RIFERIMENTO RAPIDO PYTHON 3

Basi, pandas, requests, csv, json

Basi

Variabili

```
name = "Alice"   # str
age = 20         # int
gpa = 3.85       # float
active = True    # bool
```

Tipi di Dato

`str`	Testo: `"hello"`
`int`	Intero: `42`
`float`	Decimale: `3.14`
`bool`	`True` / `False`
`list`	Ordinato, mutabile: `[1, 2, 3]`
`tuple`	Ordinato, immutabile: `(1, 2)`
`dict`	Chiave-valore: `{"a": 1}`
`set`	Elementi unici: `{1, 2, 3}`

Aritmetica

`+ - *`	Addizione, sottrazione, moltiplicazione
`/`	Divisione (float): 7/2 → 3.5
`//`	Divisione intera: 7//2 → 3
`%`	Modulo: 7%2 → 1
`**`	Potenza: 2**3 → 8

Conversione di Tipo

```
int("42")     # 42
float("3.14") # 3.14
str(100)      # "100"
list("abc")   # ['a','b','c']
```

Input Utente

```
name = input("Your name? ")
age = int(input("Age? "))
```

Stringhe

Creare Stringhe

```
s1 = 'single quotes'
s2 = "double quotes"
s3 = """triple quotes
    for multiline"""
```

f-String (Python 3.6+)

```
name = "Alice"
f"Hello, {name}!"  # Hello, Alice!
f"{2 + 3}"         # 5
f"{3.14159:.2f}"   # 3.14
f"{1000:,}"        # 1,000
```

Slicing di Stringhe

```
s = "Python"
# Index: 0 1 2 3 4 5
s[0]   # 'P'
s[-1]  # 'n'
s[2:5] # 'tho'
s[:2]  # 'Py'
s[2:]  # 'thon'
s[::-1] # 'nohtyP' (reverse)
```

Metodi delle Stringhe

`len(s)`	Lunghezza della stringa
`s.upper()`	MAIUSCOLO
`s.lower()`	minuscolo
`s.strip()`	Rimuove spazi iniziali/finali
`s.split(",")`	Divide in lista
`",".join(lst)`	Unisce lista in stringa
`s.replace(a, b)`	Sostituisce a con b
`s.find("x")`	Indice della prima occorrenza (-1 se assente)
`s.startswith(x)`	Controlla prefisso → bool
`s.endswith(x)`	Controlla suffisso → bool
`s.count(x)`	Conta le occorrenze
`"x" in s`	Verifica presenza → bool

Liste

Creare e Accedere

```
fruits = ["apple", "banana", "cherry"]
fruits[0]   # "apple"
fruits[-1]  # "cherry"
fruits[1:3] # ["banana", "cherry"]
```

List Comprehension

```
squares = [x**2 for x in range(5)]
# [0, 1, 4, 9, 16]
evens = [x for x in range(10) if x%2==0]
# [0, 2, 4, 6, 8]
```

Metodi delle Liste

`lst.append(x)`	Aggiunge in fondo
`lst.extend(lst2)`	Aggiunge tutti gli elementi di lst2
`lst.insert(i, x)`	Inserisce all'indice i
`lst.pop()`	Rimuove e restituisce l'ultimo
`lst.pop(i)`	Rimuove e restituisce all'indice i
`lst.remove(x)`	Rimuove la prima occorrenza di x
`del lst[i]`	Elimina per indice
`lst.sort()`	Ordina in-place
`sorted(lst)`	Restituisce copia ordinata
`lst.reverse()`	Inverte in-place
`len(lst)`	Numero di elementi
`x in lst`	Verifica appartenenza
`lst.index(x)`	Primo indice di x
`lst.count(x)`	Conteggio di x

Tuple e Set

Tuple (Immutabili)

```
point = (3, 4)
x, y = point  # unpacking
point[0]      # 3 (read-only)
```

Set (Elementi Unici)

```
s = {1, 2, 3}
s.add(4); s.remove(1)
a & b  # intersection
a | b  # union
a - b  # difference
```

Dizionari

Creare e Accedere

```
student = {"name": "Alice", "age": 20}
student["name"]      # "Alice"
student.get("gpa", 0) # 0 (default)
student["gpa"] = 3.85  # add/update
```

