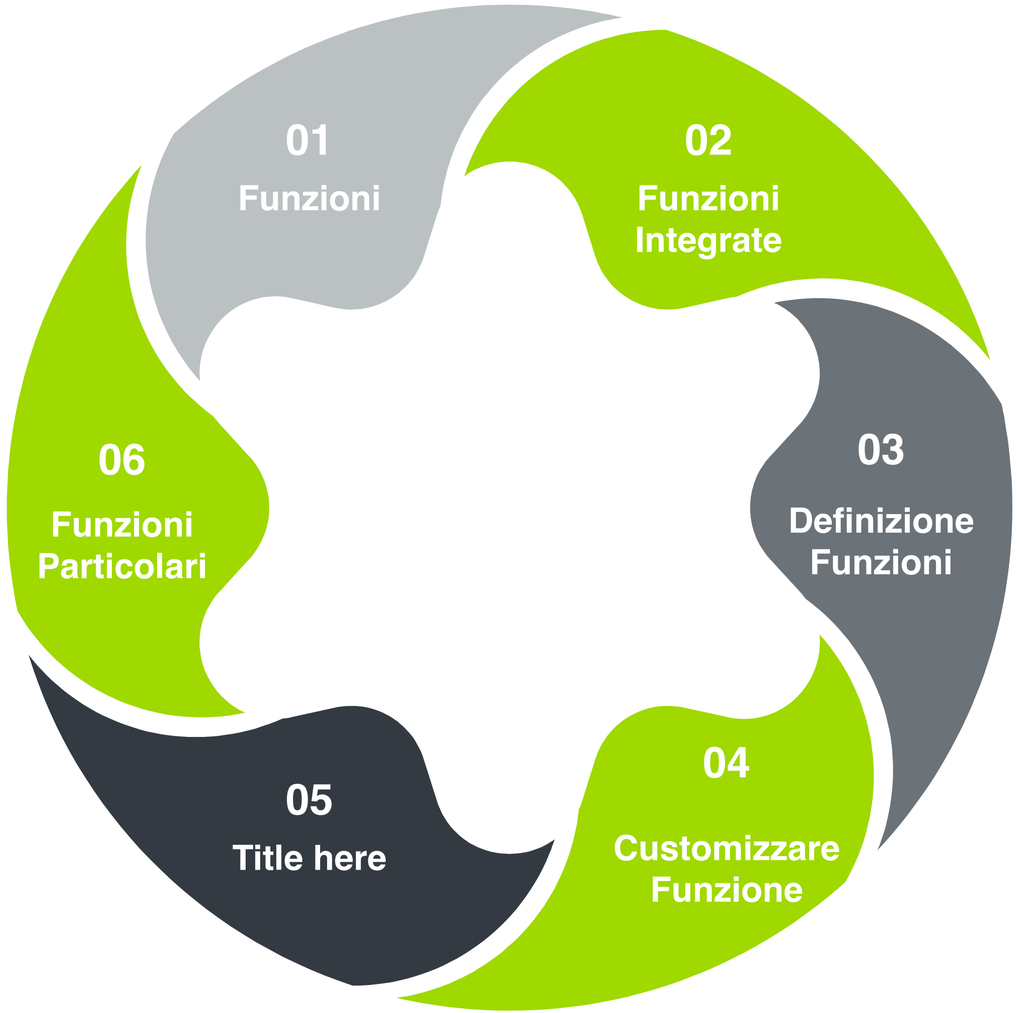
FUNZIONI

Indice



NANOCORSI

Liste in Python

Le **liste** in Python sono una serie ordinata di valori identificati da un indice.

Una **lista** è un dato composto che aggrega dei valori di qualsiasi tipo, racchiusi tra una coppia di parentesi quadre e separati da una virgola.

I valori che fanno parte di una lista si chiamano **elementi** e possono essere numeri o stringhe.

```
votazione=[18,20,25,28,'30 e lode']

print(votazione)
```

Stampare elementi di una lista

Ad ogni elemento della lista è assegnato un indice che parte da sinistra e inizia da 0. Dunque, per stampare ad esempio il primo elemento scriviamo:

print**(votazione[0])**

Mentre per stampare l'ultimo elemento scriviamo:

print**(votazione[4])**

E' possibile visualizzare delle sottoliste es: solo i primi due elementi della lista.

print**(votazione[:3])**

print(votazione)

Modificare elementi in una lista

Gli elementi della lista possono anche essere modificati, riassegnandogli un nuovo valore.

