Concetti di programmazione

con il linguaggio Python - parte 2

Le funzioni

*"Una **funzione** (detta anche routine, subroutine, procedura, sottoprogramma o metodo), in informatica e nell'ambito della programmazione, è un particolare costrutto sintattico di un determinato linguaggio di programmazione che **permette di raggruppare, all'interno di un programma, una sequenza di istruzioni in un unico blocco**, espletando così una specifica (e in generale più complessa) operazione, azione (o elaborazione) sui dati del programma stesso in modo tale che, **a partire da determinati input, restituisca determinati output**.*

*L'importanza, la praticità e i **vantaggi di una funzione** stanno nel fatto che **può essere** "**chiamata**" ("richiamata", "invocata" o "attivata") **in diversi punti** del programma di cui fa parte ogni volta in cui si ha la necessità di farlo **come se fosse una singola istruzione** senza la necessità di doverne riscrivere ogni volta il relativo codice implementando dunque il cosiddetto **riuso di codice**, cui si aggiunge una più facile manutenibilità del codice all'interno del programma ed una più facile progettazione del software secondo la classica filosofia del **divide et impera**."* (Wikipedia)

In informatica una funzione…

1. … è pezzo di programma, costituito da una sequenza d'istruzioni raggruppate in un unico blocco, a cui viene dato un nome;
2. … deve essere definita prima del suo uso (altrimenti si verifica un errore "`NameError: name '…' is not defined`";
3. … può essere utilizzata *chiamandola* tramite il suo nome;
4. … riceve input tramite i suoi *parametri*, cioè nomi di variabili scritti tra le parentesi tonde che seguono il suo nome;
5. … produce *output* per il programma che l'ha chiamata tramite l'istruzione `return`, oppure verso il mondo esterno al programma (l'utente, un file, …) tramite la funzione `print`.

Esempi

Esempio 1

```
def minimo(n1, n2, n3):
    """Questa funzione trova il valore minimo
    tra i valori passati come parametri"""

    if n1 < n2 and n1 < n3: return n1
    if n2 < n3: return n2
    return n3

# programma principale che usa (chiama) la funzione minimo() nella print

print(minimo(23, 9, 64))
```

Esempio 2

```
def mese(d):
    nMese = int(d[3:5])
    if nMese >= 1 and nMese <= 12:
        if nMese == 1: return("gennaio")
        elif nMese == 2: return("febbraio")
        elif nMese == 3: return("marzo")
        elif nMese == 4: return("aprile")
        elif nMese == 5: return("maggio")
        elif nMese == 6: return("giugno")
        elif nMese == 7: return("luglio")
        elif nMese == 8: return("agosto")
        elif nMese == 9: return("settembre")
        elif nMese == 10: return("ottobre")
        elif nMese == 11: return("novembre")
        else: return("dicembre")
    else: return 0


# programma principale che usa (chiama) la funzione mese()

data = input("Inserire una data nel formato gg/mm/aaaa: ")
nomeMese = mese(data)
if nomeMese:
    print(data[0:2], nomeMese, data[6:])
else:
    print(data, "non contiene un mese corretto")
```

Parametri delle funzione

I parametri sono come delle variabili che possono essere usate all'interno delle funzioni e servono per definire gli input che il programma chiamante passa alla funzione.

`def nome_funzione(nome_parametro1, nome_parametro2, …)`

Una volta definito un parametro, la funzione si aspetta che per esso venga passato un valore. In caso contrario si verificherà un errore.
I nomi dei parametri nella definizione di funzione vengono detti *parametri formali*, mentre i valori o le variabili usate nella chiamata della funzione vengono detti *parametri attuali*. È importante l'ordine con cui vengono scritti i parametri attuali nella chiamata della funzione perché la posizione di ogni parametro formale deve essere la stessa del parametro formale corrispondente.

Ad esempio definiamo una funzione che testa la divisibilità di un intero.

```
def divisibile(n, d):
    if n%d==0:
        return True
    else:
        return False

# - - - programma principale (main)- - -
x = int(input("Intero da testare: "))
y = int(input("Divisore: "))
if divisibile(x,y):
    print(x, "è divisibile per", y)
else:
    print(x, "non è divisibile per", y)
```

È possibile definire dei parametri opzionali impostando per essi un valore di default tramite assegnamento direttamente al momento della sua definizione. Per esempio, nella funzione precedente è possibile definire il parametro d come opzionale assegnandogli il valore di default 2:

```
def divisibile(n, d = 2):
    if n%d==0:
        return True
    else:
        return False
```

Esecuzione della funzione

Al momento della chiamata avviene l'assegnamento del valore del parametro attuale al parametro formale; quindi il blocco d'istruzioni della funzione viene eseguito e al termine vengono restituiti gli output della funzione al programma chiamante.

Campo di validita' delle variabili (*scope* delle variabili)

Una variabile definita in una funzione non e' vista da fuori della funzione.
Puo' avere stesso nome di una variabile esterna senza confusione.
Per esempio:

```
def funz():
    a = 1

a = 2
funz()
print(a)
```

stamperà 2

Esercizi

Esercizio 1: scrivere una funzione che calcoli l'area di un cerchio dato il raggio.

Esercizio 2: scrivere una funzione che testi se un intero è primo oppure no, ritornando rispettivamente i valori booleani vero o falso

Esercizio 3: scrivere un programma che stampi i primi m termini della successione di Fibonacci utilizzando una funzione fibonacci(n) che ne calcoli l'n-esimo termine.

Esercizio 3: scrivere una funzione che testi se un intero appartiene alla successione di Fibonacci. Suggerimento: utilizzare la funzione fibonacci(n) realizzata per l'esercizio precedente.

