Primi passi con Python

Introduzione alla programmazione

Cos’è Python?

python™

Linguaggio di programmazione

Per impartire istruzioni al nostro computer!

Lo strumento per realizzare il nostro gioco

https://www.python.org/

La Shell interattiva

- Apriamo Thonny e proviamo!
- Ci sono anche un Playground e una Shell online:
  - http://www.ce.unipr.it/brython/
  - https://brython.info/console.html
- Possiamo usarlo come una calcolatrice?!
- È come una chat: io chiedo, Python risponde!
  - Attenzione, però: devo usare la sua lingua

```
>>> 3 + 5
8
>>> 7 / 4
1.75
>>> 4 * 5
20
>>>
```

Qual è la sua lingua?

Vediamo le operazioni sui numeri	
Somma	+
Differenza	-
Moltiplicazione	*
Divisione	/
Divisione intera	//
Resto della divisione	%
Elevamento a potenza	**

```
>>> 4 + 6
10
>>> 8.5 - 3.2
5.3
>>> (1 + 2) * (3 + 4)
21
>>> 2022 / 476
4.2478991596638656
>>> 2022 // 476
4
>>> 2022 % 476
118
>>> 2 ** 1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376
>>>
```

Qual è la sua lingua?

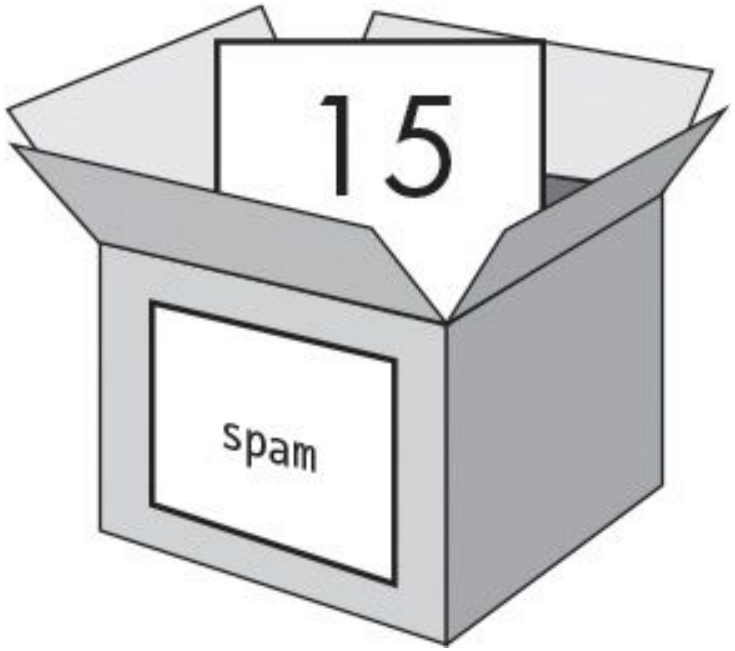
- Ma posso fare molto più che semplici operazioni:

Confronti (operazioni booleane)	
Uguale	`==`
Diverso	`!=`
Minore, minore e uguale	`<`, `<=`
Maggiore, maggiore e uguale	`>`, `>=`
Congiunzione	`and`
Disgiunzione	`or`
Negazione	`not`
Identità	`is`

```
>>> 1 is 2
False
>>> 1 == 2
False
>>> not 1 == 2
True
```

- Provare per credere!

Le variabili



Una variabile serve per ricordare il risultato di una espressione

Operazione di assegnamento: `spam = 15`

Alla sinistra un nome

Alla destra un valore

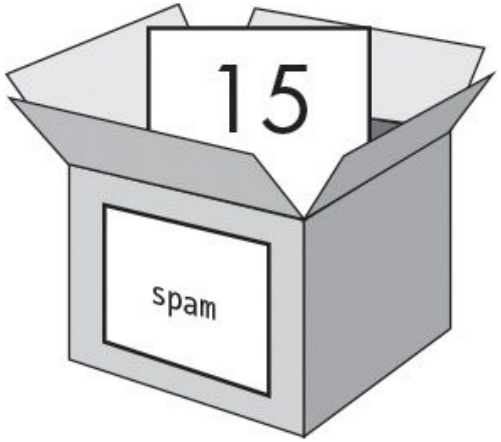
Da non confondere con l'uguaglianza:

