Introduzione alla programmazione per le scuole superiori

basics

Nov 07, 2024

CONTENTS

Questo libro fa parte del materiale pensato per le scuole superiori

Obiettivi generali. Questo lavoro punta ad essere un'opera di *formazione alla* **tirchieria**, **prigrizia** *e* **onestà**, almeno in ambito informatico. I più benvolenti potranno riassumere questo obiettivo come il desiderio di *non buttare nel WC soldi, tempo, e pazienza*.

Oltre alle nozioni minime, l'obiettivo principale di questo libro è la formazione a:

- **indipendenza** in ambito informatico: evitare di pagare per qualcosa che non serve; evitare di pagare per qualcosa di inutile o dannoso in presenza di alternative libere; meglio dedicare le risorse a ciò che vale la pena pagare
- **ordine**: i moderni strumenti informatici permettono di lavorare in maniera ordinata, risparmiando tempo, soldi e pazienza
- **trasparenza** e **onestà**: anche se spesso in maniera non lineare, la conoscenza procede seguendo il metodo scientifico: i risultati mostrati e le tesi proposte devono essere supportate da dati e logica; i dati e le analisi svolte per poter produrre risultati devono essere disponibili, controllabili e soggetti a critica. Tutto il resto, almeno qui, almeno nell'ambito della conoscenza che procede con il metodo scientifico, sarà considerata confusione nella migliore delle ipotesi o direttamente *merda*.

Metodo.

- **Impostazione degli strumenti necessari. todo**…
- **Linguaggio di programmazione.** In questa introduzione si sceglie di usare **Python** come linguaggio di programmazione. Un approccio più generale all'informatica e alla programmazione prevederebbe l'utilizzo di altri linguaggio di programmazione (come C). Considerata la **diffusione** di Python, la quantità di **librerie disponibili** (con eventuali binding a librerie sviluppate usando linguaggi di programmazione più efficienti) e strumenti per la **programmazione collaborativa** e remota (**Colab**, **Jupyter**,…), Python risulta comunque una buona scelta per un corso improntato alla presentazione delle basi di programmazione dirette a un'applicazione abbastanza immediata.

  Verranno posta attenzione sulla sintassi "particolare" di Python quando si discosta maggiormente dagli altri linguaggi di programmazione.
- **Argomenti. todo**…

Part I

Introduzione alla programmazione

CHAPTER

ONE

INTRODUZIONE ALLA PROGRAMMAZIONE - IN PYTHON

L'introduzione alla programmazione in Python userà le risorse messe a disposizione da Google con il progetto Jupyter per lo sviluppo di codice open-source, con open-standard e servizi interattivi utilizzabili su diversi dispositivi usando diversi linguaggi di programmazione, come Python, Julia o R

Oltre al dispositivo elettronico utilizzato per consultare al materiale, non è necessario nessun altro dispositivo informatico: un account Google personale permette l'accesso libero ai servizi base di cloud computing di Colab

CHAPTER

TWO

TIPI

- tipi
- variabili
  - …
  - by-reference o by-value

CHAPTER

THREE

CONTROLLO DEL FLUSSO

Nei paradigmi di *programmazione imperativa* (**todo** fare riferimento ai paradigmi di programmazione. Ha senso questa distinzione?), vengono usate delle strutture di controllo del flusso di esecuzione di un programma.

Si possono distinguere due categorie delle strutture di controllo:

- alternativa: if-then, if-then-else
- iterazione: for, while, …

3.1 Alternativa

3.1.1 if-then e if-then-else

```
""" Test if-then """

a = 2.1

word = 'odd'
if ( a % 2 == 0 ):    # automatic casting into an int -> a % 2 = floor(a) % 2
    word = 'even'

print(f"User input ({a}) is an {word} number")
```

```
User input (2.1) is an odd number
```

```
""" Test if-then-else """

a = 15.7

if ( a % 3 == 1 ):
    reminder = 1
elif ( a % 3 == 2 ):
    reminder = 2
else:
    reminder = 0

print(f"User input ({a}). {a} % 3 = {reminder}")
```

```
User input (15.7). 15.7 % 3 = 0
```

