Corso ITS:

PROGETTISTA E SVILUPPATORE SOFTWARE:

FULL STACK DEVELOPER E CLOUD SPECIALIST

Modulo: **Programmazione in Python**

Docente: *Andrea Ribuoli*

Mercoledì **9 Aprile** 2025

09:00 - 14:00

In [1]:
```
prova = "Buongiorno!"
```

In [2]:
```
type(prova)
```

Out[2]:
```
str
```

In [4]:
```
prova = 5.1
type(prova)
```

Out[4]:
```
float
```

In [10]:
```
lista = dir(prova)
for metodo in lista :
    print(metodo)
```

```
__abs__
__add__
__bool__
__ceil__
__class__
__delattr__
__dir__
__divmod__
__doc__
__eq__
__float__
__floor__
__floordiv__
__format__
__ge__
__getattribute__
__getformat__
__getnewargs__
__gt__
__hash__
__init__
__init_subclass__
__int__
__le__
__lt__
__mod__
__mul__
__ne__
__neg__
__new__
__pos__
__pow__
__radd__
__rdivmod__
__reduce__
__reduce_ex__
__repr__
__rfloordiv__
__rmod__
__rmul__
__round__
__rpow__
__rsub__
__rtruediv__
__setattr__
__setformat__
__sizeof__
__str__
__sub__
__subclasshook__
__truediv__
__trunc__
as_integer_ratio
conjugate
fromhex
hex
```

```
imag
is_integer
real
```

In [13]:
```
lista = dir(prova)      # mi restituisce l'elenco dei metodi offerti
                        # dalla classe float
for nome in lista :
    if not nome.startswith('__'):
        print(nome)
```

```
as_integer_ratio
conjugate
fromhex
hex
imag
is_integer
real
```

In [14]:
```
prova.hex()
```

Out[14]:
```
'0x1.4666666666666p+2'
```

In [15]:
```
prova = 7.2
prova.hex()
```

Out[15]:
```
'0x1.cccccccccccdp+2'
```

In [16]:
```
prova = "7.2"
prova.hex()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [16], in <cell line: 2>()
      1 prova = "7.2"
----> 2 prova.hex()

AttributeError: 'str' object has no attribute 'hex'
```

In [ ]:
```
prova = "ciao"
lista = dir(prova)      # mi restituisce l'elenco dei metodi offerti
                        # dalla classe str
for nome in lista :
    if not nome.startswith('__'):
        print(nome)
```

In [ ]:
```
prova = b"ciao"
lista = dir(prova)      # mi restituisce l'elenco dei metodi offerti
                        # dalla classe bytes
for nome in lista :
    if not nome.startswith('__'):
        print(nome)
```

In [21]:
```
ciao = "ciao€"
interno = ciao.encode()
interno
```

Out[21]:
```
b'ciao\xe2\x82\xac'
```

In [22]:
```
ciao = "ciao€"
interno = ciao.encode()
ciao == interno.decode()
```

Out[22]:
```
True
```

In [23]:
```
ciao = "ciao€"
interno = ciao.encode()
ciao == interno
```

Out[23]:
```
False
```

In [24]:
```
[] == list()
```

Out[24]:
```
True
```

In [25]:
```
{} == list()
```

Out[25]:
```
False
```

In [26]:
```
type({})
```

Out[26]:
```
dict
```

In [27]:
```
{} == dict()
```

Out[27]:
```
True
```

In [28]:
```
() == dict()
```

Out[28]:
```
False
```

In [29]:
```
type(())
```

Out[29]:
```
tuple
```

In [30]:
```
() == tuple()
```

Out[30]:
```
True
```

In [31]:
```
set()
```

Out[31]:
```
set()
```

In [33]:
```
type({ 5.5 })
```

Out[33]:
```
set
```

In [36]:
```
lista = [5, 5.3, "ciao€", b'ciao', True]
```

In [37]:
```
for elem in lista:
    print(type(elem))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bytes'>
<class 'bool'>
```

In [38]:
```
lista[2]
```

Out[38]: 'ciao€'

In [39]:
```
lista[2] = "buongiorno"
```

In [40]:
```
lista
```

Out[40]: [5, 5.3, 'buongiorno', b'ciao', True]

massimo.py

Acquisire una serie di numeri interi adottando un valore sentinella per terminare la lettura (`input()`). Quindi stampare il valore massimo. Adottare **q** come valore sentinella.