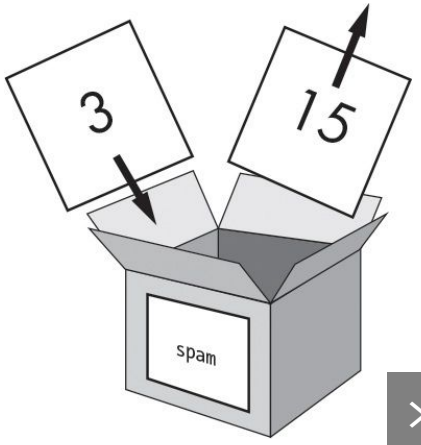
`spam == 15`

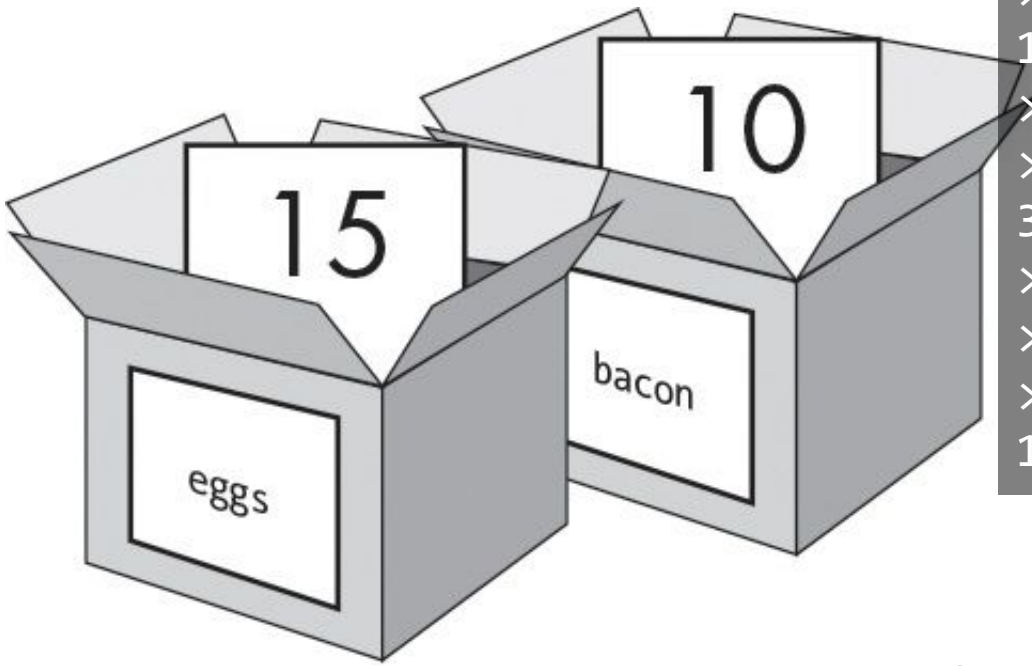
Le variabili

- Posso ri-assegnare una variabile, cambiandone il valore
- Posso creare altre variabili… anche con lo stesso valore!

- Posso vederle come:
  - Scatole che contengono “oggetti”
  - Etichette sopra a “oggetti”







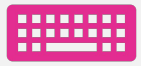

```
>>> spam = 15
>>> spam
15
>>> spam = spam - 12
>>> spam
3
>>> bacon = 5 + 5
>>> eggs = bacon + 5
>>> eggs
15
```

Le stringhe

- Le variabili possono contenere anche altri valori, non numerici…
- … ad esempio, del testo
- Valori testuali si chiamano stringhe
  - Sappiamo che sono stringhe perché le racchiudiamo tra virgolette

```
>>> str1 = "Monty Python's "
>>> str2 = 'Flying Circus'
>>> result = str1 + str2
>>> result
"Monty Python's Flying Circus"
```