Ad esempio assegniamo al primo elemento il valore '*Estate*', mentre al secondo assegniamo il valore '*Primavera*'.

votazione[4]='30 senza lode'

Concatenare liste,

concatenare due liste
Per concatenare due liste si usa l'operatore **+**, q

```
votazione_1_corso=[18,20,25,28,'30 e lode']
votazione_2_corso = [18,22,28,'30']
votazioni=votazione_1_corso+votazione_2_corso

print(votazioni)
```

Ricercare un dato in una lista

```
votazioni=[18,20,25,28,'30 e lode']


'25' in votazioni #restituisce True

'29' in votazioni #restituisce False


print(votazioni)
```

Funzioni in Python

Esistono due tipi di funzioni in Python

- **Built in Functions**
  Funzioni fornite con il linguaggio di programmazione
  raw_input(), type(), float(), max(), min(), int(), str(), …

- **Funzioni definite dall'utente**
  Sono funzioni definiti dall'utente che possono essere definiti e riutilizzati.

NANOCORSI

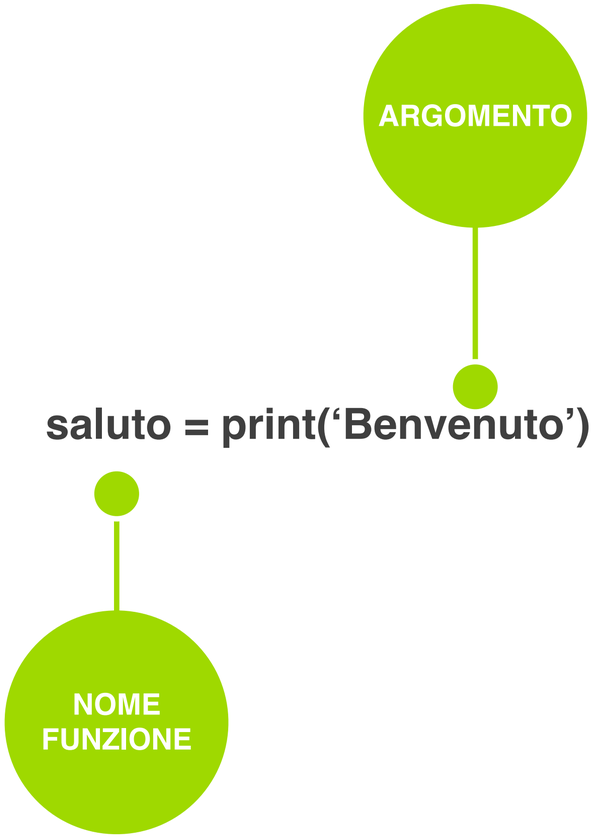
Built in Functions

Built-in Functions

A	**E**	**L**	**R**
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
any()			round()
anext()	**F**	**M**	
ascii()	filter()	map()	**S**
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	**G**	**N**	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	**H**	object()	
callable()	hasattr()	oct()	**T**
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		**P**	**V**
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	**Z**
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		_
	iter()		__import__()

