

# Esercizi di Fondamenti di Informatica

Andrea Gussoni  
andrea1.gussoni at polimi.it

Politecnico di Milano

October 31, 2019

# Table of Contents

1 Ripasso Esercitazione Precedente

2 Ambiente di Programmazione

3 Primi Programmi

## Section 1

Ripasso Esercitazione Precedente

# Ripasso

Problemi con esercizi della precedente esercitazione?

- Codifica
- Dimensionamento memoria
- Diagrammi di flusso

# Ripasso

Problemi con esercizi della precedente esercitazione?

- Codifica
- Dimensionamento memoria
- Diagrammi di flusso

# Ripasso

Problemi con esercizi della precedente esercitazione?

- Codifica
- Dimensionamento memoria
- Diagrammi di flusso

## Section 2

# Ambiente di Programmazione

# Alternative

- IDE che integra tutto il workflow (metodo consigliato se siete al primo approccio con la programmazione)
- Editor di testo e interprete Python sulla vostra macchina



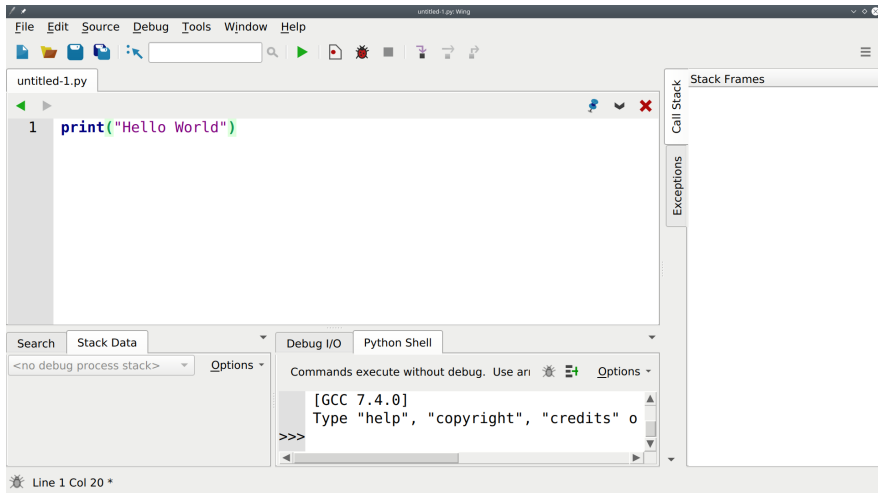
# IDE

- L'ambiente IDE permette di approcciarsi alla programmazione nel modo più veloce e semplice
- Noi useremo *Wing Python*, disponibile con licenza Freeware qui <sup>1</sup>
- Il software è disponibile per Windows, Mac OSX e Linux
- In caso non vi ritroviate con quanto visto a lezione, ricordatevi che noi useremo la versione *Wing 101*
- Attenzione alla versione Python!