Leggere e scrivere



`input`

legge una riga di testo, inserita dall'utente, in una variabile

Prima mostra un messaggio, e dà come risultato una stringa

`print`

scrive una serie di valori, su una riga

Inserisce uno spazio per separare i valori

Leggere e scrivere

- https://www.youtube.com/watch?v=Xel0c6mpqPA

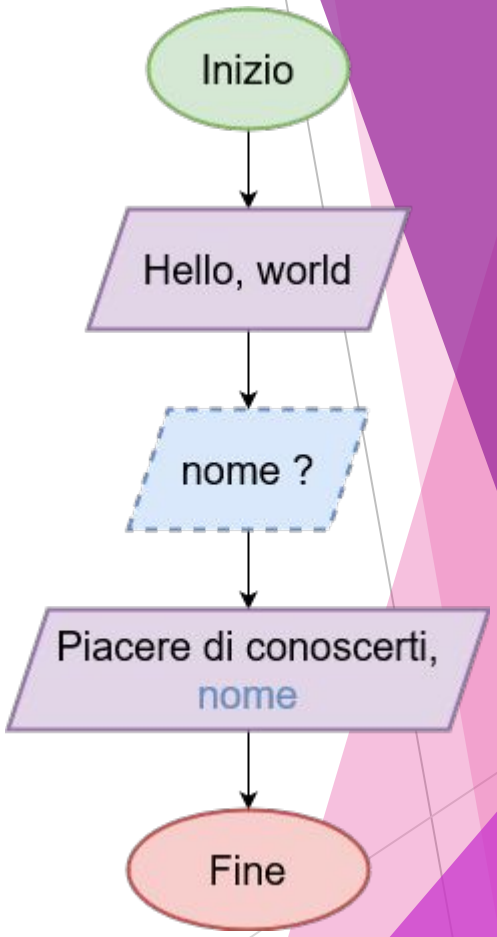
```
>>> quester = input("What is your name? ")
What is your name? Lancelot
>>> print("Right. Off you go,", quester, ".")
Right. Off you go, Lancelot .
```

Siamo pronte per creare un programma!

- Il nostro primo programma sarà Hello, world
- Da Thonny, creiamo un nuovo file…
  - … e salviamolo
- Dentro al file scriveremo le istruzioni

Hello, world

```
print("Hello, world")

nome = input("Come ti chiami? ")

print("Piacere di conoscerti,", nome)
```



- Poi clicchiamo sulla freccia verde: `Run current script`

Convertiamo gli input

- Questo codice:

```
anno_di_nascita = input("In che anno sei nata? ")

eta = 2022 - anno_di_nascita

print("Allora hai", eta, "anni!")
```

- Dà questo errore:

```
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

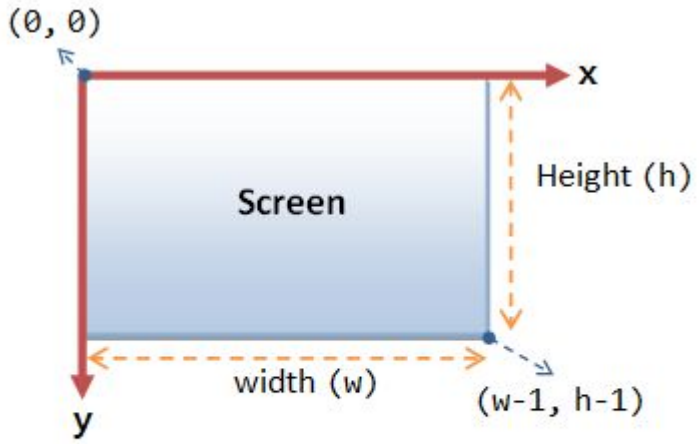
- Risolviamolo:

```
anno_di_nascita = input("In che anno sei nata? ")
anno_di_nascita = int(anno_di_nascita)

