

# Valori e Tipi

- Un **valore** è un elemento base, es. una **stringa**, un **numero**, un **carattere**...
- Ogni valore ha un corrispondente **tipo**, che indica quali operazioni si possono fare con quel valore
  - `'Ciao Mondo'` e `"un po'"` sono **stringhe**  
`type('Ciao Mondo')` restituisce `<class 'str'>`
  - `2` è un valore di tipo **intero** (la funzione `type(2)` restituisce il tipo del valore passato come parametro, in questo caso `<class 'int'>`)
  - `42.3` è un valore di tipo decimale, detto **floating-point**, 'a virgola mobile' (`type(42.3)` restituisce `<class 'float'>`)
- `type('2')` restituisce `<class 'str'>`
- `type(2.0)` restituisce ....??

## Esercizio

- Che differenza c'è tra questi due programmi ?

Programma A

```
type(2)
type('2')
```

Programma B

```
print(type(2))
print(type('2'))
print("type(2)")
```

- Eseguirli e darsi una spiegazione del loro comportamento


# Valori ed Espressioni

- i valori si possono combinare con degli **operatori** ed ottenere delle **espressioni**.
- **espressioni aritmetiche**:
  - $3+5$ ,  $3-5$ ,
  - $7/3$  (divisione),  $7//3$  (divisione intera),  $7\%3$  (modulo, i.e. resto della divisione intera)
  - $4*5$ ,  $8**3$  (potenza)
- **operatori sulle stringhe**:
  - `'arco'+'baleno'` (concatenamento)
  - `'bello'*3` (ripetizione)

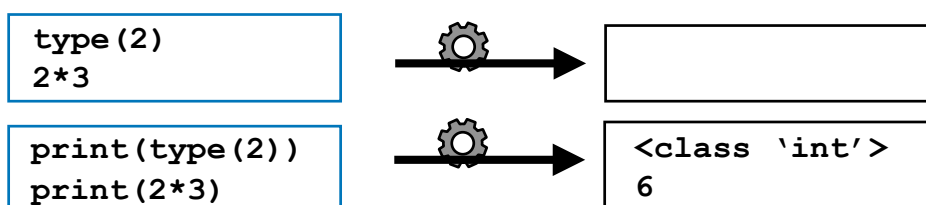
consuete regole di  
precedenza

# Espressioni e Comandi

- Un **comando** è una porzione di codice che la macchina **esegue** e che ha qualche **effetto**. es. `print('pippo')` provoca una stampa a video
- Un'**espressione** è una porzione di codice che la macchina **valuta**, cioè trova e restituisce il **valore** dell'espressione. es.  $(3+5)/3-4$  ha valore  $-8$

`print("The result of 4+8=", 4+8)`  `The result of 4+8= 12`

l'interprete **valuta** i valori delle due **espressioni** passate come parametro, ed esegue la funzione `print` con gli **argomenti**  
`"The result of 4+8="` e `12`



# Esercizio

- espressioni aritmetiche:
  - $3+5$ ,  $3-5$ ,
  - $7/3$  (divisione),  $7//3$  (divisione intera),  $7\%3$  (modulo, i.e. resto della divisione intera)
  - $4*5$ ,  $8**3$  (potenza)
- Scrivere un programma che visualizza il risultato delle precedenti espressioni:

```
print(7/3)
print(7%3)
```

```
print("Divisione: 7/3 = ", 7/3)
print("Modulo: 7%3 = ", 7%3)
```

# Esercizio

- Che output produce il seguente programma?

```
print("4-8=", 4-8)
print("4*8 = ", 4*8)
print("8/3=", 8/3)
print("8 raised to the power of 3=", 8**3)
print("7//3 = ", 7//3) # integer division
print("7%3=", 7%3) # rest of the integer division
print()
print("2*4+4*2 = ", 2*4+4*2)
print("2*(4+4)*2 = ", 2*(4+4)*2)
print('arco', 'baleno')
print('arco'+ 'baleno')
print(('arco'+ 'baleno')*3)
```

4-8= -4  
4\*8 = 32  
8/3= 2.6666666666666665  
8 raised to the power of 3=512  
7//3 = 2  
7%3=1  
  
2\*4+4\*2 = 16  
2\*(4+4)\*2 = 32  
arco baleno  
arcobaleno  
arcobalenoarcobalenoarcobaleno

- Che output produce il seguente programma?

```
print(2+4, 3*4)           6 12
```

```
print("2+4=", 2+2)        2+4=4   errore logico! (bug)  
                           l'esecuzione continua!
```

```
print(2+4=, 2+4)          SyntaxError   l'esecuzione si  
                           interrompe
```