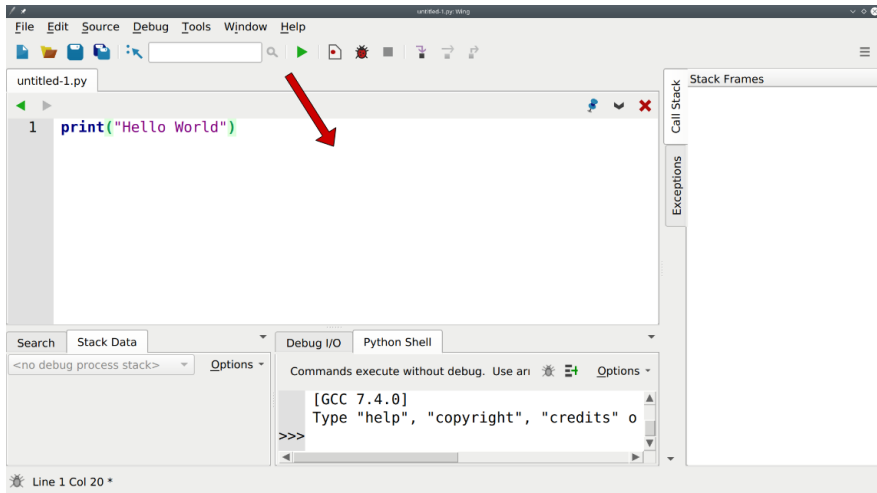
---

<sup>1</sup><https://wingware.com/>

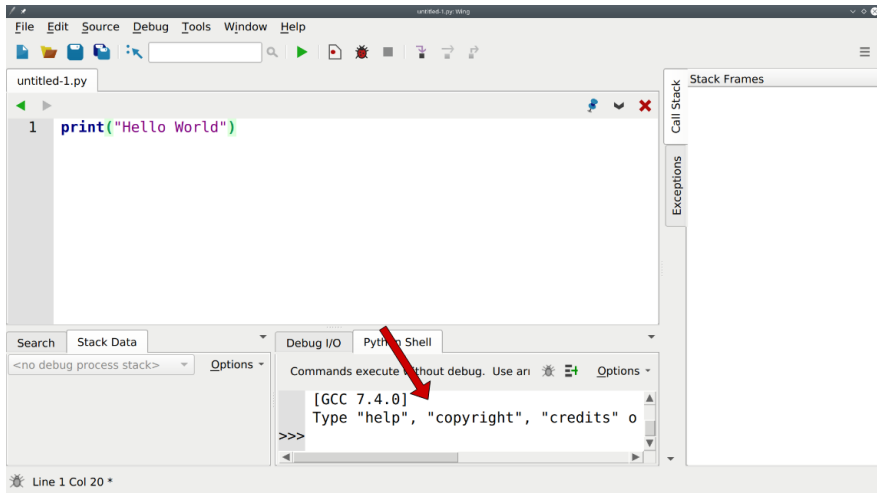
# Wing Python



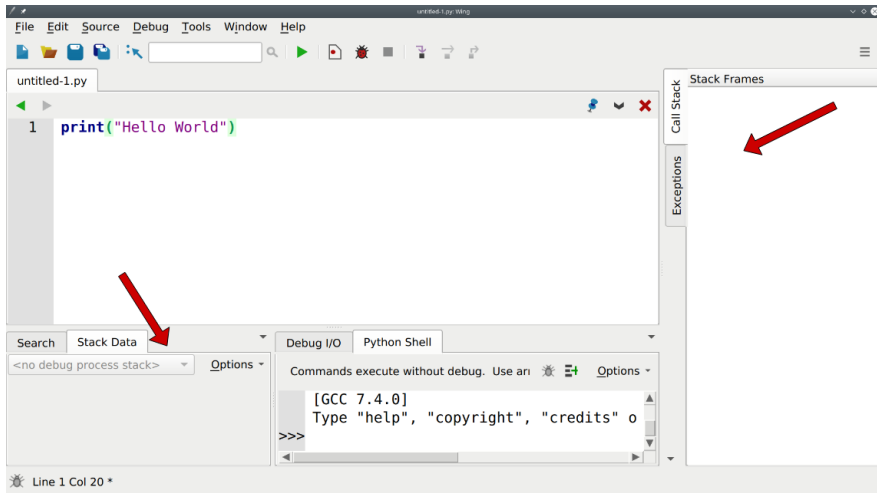
# Wing Python



# Wing Python

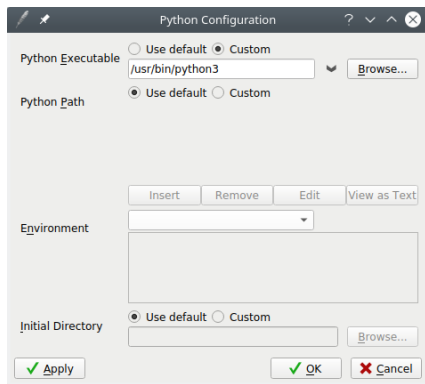


# Wing Python



# Wing Python

Fate molta attenzione alla versione dell'interprete Python che Wing sta usando, potete verificarlo sull'apposito menu di *configurazione python*:



# The Hard Way

- L'alternativa è utilizzare un semplice editor di testo per scrivere il codice e poi darlo in pasto all'interprete Python
- Solo se avete esperienza pregressa di programmazione e almeno un po' di conoscenza del terminale
- Molti editor di testo disponibili (Notepad++, Sublime Text, Atom, Vim) con cui scrivere il codice
- Per poi darlo in pasto all'interprete, basta invocare quest'ultimo con il nome del file come parametro
- Molto interessante anche usare l'interprete *ipython*

## Section 3

### Primi Programmi



# Variabili

- Una variabile è una etichetta che usiamo per riferirci ad un valore immagazzinato nella memoria durante l'esecuzione del nostro programma
- Spesso i valori utilizzati sono di tipo numerico o testuale
- Si usa l'operatore di assegnamento = per assegnare ad una variabile un valore: `a = 42`
- Fare attenzione alla differenza con l'operatore matematico di uguaglianza (che in Python diventa l'operatore `==`)

# Tipi Base

I tipi base che una variabile Python può assumere sono i seguenti:

- Valori interi (int): `I = 1`
- Valori frazionari (float): `F = 2.0`
- Valori complessi (complex): `C = 2 + 2j`
- Stringhe di caratteri (str): `Stringa = "Mondo"`
- Booleani (bool): `Flag = True`

# Operazioni Numeriche

Le operazioni effettuabili tra valori numerici sono:

- Somma:  $I + J$  (int, float, complex)
- Differenza:  $I - J$  (int, float, complex)
- Prodotto:  $I * J$  (int, float, complex)
- Divisione:  $I / J$  (int, float, complex, risultato sarà float, complex)
- Quoziente intero:  $I // J$  (int, float, risultato sarà int)
- Modulo:  $I \% J$  (int, float)
- Potenza:  $I ** J$  (int, float, complex)

# Operazioni tra Stringhe

Se definiamo le stringhe  $A = \text{"Hello"}$ ,  $B = \text{"World"}$ , abbiamo a disposizione i seguenti operatori:

- Concatenazione:  $A + B \rightarrow \text{HelloWorld}$
- Concatenazione ripetuta:  $A * 2 \rightarrow \text{HelloHello}$
- Accesso al singolo carattere:  $A[1] \rightarrow e$
- Lunghezza stringa:  $\text{len}(B) \rightarrow 5$

# Input e Output

Le funzioni di input e output ci permettono di interagire con il nostro programma:

- La principale funzione di output è la funzione `print()`, che stampa sulla console di esecuzione
- Può stampare stringhe, variabili numeriche e formattarne l'output
- Il programma:

---

```
Var = 5  
print("Var ha valore", var)
```

---

- Stamperà sulla console: Var ha valore 5

# Input e Output

- La principale funzione di input è invece la funzione `input()`
- Ci permette di acquisire da terminale un valore ed assegnarlo ad una variabile
- La funzione è bloccante, questo significa che l'esecuzione del programma si fermerà fino a quando l'utente non inserirà una serie di caratteri da tastiera. La sequenza viene considerata esaurita quando è introdotto il carattere speciale *invio*

---

```
istr = input("Inserire un intero")
a = int(istr)
```

---

- La funzione `input` assegnerà sembra un valore di tipo stringa (`istr`), che possiamo poi convertire a valore numerico intero come mostrato (a)

# Traduzione da Flowchart

Per i primi esercizi di programmazione che affronteremo:

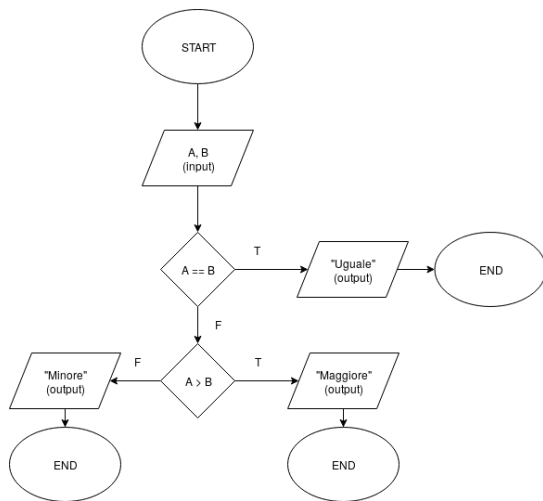
- Cominceremo con la stesura del diagramma di flusso
- Solo successivamente ci sposteremo ad implementare il programma in python

# Maggiore Minore

Ideare un programma che legge una coppia di valori interi ( $A$  e  $B$ ), e che stampi in output la stringa "maggiore" se  $A > B$ , "minore" se  $A < B$  e "uguale" se  $A = B$ .



# Maggiore Minore



# Maggiore Minore

Possibile soluzione:

---

```
a = int(input("Inserire il primo numero"))
b = int(input("Inserire il secondo numero"))

if a != b:
    if a > b:
        print("Maggiore")
    else:
        print("Minore")
else:
    print("Uguale")
```

---

# Maggiore Minore

Soluzione alternativa:

---

```
a = int(input("Inserire il primo numero"))
b = int(input("Inserire il secondo numero"))

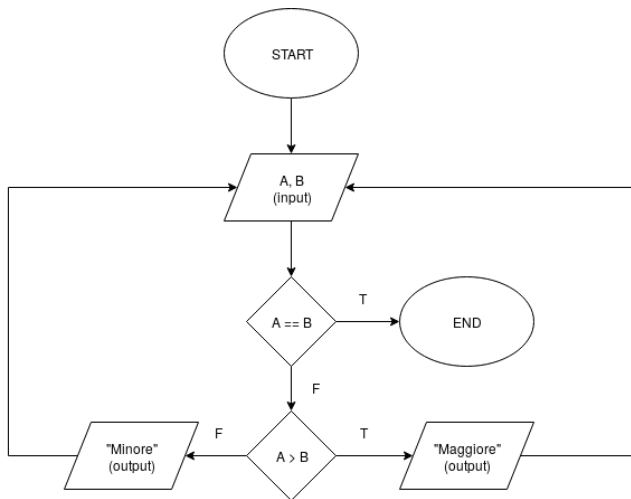
if a == b:
    print("Uguale")
elif a > b:
    print("Maggiore")
else:
    print("Minore")
```

---

# Maggiore Minore

Modificare la consegna precedente, in modo tale che il programma accetti una sequenza di coppie di numeri  $A$  e  $B$ , stampi i messaggi indicati precedentemente, ma che si concluda se per la coppia di valori inseriti  $A = B$ .

# Maggiore Minore



# Maggiore Minore

Possibile soluzione:

---

```
a = int(input("Inserire il primo numero"))
b = int(input("Inserire il secondo numero"))

while a != b:
    if a > b:
        print("Maggiore")
    else:
        print("Minore")
    a = int(input("Inserire il primo numero"))
    b = int(input("Inserire il secondo numero"))
```

---

# Maggiore Minore

Soluzione alternativa:

---

```
a = 0
b = 1

while a != b:
    a = int(input("Inserire il primo numero"))
    b = int(input("Inserire il secondo numero"))
    if a > b:
        print("Maggiore")
    else:
        print("Minore")
```

---

## Maggiore Minore

Un museo offre sconti per tre categorie di persone, i bambini (meno di 5 anni), gli studenti (tra i 18 e i 26 anni) e gli anziani (più di 70 anni).  
Ideare ora un programma che dato in input un valore intero rappresentante l'età, restituisca in output se questa persona ha diritto allo sconto.



