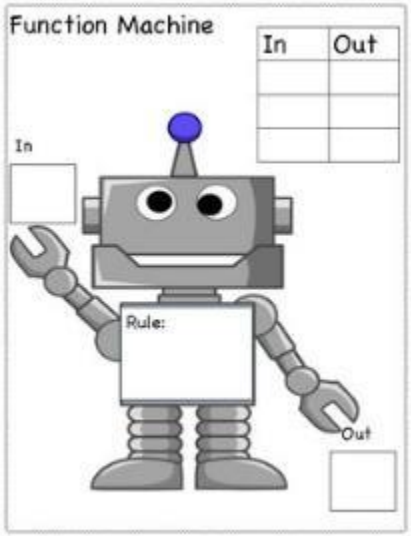


python

procedure e funzioni

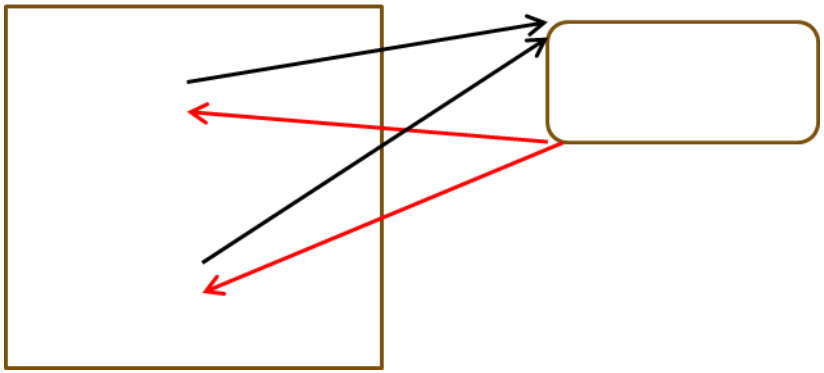
- una procedura può essere rappresentata come una ***macchina*** in grado di eseguire un certo compito quando ***attivata***
- in un primo tempo la macchina deve essere ***costruita***: la costruzione della macchina può essere paragonata con la ***dichiarazione e definizione*** della procedura
- la macchina viene poi ***attivata*** per eseguire il suo compito: può essere attivata ***più volte*** e tutte le volte ritorna ad eseguire il compito per cui è stata costruita
- l'avviamento della macchina può essere paragonato all'***esecuzione*** della procedura

python - esempio


```
#definizione

def stampa():
    print('**********************')
    print('classe 3A informatica')
    print('**********************')

#esecuzione
stampa()

#esecuzione ripetuta
for i in range(1,4):
    stampa()
```

esecuzione del codice

- la chiamata di una funzione provoca
  - l'***interruzione momentanea*** dell'esecuzione del codice,
  - l'esecuzione del ***codice della funzione***
    - al termine dell'esecuzione di questa
  - la ***ripresa*** del codice inizialmente sospeso



scambio di informazioni fra programma e procedura

- nell'esempio precedente il programma e la procedura non avevano la necessità di scambiarsi informazioni
- in generale è invece necessario uno ***scambio di informazioni*** fra programma e procedura *(o fra varie procedure)*
- la definizione di una ***variabile*** "***fuori***" dal programma e dalla procedura (***variabile globale***) permette ad entrambi di "***vederla***" quindi di scambiarsi, attraverso questa, informazioni

python – variabili globali

A. Ferrari

```
def cubo():
    ''' stampa il cubo della variabile globale x '''
    global x
    c = x ** 3
    print('cubo di',x,'=',c)

for x in range(1,11):
    cubo()
```

variabili locali e globali

- la possibilità da parte del programma e delle varie procedure di vedere e modificare il valore di una variabile globale è un aspetto
  - ***positivo***: permetto lo scambio di informazioni
  - ***negativo***: la modifica del valore di una variabile globale da parte di una procedura potrebbe alterare il comportamento dell'intero programma (***effetto collaterale***)
- esistono variabili che hanno significato solo all'interno di una procedura
  - queste variabili sono locali alla procedura (***variabili locali***) e quindi essere visibili sono in questa

- anche per la ***funzione*** è valida l'analogia con la macchina
  - la ***macchina-funzione*** oltre ad eseguire il compito per il quale è stata costruita, ***restituisce*** il ***risultato***
- anche in questo caso avremo una fase di ***dichiarazione-costruzione*** ed una di ***chiamata-avviamento***
- nella fase di chiamata avremo la ***restituzione*** di un valore: il ***risultato*** della funzione
- la funzione avrà una ***terminazione esplicita*** (***return*** seguito da una ***espressione*** che rappresenta il ***valore*** della funzione)

python - funzione


```
import random

def lancia_dado():
    n = random.randint(1,6)
    return n

#esecuzione
risultato = lancia_dado()
print('il risultato del lancio è ...',risultato)

#esecuzione ripetuta
for i in range(1,11):
    print('il risultato del lancio è ...',lancia_dado())
```