## conversioni di tipo

```
print('USD$', 100)        stampa USD$ 100
```

```
print('USD$' + 100)       Type error: can only concatenate str  
                           (not "int") to str
```

```
print('USD$' + str(100))  stampa USD$100   str(100) converte un int in str
```

`str(4)`      **è un'espressione di valore** `'4'`    converte l'int 4 nella stringa `'4'`

`str(4.56)` **è un'espressione di valore** `'4.56'`    converte il float 4.56 nella stringa `'4.56'`

`int(4.65)`    **è un'espressione di valore** `4`    converte il float 4.65 nell'int 4

`int('4')`     **è un'espressione di valore** `4`     converte la stringa `'4'` nell'int 4

`int('4.65')` **è espressione la cui valutazione produce** **ValueError:** invalid literal  
for int() with base 10: `'4.65'`

# Valori e Variabili

- una **variabile** è un **nome** che fa riferimento ad un *valore*
- l'**istruzione di assegnamento** crea una nuova variabile, specificandone il nome, e le assegna un valore

```
messaggio = "cerca la X e scava"  
n = 24  
pi = 3.141592653589793
```

**ATTENZIONE:**  
= non indica  
l'uguaglianza!!

```
print(messaggio)      cerca la X e scava  
print(pi, type(pi))  3.141592653589793 <class' float'>  
print(n+2)           26
```

↖ valuta il **valore** della variabile e  
lo usa nell'operazione `_+2`

- i nomi delle variabili possono contenere numeri e lettere,
- ma non possono iniziare con un numero
- per convenzione sono minuscoli e si usa `_` es. `person_name` e `data_di_nascita`

## Assegnamento

Un successivo assegnamento **modifica** il valore della variabile

```
messaggio = "cerca la X e scava"  
messaggio = "il tesoro non si trova mai sotto la X"  
print(messaggio)      # stampa il tesoro non ...
```

```
x = 10  
x = x + 1  
y = x
```

↖ l'espressione a destra ha valore 10+1 cioè 11

← l'espressione a destra ha valore 11

l'istruzione di assegnamento

- è sempre della forma **`variabile = espressione`**
- viene eseguita nel modo seguente:
  - si **valuta l'espressione a destra** di `=` e si ottiene un **valore**
  - questo valore viene **assegnato alla variabile a sinistra** di `=`

# Esercizio

```
# Programma che scambia il valore di due variabili

x = 5
y = 10

# crea una variabile in più di supporto
temp = x
x = y
y = temp

print('valore di x dopo lo scambio:',x)
print('valore di y dopo lo scambio:',y)
```

Modificare il programma in modo tale che alla fine **stampi anche il valore che avevano x e di y prima dello scambio**. Queste **istruzioni aggiuntive di stampa non possono avere come argomento i numeri 5 e 10**.

## input da tastiera

Analogamente a `print`, in python c'è la *built-in* function **input**:

- ferma l'esecuzione **in attesa** che l'utente digiti qualcosa;
- alla pressione del tasto return/enter, il programma riprende
- e la funzione **input** **restituisce** ciò che è stato inserito sotto forma di **stringa**, che **può essere memorizzata in una variabile**.
- Si può anche specificare un *prompt* come testo che chiede l'input

```
testo1 = input("Inserisci testo ")
print("Hai inserito: ", testo1)
testo_2 = input("Come ti chiami? ")
print("Hai inserito: ", testo_2)
input("Oggi piove?")
print("Bye bye")
```

← il dato inserito dall'utente è restituito dalla funzione input, ma non è memorizzato quindi **va perduto**

# Come si comporta il seguente programma?

```
text_anni = input("quanti anni hai?")
eta = int(text_anni)
print(text_anni * 2)
print(eta * 2)
print(type(text_anni), type(eta))
input()
print('e ora?')
```

# Come si comportano i seguenti programmi?

```
anni = input("quanti anni hai?")
y = anni + 10
print(y)
```

```
text_anni = input("quanti anni hai?")
eta = int(text_anni)
y = eta + 10
print(y)
```

# Esercizi

```
name = input ("your name: ")
age = input ("your age: ")
year = input("your birth year: ")
future_age = int(year)+10
print("The age of ", name, "in 2032 will be ", future_age)
```

dov'è l'errore?

```
n = input()    # inserisce 3
x = n+'4'
y = int(n)+3
print(x, y)
```

cosa stampa?