# Maggiore Minore

Possibile soluzione:

---

```
x = int(input("Inserire anni"))

if x <= 5:
    print("Sconto")
elif x >= 18 and x <= 26:
    print("Sconto")
elif x >= 70:
    print("Sconto")
else:
    print("Prezzo intero")
```

---

# Maggiore Minore

Con un solo if

---

```
x = int(input("Inserire anni"))

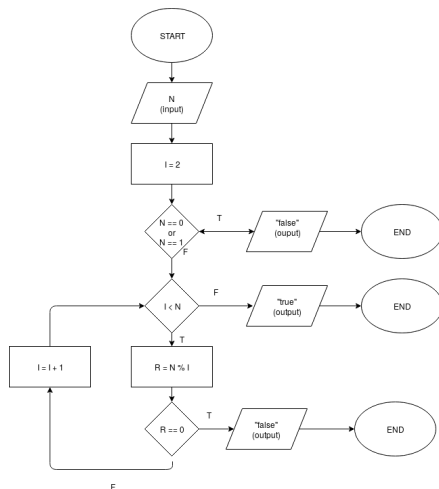
if (x <= 5) or (x >= 18 and x <= 26) or (x >= 70):
    print("Sconto")
else:
    print("Prezzo intero")
```

---

# Numeri Primi

Scrivere un programma che dato in input un numero intero  $X > 0$ , verifica se  $X$  è primo.

# Numeri Primi



# Numeri Primi

Possibile soluzione:

---

```
x = int(input("Inserire il numero"))  
i = 2
```

```
if x == 0 or x == 1:  
    print("Non primo")  
    exit()
```

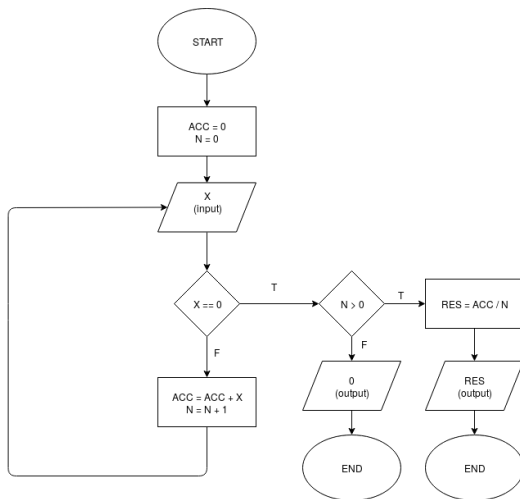
```
while i < x:  
    if x % i == 0:  
        print("Non primo")  
        exit()  
    i = i + 1  
print("primo")
```

---

# Media Aritmetica

Scrivere un programma che calcoli la media aritmetica di una serie di valori in input. Quando viene ricevuto il valore 0, l'immissione dei valori termina e la media viene calcolata.

# Media Aritmetica



# Media Aritmetica

Possibile soluzione:

---

```
acc = 0
n = 0

x = int(input("Inserire il valore"))

while x != 0:
    acc = acc + x
    n = n + 1
    x = int(input("Inserire il valore"))

if n > 0:
    res = float(acc) / n
    print("Media:", res)
else:
    print("Impossibile calcolare la media")
```

---



# Monte Carlo

Scrivere un programma che calcoli il valore di  $\pi$ , utilizzando il metodo di Monte Carlo.

# Monte Carlo I

Possibile soluzione:

---

```
from random import random

tentativi = 10000
interni = 0
i = 0

while i < tentativi :

    # Generiamo casualmente due valori per le
    # coordinate
    r = random()
    x = -1 + 2 * r
    r = random()
    y = -1 + 2 * r
```

## Monte Carlo II

```
# Verifichiamo se il punto cade all'interno
# della circonferenza
if x ** 2 + y ** 2 <= 1:
    interni = interni + 1

# Aumentiamo il counter delle
# iterazioni
i = i + 1

# Il rapporto tra punti interni e punti totali
# dovrebbe essere all'incirca al rapporto tra
# area del cerchio e area del quadrato
pi = 4.0 * interni / tentativi
print("La nostra stima per pi:", pi)
```

---