Sezione aurea (o rapporto aureo)

La sezione aurea è la parte **a** di una linea **L** divisa nelle due parti diseguali **a** e **b,** (cioè **L = a + b**)- La lunghezza **a** (il segmento maggiore) ha una proporzione matematica particolare rispetto alla parte di linea rimanente **b (**il segmento minore):

a : b = L : a

"Questa proporzione è molto frequente in natura, e viene riconosciuta come ideale di bellezza e armonia. Nelle opere d'arte viene spesso usato il rettangolo aureo, la cui base è la sezione aurea dell'altezza. In pratica, se l'altezza è pari a 1, la base è 0,618. L'esempio più famoso di utilizzo delle proporzioni auree in architettura è il Partenone di Atene: la sua facciata, infatti, si può perfettamente inscrivere in un rettangolo aureo. Anche nella facciata del Palazzo dell'Onu a New York, al cui progetto ha partecipato Le Corbusier, si trovano rettangoli aurei. Nella Gioconda di Leonardo il rapporto aureo è stato individuato nella disposizione dei lineamenti del viso, nell'area che va dal collo a sopra le mani e in quella che va dalla scollatura dell'abito fino a sotto le mani. In anatomia si trova la sezione aurea nel rapporto tra l'altezza di un individuo e la distanza del suo ombelico da terra. Sono rettangoli aurei persino le carte da gioco napoletane e con esse molte tessere di uso comune (carte di credito, bancomat ecc)."
[tratto da https://www.focus.it/scienza/scienze/cose-la-sezione-aurea)]

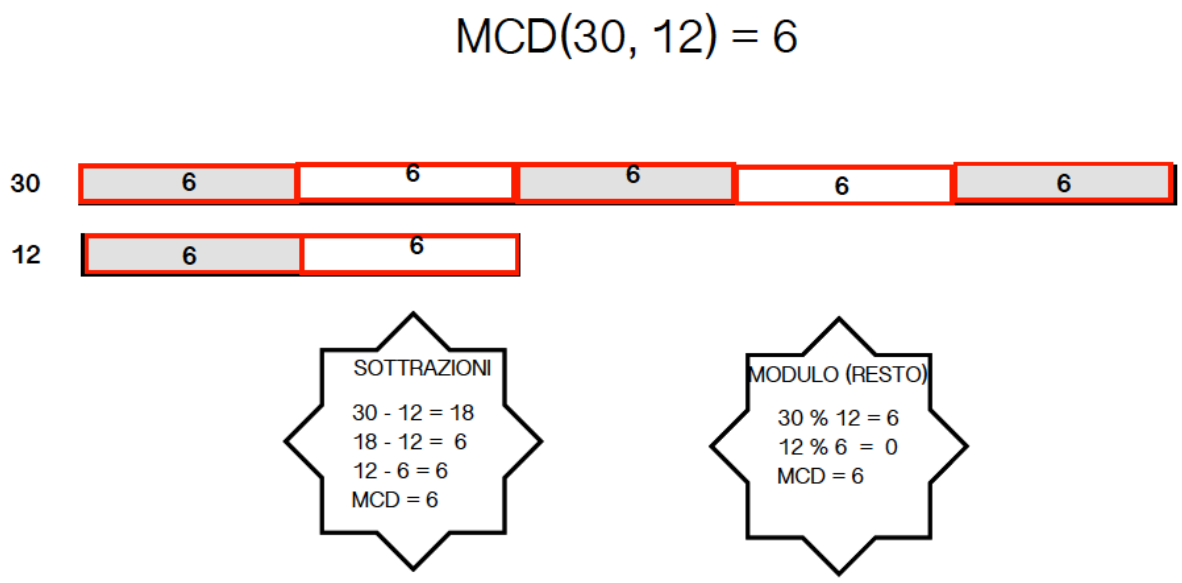
Keplero scoprì che il rapporto fra due numeri consecutivi della successione di Fibonacci approssimava via via, sempre più precisamente, il numero aureo.

Esercizio 4: utilizzando la funzione fibonacci(n) scrivere un programma che riceva in input l'intero m e stampi le approssimazioni della sezione aurea fino a fibonacci(m)/fibonacci(m-1).

Algoritmo di Euclide per il calcolo del MCD

"L'algoritmo di Euclide è un algoritmo per trovare il massimo comune divisore (indicato di seguito con MCD) tra due numeri interi. È uno degli algoritmi più antichi conosciuti, essendo presente negli Elementi di Euclide[1] intorno al 300 a.C.; tuttavia, probabilmente l'algoritmo non è stato scoperto da Euclide, ma potrebbe essere stato conosciuto anche 200 anni prima.

Certamente era conosciuto da Eudosso di Cnido intorno al 375 a.C.; Aristotele (intorno al 330 a.C.) ne ha fatto cenno ne I topici, 158b, 29-35. L'algoritmo non richiede la fattorizzazione dei due interi.
Dati due numeri naturali a e b, si controlla se b è zero (questa prima fase rientra ovviamente nell'ambito di un uso moderno dell'algoritmo ed era ignorata da Euclide e dai suoi predecessori, che non conoscevano lo zero). Se lo è, a è il MCD. Se non lo è, si divide a / b e si assegna ad r il resto della divisione (operazione indicata con "a modulo b" più sotto). Se r = 0 allora si può terminare affermando che b è il MCD cercato, altrimenti occorre assegnare a = b e b = r e si ripete nuovamente la divisione. L'algoritmo può essere anche espresso in modo naturale utilizzando la ricorsione in coda."
[tratto da https://it.wikipedia.org/wiki/Algoritmo_di_Euclide)]



Esercizio 5: realizzare la funzione mcd(x, y) che calcoli il massimo comune divisore tra due interi utilizzando l'algoritmo di Euclide.