eta = 2022 - anno_di_nascita

print("Allora hai", eta, "anni!")
```

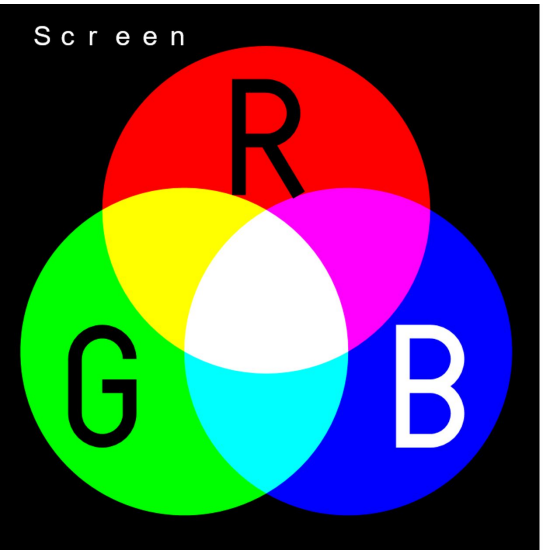
Disegnare su schermo

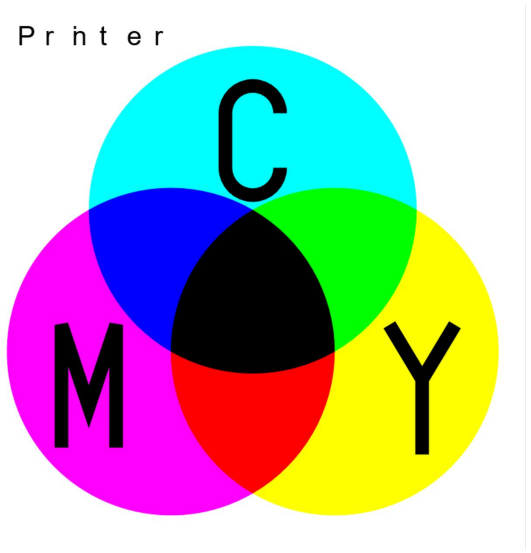
- L'immagine digitale è una griglia di pixel
  - Picture-element
- Come una scacchiera

- Disegneremo su canvas (tela)
- Attenzione alle coordinate
  - L'origine è in alto a sinistra



The 2D Screen Coordinates: The origin is located at the top-left corner, with x-axis pointing left and y-axis pointing down.





I colori!

- Sullo schermo, c'è una sintesi addittiva dei colori
- I primari sono:
  - Rosso
  - Verde
  - Blu

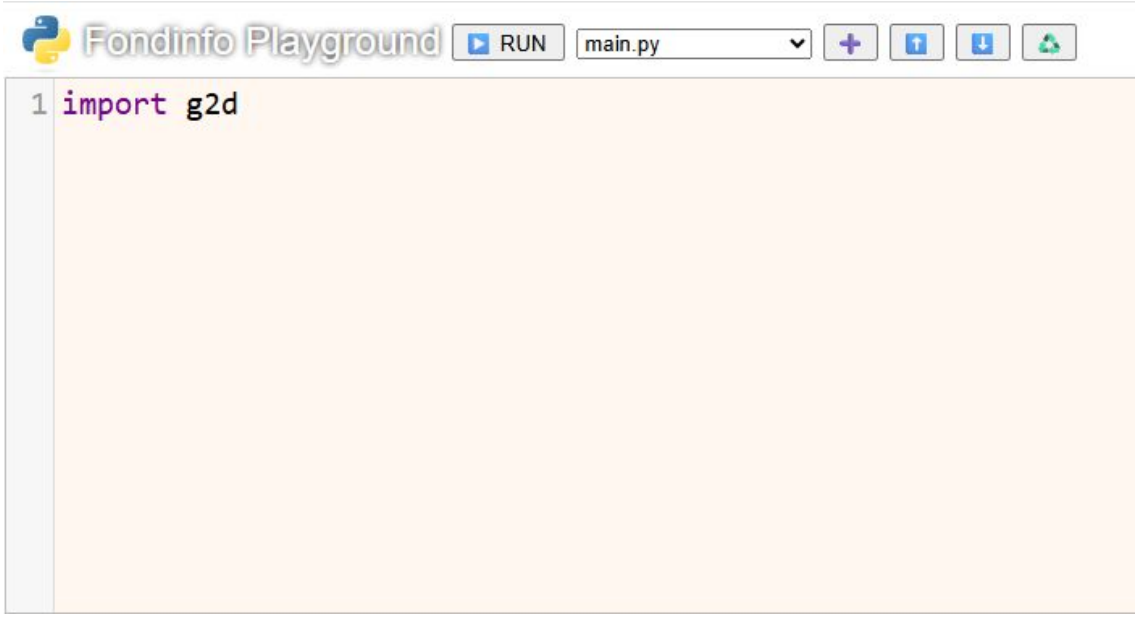
Disegnare con Python: la Tupla

- Sequenza immutabile di valori, tra parentesi tonde
  - Anche di *tipo diverso*

- Utile per la grafica!
  - `color = (red, green, blue)`
    - Ogni colore va da `0 a 255`
  - `size = (width, height)`
    - È la dimensione dell'immagine (risoluzione)
  - `point = (x, y)`
    - Ogni punto va da `0` alla dimensione dell'immagine (larghezza per la x, altezza per la y)

Disegnare con Python: g2d

- Useremo un modulo
  - Cioè uno strumento che fornisce funzionalità già esistenti
  - Che vuol dire?
  - È come se avessimo un libro da consultare... alla pagina che vogliamo

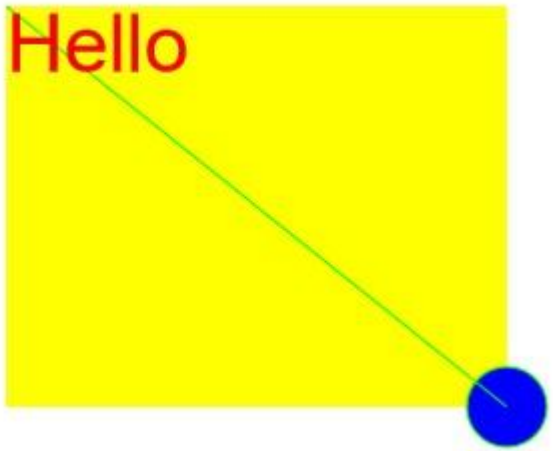


https://fondinfo.github.io/play/

Rettangoli e cerchi...