3.1.2 switch

3.2 Iterazione

3.2.1 for loop

3.2.2 while loop

3.2.3 altri cicli

CHAPTER

FOUR

FUNZIONI

```
#> Define a function to tell if a number is positive or not
def is_positive(x):
    """ Function returning True if x>0, False if x<=0 """
    return x > 0

#> User input to test the function: tuple of numbers
n_tuple = [ -1, 2., .003, 1./3., -2.2, 0, -7.4 ]

#> Test function on all the elements in the user-defined tuple
for n in n_tuple:
    if ( is_positive(n) ):
        string = ''
    else:
        string = 'not '

    print(f'{n} is '+string+'positive')
```

```
-1 is not positive
2.0 is positive
0.003 is positive
0.3333333333333333 is positive
-2.2 is not positive
0 is not positive
-7.4 is not positive
```

4.1 Default value of function arguments

```
#> Define a user function to tell is the first argument is greater than the second.
# If no second argument is given, it's set = 0 (default value, defined in the␣
 ↪function)
def is_greater_than(x, y=0):
    """  """
    return is_positive(x-y)

a, b = -2, -3
print(f"Is {a} greater than {b}? is_greater_than({a}, {b}):"+str(is_greater_than(a,␣
 ↪b)))
print(f"Is {a} greater than  0 ? is_greater_than({a},    ):"+str(is_greater_than(a)))
```

```
Is -2 greater than -3? is_greater_than(-2, -3):True
Is -2 greater than  0 ? is_greater_than(-2,    ):False
```

CHAPTER

FIVE

CLASSI

- classi, metodi, oggetti
- ereditarietà, overloading

CHAPTER

SIX

LIBRERIE

- Definizione del concetto
- librerie “standard”, scritte da qualcun’altro
- scrivere una libreria

Part II

Introduzione al calcolo numerico

CHAPTER

SEVEN

INTRODUZIONE AL CALCOLO SCIENTIFICO

- Equazioni lineari
- Equazioni non lineari
- Approssimazione di funzioni
- Derivate
- Integrali
- Equazioni differenziali ordinarie:
  - problema di Cauchy ai valori iniziali
  - problema ai valori al contorno
- Ottimizzazione, vincolata e non

7.1 Sistemi lineari

La soluzione di sistemi lineari è un problema che compare in molte altre applicazioni di calcolo numerico …

Esistono due grandi classi di metodi/algoritmi per la soluzione di sistemi lineari:

- i metodi diretti, che si basano su una fattorizzazione della matrice
- i metodi indiretti, che si basano sul calcolo di prodotti matrice-vettore

todo

- Riferimento alla sezione di matematica sull'**algebra lineare** per dare le basi: cos'è un sistema lineare? Come può essere scritto usando il formalismo matriciale? Caratteristiche principali di una matrice (rango, nucleo, spettro, …)
- Riferimento ai problemi numerici: condizionamento?

7.2 Problemi non lineari

7.2.1 Metodo di bisezione

Il metodo di bisezione per la ricerca degli zeri di una funzione continua $F(x)$ si basa sul teorema dei valori intermedi per le funzioni continue.

Dati due numeri reali a, b tali che $f(a)\, f(b) < 0$, allora esiste un punto $c \in (a, b)$ tale che $f(c) = 0$.

```
"""
Example of bisection method

Find the solution of the problem f(x) = 0
with f(x) = e^x - x

"""

import numpy as np
from time import time

# Function f and its derivative
f  = lambda x: np.exp(x) + x
df = lambda x: np.exp(x) + 1

# Parameters of the bi-section method
tol = 1e-6
max_niter = 100

t1 = time()

# Find 2 values so that $f(a) f(b) < 0$
a, b = -2., 0.
niter = 0

if ( not f(a) * f(b) < 0 ):
    print("Bisection algorithm can't start, f(a)f(b)>= 0")
else:
    x = .5 * (a+b)
    fx = f(x)
    while ( np.abs(fx) > tol and niter < max_niter ):

        if ( f(x) * f(a) <= 0 ):  # new range [a,c]
            b = x
        else:  # new range [a,b]
            a = x

        # Update solution and residual
        x = .5 * (a+b)
        fx = f(x)

        # Update n.iter
        niter += 1

print("Bisection method summary")
if ( niter < max_niter ):
    print(f"solution, x = {x}")
    print(f"res  : {f(x)}")
    print(f"niter: {niter}")
    print(f"etime: {time()-t1}")
else:
    print(f"max n.iter reached without convergence")
    print(f"res: {f(x)}")
```

```
Bisection method summary
solution, x = -0.567143440246582
```