scambio informazioni fra funzioni

- la ***memoria globale*** permette lo scambio di informazioni fra funzioni
- risulta però un metodo
  - ***complesso*** (necessità di sapere quali sono le variabili che servono a tutte le funzioni)
  - ***pericoloso*** (una funzione può alterare in modo inatteso una variabile globale)
- l'utilizzo dei ***parametri*** permette di ovviare al problema:
  - la funzione opera ***formalmente*** su variabili ***(parametri formali)*** che vengono associate a ***valori specifici*** al momento della chiamata ***(parametri attuali)***

definizione di funzione

- ***operatore***, applicato a ***operandi***, per ottenere un ***risultato***
- **`def`** per ***definire*** una funzione
- **`return`** per terminare e restituire un ***risultato***

```
def ipotenusa(c1, c2):
    ip = (c1 ** 2 + c2 ** 2) ** 0.5
    return ip
```

chiamata di funzione

- **def** definisce una funzione, ma non la esegue!
- per far ***eseguire*** una funzione è necessario «***chiamarla***»

```
lato1 = float(input(primo cateto? '))
lato2 = float(input(secondo cateto? '))
lato3 = ipotenusa(lato1, lato2)
print('lato3:', lato3)
```

- la definizione della funzione opera sui ***parametri formali***
- al momento della chiamata si definiscono i ***parametri attuali***
- le variabili definite nella funzione rimangono locali a questa

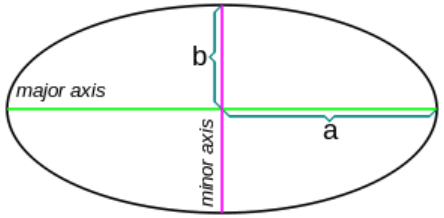
```
import random

def lancia_dado():
    n = random.randint(1,6)
    return n

risultato = lancia_dado()
print('il risultato del lancio è ...',n) #errore!!!
```

python
procedure e funzioni

esercizi

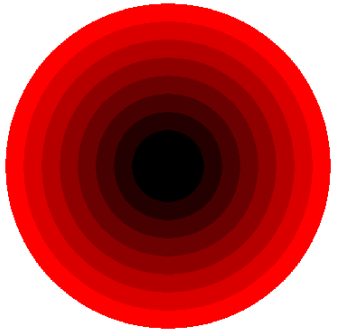
3.1 area di un'ellisse

- definire una funzione ***area_ellisse*** che:
  - riceve come parametri i ***semiassi*** di una ellisse: **a, b**
  - restituisce come risultato l'area dell'ellisse: $\pi \cdot a \cdot b$
- chiedere all'utente due valori:
- invocare la funzione area_ellisse con questi parametri
- stampare il risultato ottenuto

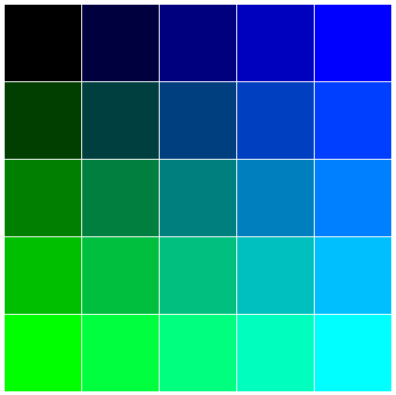


3.2 cerchi concentrici

- chiedere all'utente il ***numero*** di cerchi da disegnare
- ***disegnare*** i cerchi con ***raggio decrescente***, ma tutti con lo ***stesso centro***
- far variare il ***colore*** dei cerchi
  - dal ***rosso*** del livello più ***esterno***
  - fino al ***nero*** del livello più ***interno***

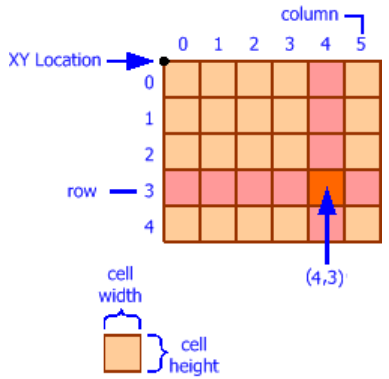


cominciare a disegnare un grosso cerchio rosso
poi, inserire l'operazione di disegno in un ciclo,
togliendo ad ogni passo 10 (p.es.) al raggio e al livello di rosso
infine, determinare automaticamente, prima del ciclo, le variazioni migliori per raggio e colore



3.3 griglia di colori

- chiedere all'utente dei valori per ***righe*** e ***colonne***
- mostrare una griglia di ***rettangoli*** di dimensione ***righe×colonne***
- partire da un rettangolo nero in ***alto a sinistra***
- in ***orizzontale*** aumentare gradatamente la componente di ***blu***
- in ***verticale*** aumentare gradatamente la componente di ***verde***



cominciare a creare una griglia di riquadri tutti neri
con due cicli for annidati
lasciare tra i riquadri un piccolo margine