```
import g2d

g2d.init_canvas((600, 400))  # width, height

g2d.set_color((255, 255, 0))  # red + green = yellow
g2d.draw_rect((150, 100), (250, 200))  # left-top, size

g2d.set_color((0, 0, 255))
g2d.draw_circle((400, 300), 20)  # center, radius


g2d.main_loop()  # manage the window/canvas
```

… linee e testi




```
# ...

g2d.set_color((0, 255, 0))
g2d.draw_line((150, 100), (400, 300))   # pt1, pt2

g2d.set_color((255, 0, 0))
g2d.draw_text("Hello", (150, 100), 40)  # text, left-top, font-size

# ...
```

Battery included

- Python ha già molti moduli inclusi
  - Non necessitano d'installazione
- `import` ... importa uno o più moduli
  - Nomi da usare con prefisso (*namespace*)
- `from` ... `import` ... importa solo alcuni nomi
  - Nomi da usare senza prefisso

```
import math
y = math.sin(math.pi / 4)
print(y)  # √2 / 2
```

```
from math import sin, pi
print(sin(pi / 4))
```

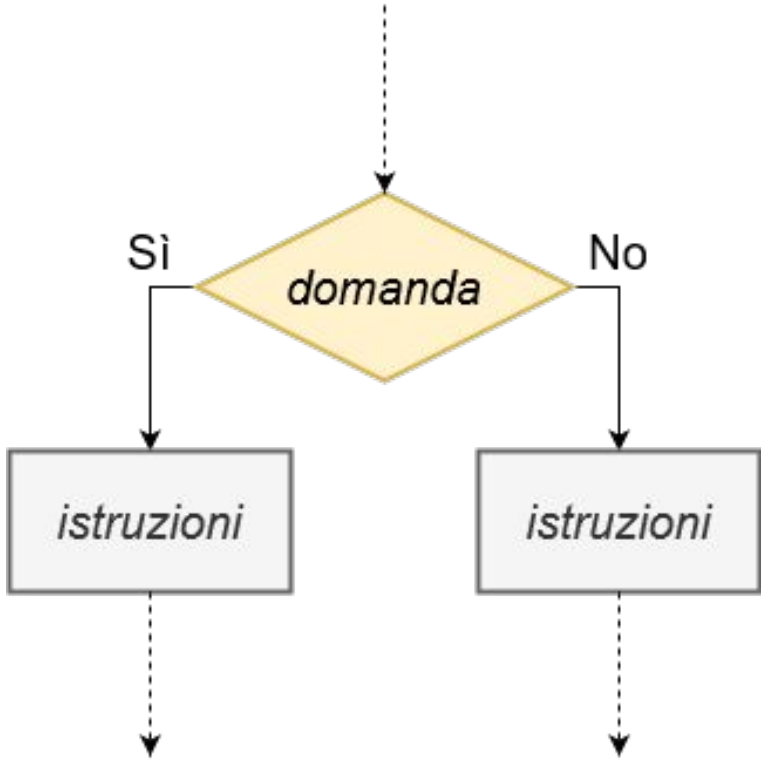
- Tutti gli `import` si scrivono all'inizio del programma
  - Preferibile, per evidenziare le dipendenze

Strutture di controllo

Selezionare e iterare

Se… allora… altrimenti…

- Possiamo chiederci se una certa condizione è verificata
  - Fare una domanda al nostro programma
- Nel qual caso, procedere con alcune operazioni
- Altrimenti, farne delle altre!
  - Oppure nessuna



Il costrutto `if else`