(continues on next page)

(continued from previous page)

```
res  : -2.348157265297246e-07
niter: 20
etime: 0.0008769035339355469
```

7.2.2 Metodo di Newton

Per trovare la soluzione del problema non lineare

$$F(x) = 0 \,,$$

il metodo di Newton sfrutta l'espansione in serie troncata al primo grado della funzione $F(x)$, per scrivere

$$0 = F(x^n + \Delta x) \approx F(x^n) + F'(x^n)\Delta x$$

e ottenere l'incremento della soluzione Δx come soluzione del sistema lineare

$$F'(x^n)\Delta x = -F(x^n)$$

e aggiornare la soluzione $x^{n+1} = x^n + \Delta x$.

```
"""
Example of Newton method

Find the solution of the problem f(x) = 0
with f(x) = e^x - x

"""

# import numpy as np (already imported?)

# # Function f and its derivative (already defined?)
# f  = lambda x: np.exp(x) + x
# df = lambda x: np.exp(x) + 1

# Parameters of the Newton method, for stopping criteria
tol = 1e-6          # tolerance on the residual |f(x)| < tol
max_niter = 10      # max n. of iterations      niter > max_niter

t1 = time()

# Initial guess, residual and number of iterations
x = -1.
res = f(x)
niter = 0

# Newton algorithm
while ( np.abs(res) > tol and niter < max_niter ):
    # Solve linear approximation step, and update solution
    dx = - res / df(x)
    x += dx

    #> Evaluate new residual and n. of iter
    res = f(x)
    niter += 1
```

(continues on next page)

(continued from previous page)

```
print("Newton method summary")
if ( niter < max_niter ):
    print(f"solution, x = {x}")
    print(f"res  : {res}")
    print(f"niter: {niter}")
    print(f"etime: {time()-t1}")
else:
    print(f"max n.iter reached without convergence")
    print(f"res: {res}")
    print(f"etime: {time()-t1}")
```

```
Newton method summary
solution, x = -0.567143285989123
res  : 6.927808993140161e-09
niter: 3
etime: 0.0005042552947998047
```

7.3 Approssimazione di funzioni

7.3.1 Interpolazione

7.3.2 Regressione

7.4 Derivate di funzioni

7.4.1 Differenze finite

Il calcolo della derivata di una funzione $f(x)$ derivabile in un punto x_0 può essere svolto utilizzando l'espansione locale in serie di Taylor di una funzione.

Derivata prima

Usando le espansioni

$$f(x+\Delta x) = f(x) + f'(x)\Delta x + f''(x)\frac{\Delta x^2}{2} + f'''(x)\frac{\Delta x^3}{3!} + o(x^3)$$

$$f(x-\Delta x) = f(x) - f'(x)\Delta x + f''(x)\frac{\Delta x^2}{2} - f'''(x)\frac{\Delta x^3}{3!} + o(x^3)$$

si possono ricavare gli schemi del primo ordine

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + o(\Delta x)$$

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + o(\Delta x)$$

e lo schema del secondo ordine usando le differenze centrate

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + o(\Delta x^2)$$

Derivata seconda

Usando le stesse espansioni in serie, si può ottenere uno schema del secondo ordine per la derivata seconda

$$f''(x) = \frac{f(x + \Delta x) - 2f(x) + f(x - \Delta x)}{\Delta x^2} + o(\Delta x^2)$$

7.5 Integrali

7.5.1 Integrazione di Newton-Cotes

- Formula del punto medio
- Formula del trapezio

7.5.2 Integrazione di Gauss

L'integrazione di Gauss permette di calcolare in maniera esatta l'integrale di una funzione polinomiale $p^(n)(x)$ su un intervallo $[a, b]$, come somma pesata della funzione valuatata in alcuni punti dell'intervallo,

$$\int_a^b p^{(n)}(x)dx = \sum_g w_g f(x_g) \ .$$

Per motivi di generalizzazione dell'algoritmo, nella definizione dei **pesi** w_i e dei **nodi di Gauss** x_i, l'integrale viene riportato all'integrale su un intervallo di riferimento, tramite una trasformazione di coordinate.