In [43]:
```
prova = []
prova.append(5)
prova.append(5.5)
prova
```

Out[43]: [5, 5.5]

In [44]:
```
!python resoconto.py massimo
```

```
 1  Mirco  Azzolini
 2  Wallace  Bezerra Beretta
 3  Alexandru Razvan  Brasovianu
 4  Edoardo  Caprini
 5  Maryuri  Catozzi
 6  Federico  De Grandis
 7  Maikol  Freddari
 8  Sofia  Gaona
 9  Alessia  Gasparini
10  Enrico  Giorgi
11  Andrea  Kanakciu
12  Francesco  Marinelli
13  Filippo  Martino
14  Eleonora  Moroni
15  Norman  Muzi
16  Mattia  Roberti
17  Alessandro  Rovinelli
18  Davide  Sambughi
19  Maximiliano  Serafini
20  Giovanni  Sperandini
21  Alessio  Stomeo
22  Lesly Pierina  Vera Castillejo
```

In [41]:
```python
prova = []
lista = dir(prova)      # mi restituisce l'elenco dei metodi offerti
                        # dalla classe list
for nome in lista :
    if not nome.startswith('__'):
        print(nome)
```

```
append
clear
copy
count
extend
index
insert
pop
remove
reverse
sort
```

Esercizio (pagina 413, **P6.32**)

Un negozio di animali domestici vuole fare uno sconto ai clienti che acquistano un animale (o più) e almeno cinque altri articoli. Lo sconto è il 20% del costo degli altri articoli, mentre gli animali sono esclusi.

Scrivete la funzione `discount(prices, isPet, nItems)` che calcoli lo sconto sulla base delle informazioni ricevute sulla vendita in esame, costituita da `nItems` articoli: per l'articolo i-esimo, `prices[i]` è il prezzo prima dell'eventuale sconto e `isPet[i]` è *True* se l'articolo i-esimo è un animale.

Scrivete un programma che chieda al cassiere di digitare tutti i prezzi, ciascuno seguito da una *Y* se si tratta di un animale e da una *N* per tutti gli altri articoli, usando il prezzo *-1* come sentinella. Memorizzate in una lista i dati acquisiti, poi invocate la funzione che avete progettato e visualizzate lo sconto.

sconto.py

In [50]:
```
riga = "123.45 Y"
```

In [51]:
```
riga[0:riga.find(" ")]
```

Out[51]:
```
'123.45'
```

In [52]:
```
riga[-1]
```

Out[52]:
```
'Y'
```

In [ ]:
```
def discount(prices, isPet, nItems):
    SCONTO_APPLICATO = 0.2
    print(prices, isPet, nItems)
    if (nItems == 0 or
        nItems < 6 or
        len(set(isPet)) == 1):
        return 0.00;
    i = 0
    cnt = 0
    somma = 0.0
    while i < nItems:
        if not isPet[i]: # siccome isPet[i] e True se l'i-esimo
                         # elemento è un animale, not è il contrario
            somma += prices[i]
            cnt += 1
        i += 1
    if cnt >= 5:
        return somma * SCONTO_APPLICATO
    else:
        return 0.00;

prezzi = []
animale = []
counter = 0
riga = input("Passami prezzo e Y/N: ")
while riga != "-1":
    prezzi.append(float(riga[0:riga.find(" ")]))
    animale.append(riga[-1] == "Y")
    counter += 1
    riga = input("Passami prezzo e Y/N: ")
print(discount(prezzi, animale, counter))
```

```
Passami prezzo e Y/N:      if nItems < 6:         return 0.00;
```

In [2]:
```
!python3 sconto.py < sconto.txt
```

```
Passami prezzo e Y/N: Passami prezzo e Y/N: Passami prezzo e Y/N: Passami pr
ezzo e Y/N: Passami prezzo e Y/N: Passami prezzo e Y/N: Passami prezzo e Y/
N: [2.69, 2.69, 2.69, 2.69, 2.69, 55.6]
[False, False, False, False, False, True]
6
2.69
```

In [3]:
```
!python3 sconto.py < sconto.txt
```

```
Passami prezzo e Y/N: Passami prezzo e Y/N: Passami prezzo e Y/N: Passami pr
ezzo e Y/N: Passami prezzo e Y/N: [2.69, 2.69, 2.69, 55.6]
[False, False, False, True]
4
0.0
```