```
if anno_di_nascita > 2022:
    print("Non sei ancora nata?")
else:
    eta = 2022 - anno_di_nascita
    print("Allora hai", eta, "anni!")
```

- Attenzione alla sintassi
  - Dopo `if` ci si aspetta una condizione (domanda) e i due punti
  - Anche dopo `else` ci sono i due punti
  - Le istruzioni vengono indentate
- Provate a disegnare il diagramma di flusso di questo programma!

Di che colore è il cerchio?

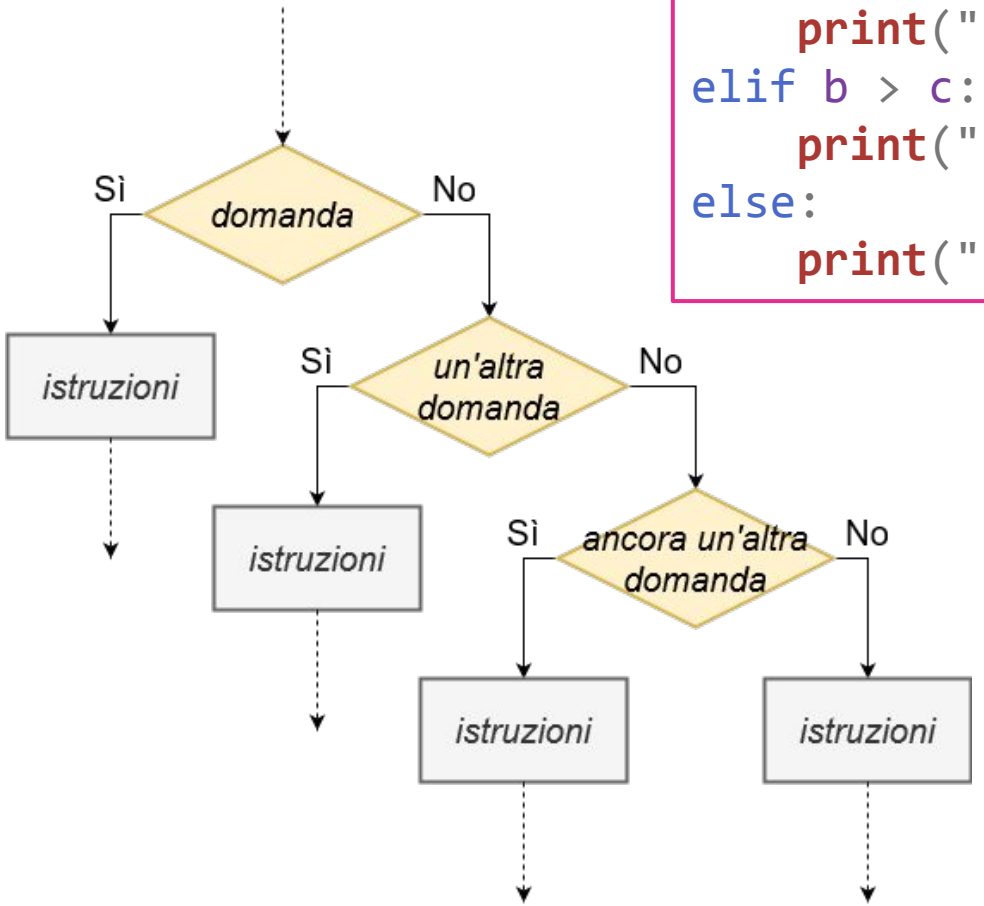
```
import g2d

g2d.init_canvas((400, 400))

raggio = int(g2d.prompt("Raggio? [50-99]"))

if 50 <= raggio and raggio <= 99:
    g2d.set_color((0, 0, 255))
    g2d.draw_circle((200, 200), raggio)

g2d.set_color((255, 255, 0))
g2d.draw_circle((200, 200), 25)

g2d.main_loop()
```

Di che colore è il cerchio?

```
import g2d

g2d.init_canvas((400, 400))

raggio = int(g2d.prompt("Raggio? [50-99]"))

if 50 <= raggio and raggio <= 99:
    g2d.set_color((0, 0, 255))
    g2d.draw_circle((200, 200), raggio)
else:
    g2d.alert("Raggio troppo grande o troppo piccolo!")