https://docs.python.org/3/library/functions.html

Anatomia di una funzione



Definizione di una funzione

Definizione di una funzione

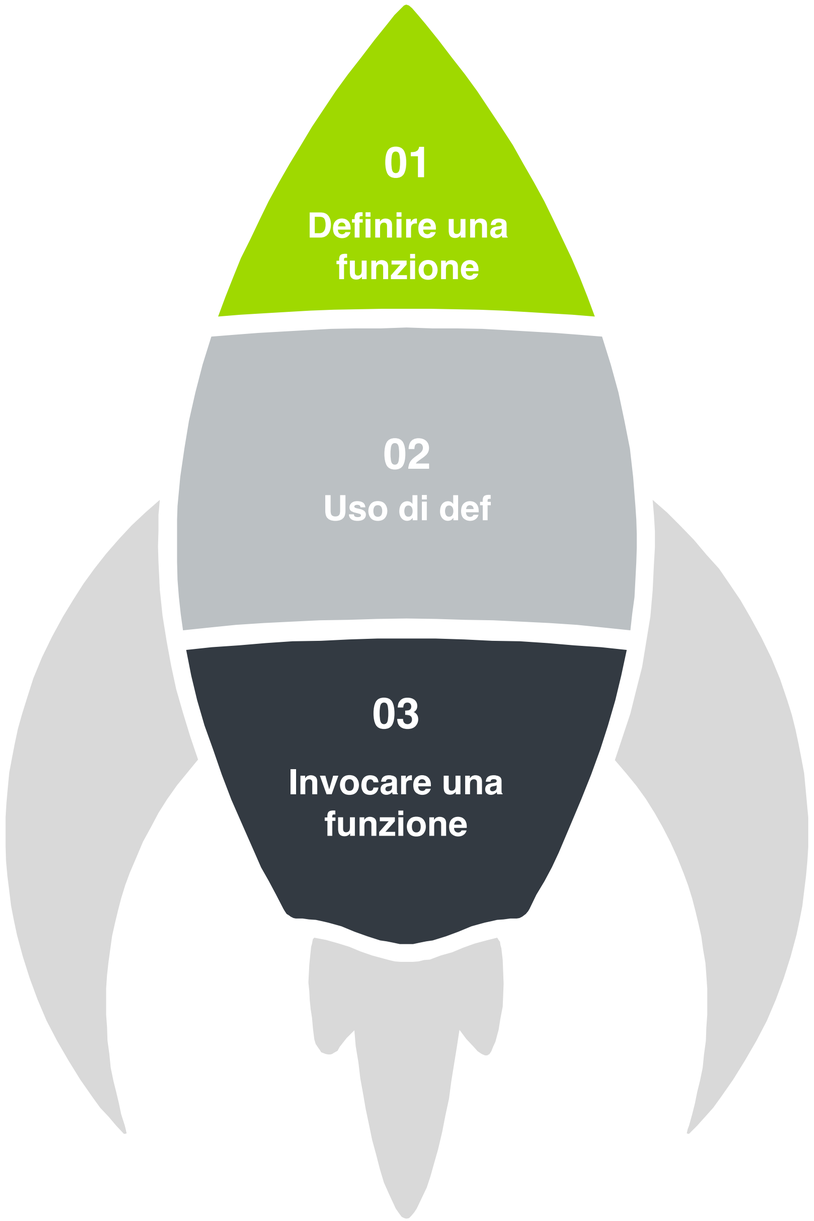
In Python una funzione è un porzione di codice riutilizzabile che può: accettare argomenti in input, eseguire calcoli e quindi restituisce un risultato.

Uso di DEF

Per definire una funzione utilizziamo la parola chiave **def**, cseguita dal nome che vogliamo dare

