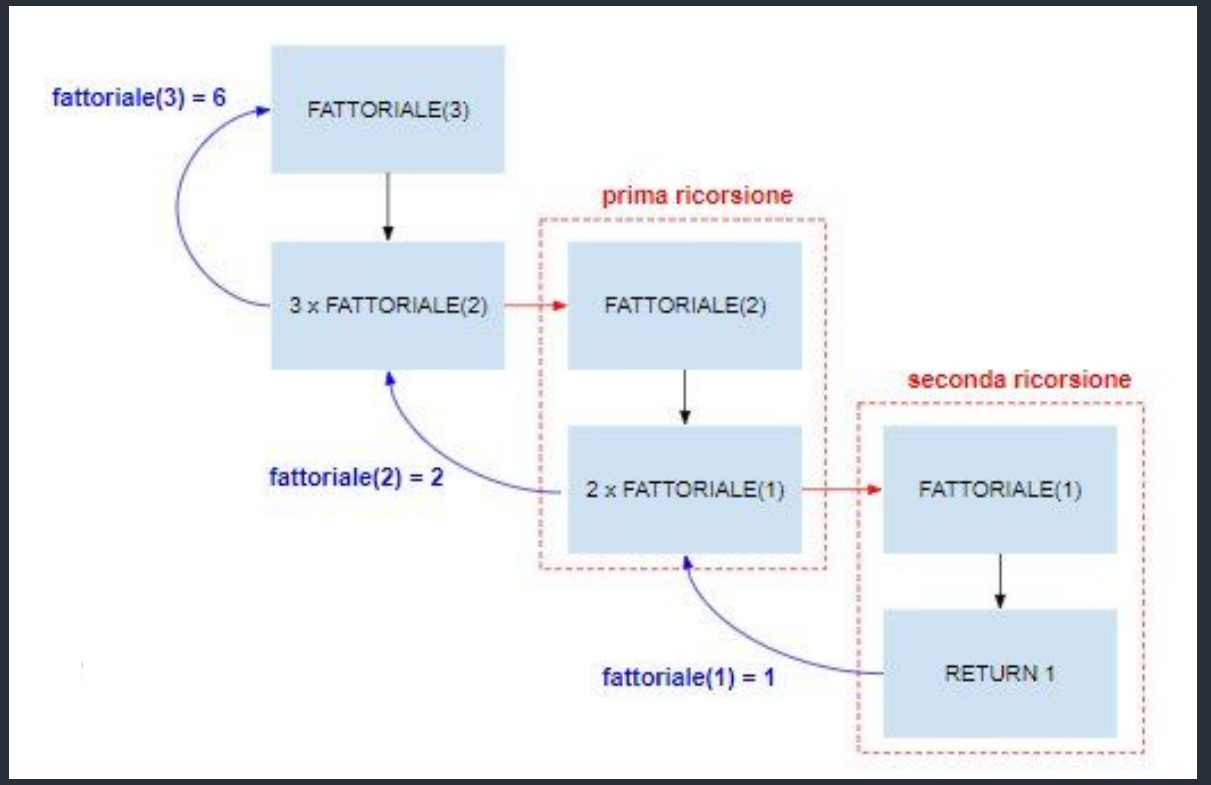
```
def somma_tre_numeri(a,b,c):
 return
somma_due_numeri(somma_due_numeri
(a,b),c)

somma_tre_numeri(2,4,12)
```

Let’s Code!

https://www.labmetodiemodelli.it/

5. La ricorsione

la ricorsione è una funzione che chiama se stessa, il cui scopo è quello di semplificare l'esecuzione di funzioni ripetitive.



https://www.labmetodiemodelli.it/

4. Function – Esercizio 2

Scrivete in python la definizione di
4 funzioni che eseguano le 4
operazioni aritmetiche principali

Let’s Code!

https://www.labmetodiemodelli.it/

6. La gestione delle eccezioni

Cos'è un'**eccezione**? Si tratta di un evento imprevisto che impedisce o altera l'esecuzione del programma. Ad esempio, la divisione per zero di un valore numerico, ecc. Le eccezioni servono a gestire il verificarsi di questi eventi anomali.

```
In [ ]: # proviamo a sommare un numero e una stringa

In [1]: 37 + 'stringa'

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-1bb880883738> in <module>()
----> 1 37 + 'stringa'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Let’s Code!

https://www.labmetodiemodelli.it/

Prossimi Appuntamenti

17 MAR	Le librerie per la data science
24 MAR	Le librerie per la data science parte 2 + 1° Test Intermedio
31 MAR	La Data Analysis & Data Visualization

https://www.labmetodiemodelli.it/