In [4]:
```
lista_iniziale = ["Andrea", "Ribuoli"]
```

In [7]:
```
lista_iniziale = ["Andrea", "Ribuoli"]
lista_iniziale[0] = "Giovanni"
lista_iniziale
```

Out[7]:
```
['Giovanni', 'Ribuoli']
```

In [8]:
```
lista_iniziale = ["Andrea", "Ribuoli"]
lista_secondaria = lista_iniziale
lista_iniziale[0] = "Giovanni"
lista_iniziale
```

Out[8]:
```
['Giovanni', 'Ribuoli']
```

In [9]:
```
lista_secondaria
```

Out[9]:
```
['Giovanni', 'Ribuoli']
```

In [10]:
```
lista_secondaria.append(30)
```

In [11]:
```
lista_iniziale
```

Out[11]:
```
['Giovanni', 'Ribuoli', 30]
```

---

uso il costruttore per una copia "profonda"

In [12]:
```
lista_iniziale = ["Andrea", "Ribuoli"]
lista_secondaria = list(lista_iniziale)
lista_iniziale[0] = "Giovanni"
lista_iniziale
```

Out[12]:
```
['Giovanni', 'Ribuoli']
```

In [13]:
```
lista_secondaria
```

Out[13]:
```
['Andrea', 'Ribuoli']
```

In [14]:
```
lista_secondaria[2]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [14], in <cell line: 1>()
----> 1 lista_secondaria[2]

IndexError: list index out of range
```

In [15]:
```
lista_secondaria.insert( 1, 30)
```

In [22]:
```
lista_secondaria
```

Out[22]:
```
[30, 'Andrea', 30, 'Ribuoli', 30, 30]
```

In [21]:
```
lista_secondaria.insert( 0, 30)
```

In [24]:
```
30 in lista_secondaria
```

Out[24]:
```
True
```

In [31]:
```
indice = lista_secondaria.index(30)
indice = lista_secondaria.index(30, indice + 1)
indice = lista_secondaria.index(30, indice + 1)
indice = lista_secondaria.index(30, indice + 1)
indice = lista_secondaria.index(30, indice + 1)
indice
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [31], in <cell line: 5>()
      3 indice = lista_secondaria.index(30, indice + 1)
      4 indice = lista_secondaria.index(30, indice + 1)
----> 5 indice = lista_secondaria.index(30, indice + 1)
      6 indice

ValueError: 30 is not in list
```

In [32]:
```
30 in lista_secondaria[4:5]
```

Out[32]:
```
True
```

In [33]:
```
30 in lista_secondaria[3:4]
```

Out[33]:
```
False
```

In [36]:
```
a = { 1, 3, 5, 7, 9 }
b = { 2, 4, 6, 8, 10 }
a.intersection(b)
```

Out[36]:
```
set()
```

In [37]:
```
a = { 1, 3, 5, 7, 9 }
b = { 2, 4, 6, 7, 8, 9, 10 }
a.intersection(b)
```

Out[37]: {7, 9}

In [38]:
```
a = { 1, 3, 5, 7, 9 }
b = { 2, 4, 6, 7, 8, 9, 10 }
a.difference(b)
```

Out[38]: {1, 3, 5}

In [39]:
```
a = { 1, 3, 5, 7, 9 }
b = { 2, 4, 6, 7, 8, 9, 10 }
b.difference(a)
```

Out[39]: {2, 4, 6, 8, 10}

In [25]:
```
lista = dir({})          # mi restituisce l'elenco dei metodi offerti
                         # dalla classe set
for nome in lista :
    if not nome.startswith('__'):
        print(nome)
```

```
clear
copy
fromkeys
get
items
keys
pop
popitem
setdefault
update
values
```

In [42]:
```
saluto = "Ci vediamo puntuali domani mattina"
lista = list(saluto)
insieme = set(lista)
insieme
```

Out[42]: {' ', 'C', 'a', 'd', 'e', 'i', 'l', 'm', 'n', 'o', 'p', 't', 'u', 'v'}

In [14]:
```
saluto = "Ci vediamo puntuali domani mattina"
lista = list(saluto)
insieme = set(lista)
occorrenze = {}    # chiave sarà la lettera, il valore in conteggio
for lettera in insieme:
    occorrenze[lettera] = saluto.count(lettera)
cnt = 0
for val in occorrenze.values():
    cnt += val
cnt == len(saluto)
occorrenze
```