g2d.set_color((255, 255, 0))
g2d.draw_circle((200, 200), 25)

g2d.main_loop()
```

Annidare gli `if`

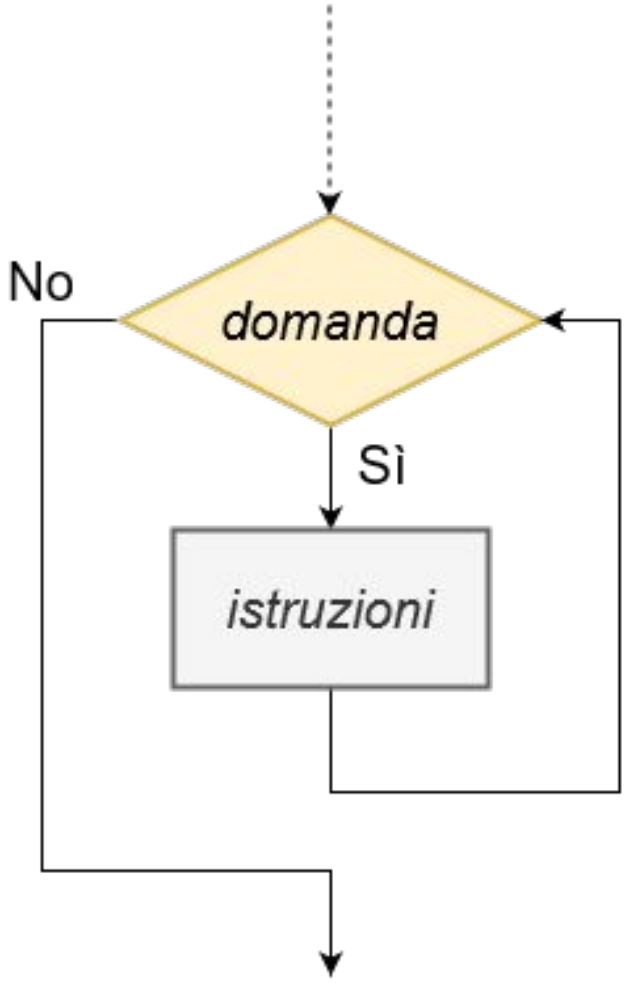
- Python fornisce anche il comando `elif` per chiederci «Altrimenti, se…»
  - `elif` sta per else if
- Anche in questo caso, `elif` è seguito dai due punti e le istruzioni sono indentate




```
if a > b and a > c:
    print("Il massimo è", a)
elif b > c:
    print("Il massimo è", b)
else:
    print("Il massimo è", c)
```

Mentre…

- Possiamo continuare ad eseguire delle istruzioni mentre una condizione è verificata
- Questo si chiama ciclo, o loop



Il costrutto `while`

- Dopo `while` c'è una condizione, seguita dai due punti
- Il blocco di istruzioni da ripetere è indentato

```
while anno_di_nascita > 2022:
    anno_di_nascita = input("Non è possibile! In che anno sei nata? ")
    anno_di_nascita = int(anno_di_nascita)

eta = 2022 - anno_di_nascita
print("Allora hai", eta, "anni!")
```

- Disegnate il diagramma di flusso di questo programma!

Di che colore è il cerchio?

```
import g2d

raggio = int(g2d.prompt("Raggio? [50-99]"))

while raggio < 50 or raggio > 99:
    g2d.alert("Out of range!")
    raggio = int(g2d.prompt("Raggio? [50-99]"))

g2d.set_color((0, 0, 255))
g2d.draw_circle((200, 200), raggio)

g2d.main_loop()
```

Proviamo a leggere questo programma per capire cosa succede:

```
import random

nome = input("Ciao! Come ti chiami? ")

numero_da_indovinare = random.randint(1, 20)

print("Bene", nome + ", sei pronta? Sto pensando a un numero tra 1 e 20")

mio_numero = input("Indovina il numero: ")
mio_numero = int(mio_numero)

tentativi = 1
```

Continua! »

```
while mio_numero != numero_da_indovinare and tentativi < 6:

    if mio_numero < numero_da_indovinare:
        print("No, troppo basso!")

    if mio_numero > numero_da_indovinare:
        print("No, troppo alto!")

    mio_numero = input("Indovina il numero: ")
    mio_numero = int(mio_numero)