Per domini 1D, l'intervallo di riferimento per la quadratura di Gauss è l'intervallo $\xi = [-1, 1]$ e il cambio di variabili è

$$x = \frac{a+b}{2} + \frac{b-a}{2}(\xi - 1) \ ,$$

così che l'integrale originale può essere scritto come

$$\begin{aligned}\int_{x=a}^{b} p^{(n)}(x)dx &= \int_{\xi=-1}^{1} p^(n)(x(\xi))\frac{dx}{d\xi}d\xi = \\ &= \frac{b-a}{2}\int_{\xi=-1}^{1} p^{(n)}(x(\xi))d\xi = \frac{b-a}{2}\sum_g w_g\, p^{(n)}\left(x(\xi_g)\right)\end{aligned}$$

7.6 Equazioni differenziali ordinarie

7.6.1 Problemi di Cauchy ai valori iniziali

Approccio a un problema di Cauchy di ordine n

Un problema di Cauchy di ordine n

$$\begin{cases} F(y^{(n)}(x), y^{(n-1)}(x), \dots, y'(x), y(x), x) = 0 \\ y(x_0) = y^0 \\ y'(x_0) = y'^0 \\ \dots \\ y^{(n-1)}(x_0) = y^{(n-1),0} \end{cases}$$

con funzione incognita $y(x) : D \in \mathbb{R} \to \mathbb{R}$, può essere riscritto come un problema di *"ordine 1"* per la funzione incognita $\mathbf{z}(x) : D \in \mathbb{R} \to \mathbb{R}^n$, definita come

$$\mathbf{z}(x) = (z_0(x), z_1(x), \dots z_{n-1}(x)) := (y(x), y'(x), \dots, y^{(n-1)}(x)) \, .$$

Esplicitando le relazioni tra le componenti di $\mathbf{z}(x)$ e le derivate della funzione $y(x)$, $z_k(x) = y^{(k)}(x) = y^{(k-1)'}(x) = z'_{k-1}(x)$, il problema di Cauchy può essere riformulato come

$$\begin{cases} z'_0 - z_1 = 0 \\ z'_1 - z_2 = 0 \\ \dots \\ z'_{n-2} - z_{n-1} = 0 \\ F(z'_{n-1}(x), z_{n-1}(x), \dots, z_1(x), z_0(x)) = 0 \, , \end{cases} \quad , \quad \text{i.c.} \quad \begin{cases} z_0(x_0) = y^0 \\ z_1(x_0) = y'^0 \\ \dots \\ z_{n-1}(x_0) = y^{(n-1),0} \end{cases}$$

che può essere riscritto con il formalismo vettoriale come

$$\begin{cases} \mathbf{F}(\mathbf{z}'(x), \mathbf{z}) = \mathbf{0} \\ \mathbf{z}(x_0) = \mathbf{z}_0 \end{cases}$$

Caratteristiche (cenni)

- accuratezza, consistenza, convergenza
- stabilità: 0-, A- condizionata e incondizionata

Schemi numerici

Schemi numerici a un passo

Schemi numerici multi-step

7.6.2 Problemi al contorno

Differenze finite

Elementi finiti

Volumi finiti

7.7 Ottimizzazione

Le tecniche di ottimizzazione sono alla base di molti metodi di interesse, dall'approssimazione di funzioni, alla regolazione e controllo, agli algoritmi usati in intelligenza artificiale

7.8 Introduzione ai metodi in AI

- SL, supervised learning: regression and classification
- UL, unsupervised learning: clustering
- ML, machine learning: control

Part III

Supporto tecnico

CHAPTER

EIGHT

SUPPORTO TECNICO