Out[14]:
```
{'C': 1,
 't': 3,
 'd': 2,
 'i': 5,
 'n': 3,
 'u': 2,
 'e': 1,
 ' ': 4,
 'o': 2,
 'v': 1,
 'm': 3,
 'l': 1,
 'p': 1,
 'a': 5}
```

In [13]:
```
occorrenze
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [13], in <cell line: 1>()
----> 1 occorrenze.max()

AttributeError: 'dict' object has no attribute 'max'
```

conta.py

In [19]:
```
def conta(stringa):
    return (["i", "a"], { "b", "c" }, { "a": 5, "d": 2, "e": 1})

saluto = "Ci vediamo puntuali domani mattina"
risultato = conta(saluto)
print(f"a) le lettere più frequenti sono: {risultato[0]}")
print(f"b) le lettere minuscole non presenti: {risultato[1]}")
print(f"c) occorrenze per ciascuna lettera: {risultato[2]}")
```

```
a) le lettere più frequenti sono: ['i', 'a']
b) le lettere minuscole non presenti: {'b', 'c'}
c) occorrenze per ciascuna lettera: {'a': 5, 'd': 2, 'e': 1}
```

In [36]:
```
def conta(stringa):
    insieme = set(stringa)
    occorrenze = {}
    for lettera in insieme:
        occorrenze[lettera] = stringa.count(lettera)
    chiavi   = list(occorrenze.keys())
    conteggi = list(occorrenze.values())
    valore_massimo = max(conteggi)
    return ([chiavi[conteggi.index(valore_massimo)]],
            set(list("abcdefghijklmnopqrstuvwxyz")) - set(chiavi),
            occorrenze)

saluto = "Ci vediamo puntuali domani mattina"
#saluto = "ciao a voi"
risultato = conta(saluto)
```

```
print(f"a) le lettere più frequenti sono: {risultato[0]}")
print(f"b) le lettere minuscole non presenti: {risultato[1]}")
print(f"c) occorrenze per ciascuna lettera: {risultato[2]}")
```

```
a) le lettere più frequenti sono: ['i']
b) le lettere minuscole non presenti: {'s', 'k', 'b', 'q', 'z', 'h', 'j', 'w
', 'f', 'g', 'c', 'r', 'x', 'y'}
c) occorrenze per ciascuna lettera: {'C': 1, 't': 3, 'd': 2, 'i': 5, 'n': 3,
'u': 2, 'e': 1, ' ': 4, 'o': 2, 'v': 1, 'm': 3, 'l': 1, 'p': 1, 'a': 5}
```

In [20]:
```
dizio = {"a": 5, "b": 4, "c":5 , "d": 2 }
```

In [23]:
```
max(dizio)
```

Out[23]: 'd'

In [24]:
```
dizio.values()
```

Out[24]: dict_values([5, 4, 5, 2])

In [39]:
```
set1 = { 2, 4, 6, 8, 10, 12, 14, 16 }
set2 = { 1, 2, 4, 8, 16 }
set3 = { 3, 6, 9, 12, 15 }

# a
set1.union(set2).union(set3)
```

Out[39]: {1, 2, 3, 4, 6, 8, 9, 10, 12, 14, 15, 16}

In [43]:
```
# b
(set1.difference(set2).difference(set3)).union(
 set2.difference(set1).difference(set3)).union(
 set3.difference(set1).difference(set2))
```

Out[43]: {1, 3, 9, 10, 14, 15}

In [44]:
```
# c
(set1.intersection(set2).difference(set3)).union(
 set2.intersection(set3).difference(set1)).union(
 set3.intersection(set1).difference(set2))
```

Out[44]: {2, 4, 6, 8, 12, 16}

In [46]:
```
# d
d = set(range(1,26))
d.difference(set1)
```

Out[46]: {1, 3, 5, 7, 9, 11, 13, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25}

In [47]:
```
# e
d.difference(set1).difference(set2).difference(set3)
```

Out[47]: {5, 7, 11, 13, 17, 18, 19, 20, 21, 22, 23, 24, 25}

In [49]:
```
# f
d.difference(set1.difference(set2).difference(set3))
```

Out[49]:
```
{1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 11,
 12,
 13,
 15,
 16,
 17,
 18,
 19,
 20,
 21,
 22,
 23,
 24,
 25}
```