    tentativi = tentativi + 1


if mio_numero == numero_da_indovinare:
    print("Brava", nome + "! Hai indovinato in", tentativi, "tentativi")
else:
    print("Peccato. Il numero a cui stavo pensando era", numero_da_indovinare)
```

Esempio di output:

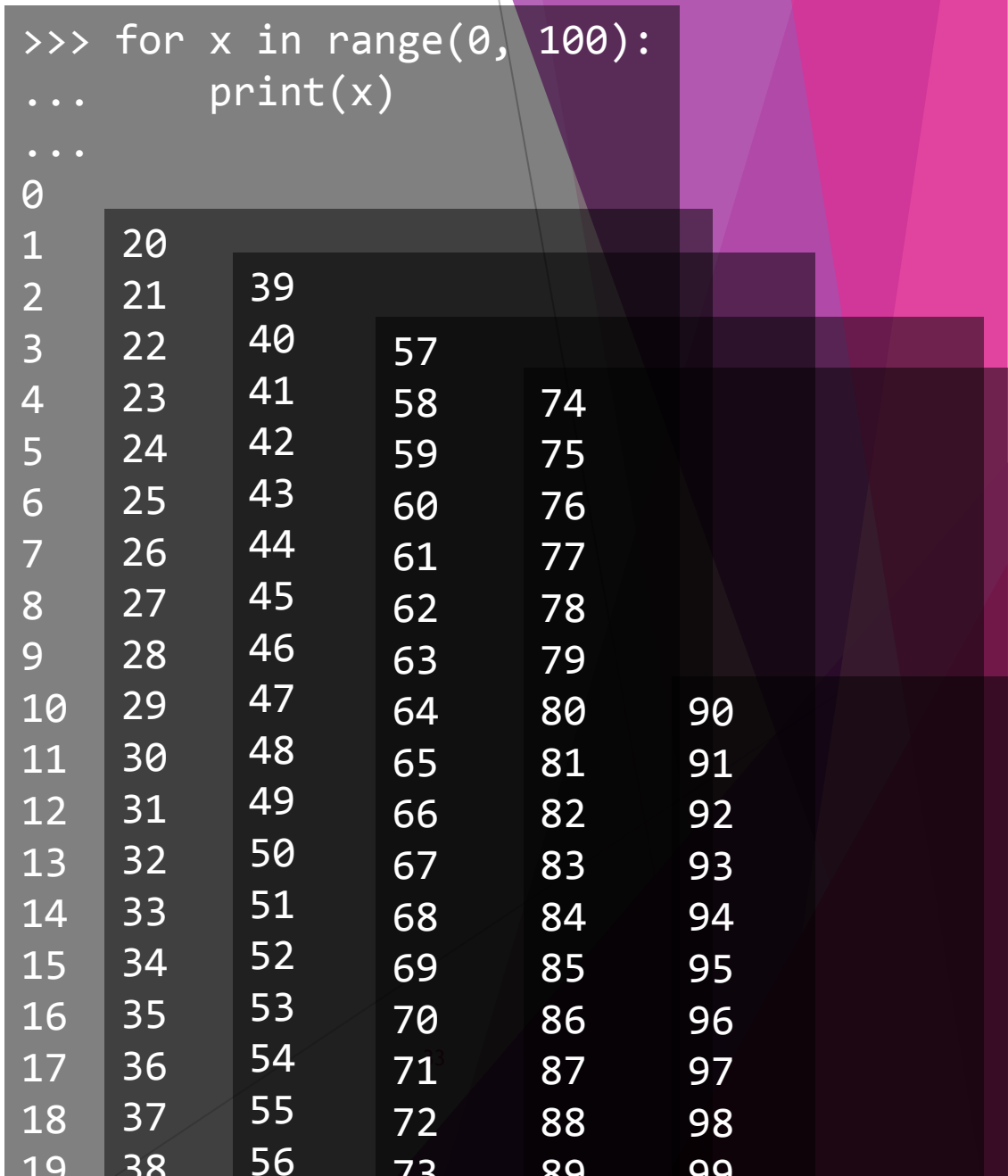
```
Ciao! Come ti chiami? Alice
Bene Alice, sei pronta? Sto pensando a un numero tra 1 e 20
Indovina il numero: 10
No, troppo basso!
Indovina il numero: 15
No, troppo alto!
Indovina il numero: 12
No, troppo alto!
Indovina il numero: 11
Brava Alice! Hai indovinato in 4 tentativi
```

Per…
il costrutto for

- Possiamo anche ‘ciclare’ su una sequenza di valori
  - ‘per x che va da 0 a 100’

```
for x in range(0, 100):
    print(x)
```

  - 0 è incluso
  - 100 è escluso
  - Le istruzioni all’interno vanno indentate



Sequenza di quadrati

```
import g2d

g2d.init_canvas((300, 300))

for i in range(5):  # (0, 1, 2, 3, 4)
    red = i * 60
    pos = i * 50
    g2d.set_color((red, 0, 0))
    g2d.fill_rect((pos, pos), (100, 100))

g2d.main_loop()
```