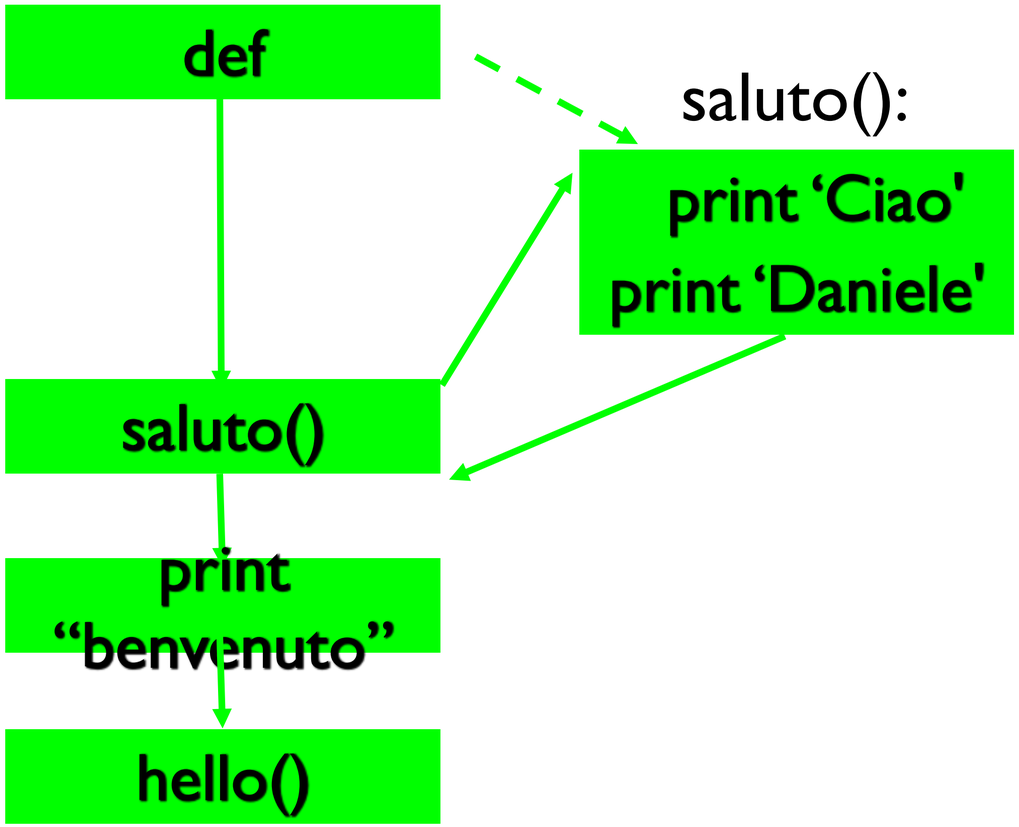
Invocare una funzione

Invochiamo la funzione utilizzando il nome della funzione, le parentesi con gli argomenti



NANOCORSI

Stored (and reused) Steps



Program:

```
// Definisco la funzine
def saluto():
    print("Ciao, benvenuto nel nostro software")


saluto()
```

snappify.io

We call these reusable pieces of code “functions”.

Costruire una nostra funzione

Creiamo una nuova funzione usando la parola riservata **def** seguita dal nome della funzione, parametri opzionali tra parentesi, quindi aggiungiamo i due punti.

Rientriamo nel corpo della funzione

Questo definisce la funzione ma non esegue il corpo della funzione

```
def inserisci_saluti():
    print “Cordiali Saluti"
```

Definizioni ed uso

Definita una funzione possiamo invocarla tutte le volte che vogliamo

Questo permette il riuso del software

Argomento

Un argomento è un valore che passiamo alla funzione come input quando chiamiamo la funzione

Usiamo argomenti in modo da poter indirizzare la funzione a fare diversi tipi di lavoro

Poniamo gli argomenti tra parentesi dopo il nome della funzione

```
lista=[18, 22, 18, 26, 30, 19]
print(max(lista))
```

Argomento

Parametri

Un parametro è una variabile che utilizziamo nella definizione della funzione che è un "handle" che consente al codice nella funzione di accedere agli argomenti per una particolare chiamata di funzione.

```
def greet(lang):
        if lang == 'es':
                print 'Hola'
        elif lang == 'fr':
                print 'Bonjour'
        else:
                print 'Hello‘

greet('en')
Hello
greet('es')
Hola
greet('fr')
Bonjour
```

Return Values

Spesso una funzione prende i suoi argomenti, esegue dei calcoli e restituisce un valore da utilizzare come valore della chiamata di funzione nell'espressione chiamante.

La parola chiave return viene utilizzata per questo.

```
def greet():
    return "Hello "

print greet(), "Glenn"
print greet(), "Sally"
```

```
Hello Glenn
Hello Sally
```

Esercizio 1

- *Create una lista di oggetti*
- *Stampare la lista*
- *stampare il secondo elemento della lista*
- *Sostituire il 3 valore della lista*
- *Stampare la nuova lista*
- *Stampare i primi 3 elementi della lista*
- *Rimuovere il 2 elemento della lista*
- *Contare quante volte un elemento è presente nella lista*

Esercizio II

- *Utilizzare e documentare tre funzioni built in all'interno di un software e commentarne il suo utilizzo.*

Esercizio 3

- *Definire 4 funzioni per le operazioni matematiche: addizione, sottrazione, divisione, moltiplicazione*
- *Chiedere all'utente un primo numero*
- *Chiedere all'utente un secondo numero*
- *Chiedere all'utente l'operazione da effettuare*
- *Stampare a video il risultato*