Dict Comprehension

```
sq = {x: x**2 for x in range(5)}
# {0:0, 1:1, 2:4, 3:9, 4:16}
```

Iterare

```
for k, v in student.items():
    print(f"{k}: {v}")
```

Metodi dei Dizionari

`d.keys()`	Tutte le chiavi
`d.values()`	Tutti i valori
`d.items()`	Tutte le coppie (chiave, valore)
`d.get(k, default)`	Ottieni con valore di default
`d.update(d2)`	Unisce d2 in d
`d.pop(k)`	Rimuove e restituisce il valore
`del d[k]`	Elimina la chiave
`"k" in d`	La chiave esiste? → bool
`len(d)`	Numero di voci

Controllo del Flusso

if / elif / else

```
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"
```

Operatore Ternario

```
status = "pass" if score >= 60 else "fail"
```

Cicli

Ciclo for

```
for fruit in ["apple", "banana"]:
    print(fruit)
```

range()

```
range(5)       # 0, 1, 2, 3, 4
range(2, 5)    # 2, 3, 4
range(0, 10, 2) # 0, 2, 4, 6, 8
```

Ciclo while

```
while count < 10:
    count += 1
```

enumerate() e zip()

```
for i, val in enumerate(["a", "b"]):
    print(i, val) # 0 a, 1 b

for a, b in zip([1, 2], ["x", "y"]):
    print(a, b) # 1 x, 2 y
```

break e continue

```
for x in range(10):
    if x == 5: break      # stop loop
    if x % 2 == 0: continue # skip
```

Funzioni

Definire e Chiamare

```
def greet(name, greeting="Hi"):
    return f"{greeting}, {name}!"

greet("Alice")          # "Hi, Alice!"
greet("Bob", "Hello") # "Hello, Bob!"
```

Valori di Ritorno Multipli

```
def min_max(lst):
    return min(lst), max(lst)
lo, hi = min_max([3, 1, 4, 1, 5])
```

*args e **kwargs

```
def total(*args): # args is a tuple
    return sum(args)
total(1, 2, 3) # 6

def info(**kwargs): # kwargs is a dict
    print(kwargs)
```

Funzioni Lambda

```
square = lambda x: x**2
square(5) # 25
sorted(lst, key=lambda x: x["age"])
```

Classi

```
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        return f"{self.name} says Woof!"

dog = Dog("Rex", "Lab")
dog.bark() # "Rex says Woof!"
```

Ereditarietà

```
class Puppy(Dog):
    def __init__(self, name, breed, toy):
        super().__init__(name, breed)
        self.toy = toy
```

Gestione degli Errori

```
try:
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
finally:
    print("Always runs")
```

File I/O

Leggere File

```
with open("data.txt") as f:
    content = f.read() # full text

with open("data.txt") as f:
    for line in f: # line by line
        print(line.strip())
```

Scrivere File

```
with open("out.txt", "w") as f:
    f.write("Hello\n")
```

"r" = lettura "w" = scrittura (sovrascrittura) "a" = aggiunta

CSV

```
import csv

with open("data.csv") as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(row["name"])

with open("out.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["name", "age"])
```

JSON

```
import json

data = json.loads('{"name":"Alice"}') # parse
text = json.dumps(data)               # serialize

with open("data.json") as f:
    data = json.load(f)               # read file
with open("out.json", "w") as f:
    json.dump(data, f, indent=2)      # write file
```

Richieste HTTP

```
import requests

# GET
r = requests.get("https://api.example.com/data")
r.status_code # 200
data = r.json() # parse JSON

# POST
r = requests.post(url, json={"key": "val"})
```

Basi di pandas

```
import pandas as pd
df = pd.read_csv("data.csv")
df.head()           # first 5 rows
df.shape            # (rows, cols)
df["name"]          # single column
df[df["age"] > 20]  # filter rows
```

Built-in Utili

`print()`	Stampa sulla console
`len()`	Lunghezza / conteggio
`type()`	Tipo dell'oggetto
`range()`	Sequenza di numeri
`enumerate()`	Coppie indice + valore
`zip()`	Accoppia elementi da iterabili
`sorted()`	Restituisce copia ordinata
`sum() min() max()`	Funzioni di aggregazione

Moduli

```
import math
from math import sqrt, pi
import pandas as pd # alias
```