## Valori e Tipi

- Un **valore** è un elemento base, es. una lettera, un numero, una parola...
- Ogni valore ha un corrispondente **tipo**, che indica quali operazioni si possono fare con quel valore
  - **2** è un valore di tipo **intero** (la funzione `type(2)` restituisce il tipo del valore passato come parametro, in questo caso `<class 'int'>`)
  - **42.3** è un valore di tipo decimale, detto **floating-point**, 'a virgola mobile' (`type(42.3)` restituisce `<class 'float'>`)
  - **'Ciao Mondo'** e **"un po'"** sono **stringhe**  
`type('Ciao Mondo')` restituisce `<class 'str'>`
- **True** e **False** sono i (soli) due valori di tipo **booleano**  
(`type(True)` restituisce `<class 'bool'>`)



# espressioni booleane

- le **espressioni booleane** (o *predicati*) sono espressioni la cui valutazione produce vero o falso,
- ad es. sono expr booleane quelle che si ottengono con i seguenti operatori:

- operatori di confronto**

5 == 6

x != 5

x > y

x < y

5 >= 3

3 <= 3



= è per assegnamento  
== è uguaglianza

## Esempi:

5 == 6 ha valore **False**

2+3 == 5 ha valore **True**

dati gli assegnamenti

x = 4    y = -4    z = x+y

x > y ha valore **True**

x < y ha valore **False**

x >= z ha valore **True**

z <= 0 ha valore **True**

# espressioni booleane

- le **espressioni booleane** (o *predicati*) sono espressioni la cui valutazione produce vero o falso,
- ad es. sono expr booleane quelle che si ottengono con i seguenti operatori:

- operatori di confronto**

5 == 6

x != 5

x > y

x < y

5 >= 3

3 <= 3



- operatori logici**

(x>0) **and** (x<10)

(x>0) **or** (y>0)

**not** (x==y)



## Esempi:

dati gli assegnamenti n=4    m=3    a=7

((n+m)/2 > 0) **and** (a==7) ha valore **True**

**not** (m==3 **or** n>5) ha valore **False**

# Esercizi

Le seguenti tre espressioni booleane sono **equivalenti**, cioè per qualsiasi valore delle variabili  $x$  e  $y$ , sono tutte e tre vere oppure tutte e tre false

$x \leq y$                        $(x < y) \text{ or } (x == y)$                        $\text{not } (x > y)$

1. Scrivere due espressioni booleane equivalenti all'espressione  $x \geq y$
2. Scrivere due espressioni booleane equivalenti a  $x \neq y$
3. Scrivere un'espressione booleana che ha valore **True** se e solo se la variabile  $x$  ha un valore che **sta** nell'intervallo di numeri  $[0, \dots, 10]$
4. Scrivere un'espressione booleana che ha valore **True** se e solo se  $x$  **non sta** nell'intervallo di numeri  $[0, \dots, 10]$
5. Scrivere un'espressione booleana che coinvolge 3 variabili  $x, y, z$  e ha valore **True** se e solo se  $x$  **sta** nell'intervallo di numeri  $[y, \dots, z]$

# Esercizi

Le seguenti tre espressioni booleane sono **equivalenti**, cioè per qualsiasi valore delle variabili  $x$  e  $y$ , sono tutte e tre vere oppure tutte e tre false

$x \leq y$                        $(x < y) \text{ or } (x == y)$                        $\text{not } (x > y)$

1. Scrivere due espressioni booleane equivalenti all'espressione  $x \geq y$
2. Scrivere due espressioni booleane equivalenti a  $x \neq y$
3. Scrivere un'espressione booleana che ha valore **True** se e solo se la variabile  $x$  ha un valore che **sta** nell'intervallo di numeri  $[0, \dots, 10]$
4. Scrivere un'espressione booleana che ha valore **True** se e solo se  $x$  **non sta** nell'intervallo di numeri  $[0, \dots, 10]$
5. Scrivere un'espressione booleana che coinvolge 3 variabili  $x, y, z$  e ha valore **True** se e solo se  $x$  **sta** nell'intervallo di numeri  $[y, \dots, z]$

come si può usare il  
computer per controllare  
la soluzione?

# Esercizio

Quali delle espressioni seguenti è equivalente all'espressione `num <= 23` ?  
(Sugg. la lista contiene 2 espressioni equivalenti)

- `num < 23 and num == 23`
- `num < 23 or num == 23`
- `num < 23 or num = 23`
- `not num > 23`
- `not num > 22`

come si può usare il  
computer per controllare  
la soluzione?

controllare la  
soluzione usando  
python

# Esercizio

Quali delle seguenti espressioni booleane ha valore True?

- A. `(True and False) and (not (True and False))`
- B. `(not (True or False)) or (True or False)`

- solo A
- solo B
- A e B
- né A né B

controllare la  
soluzione usando  
python

# Esercizio

- Scrivere un programma che calcola e stampa quanti secondi ci sono in 3 ore e quanti in 12 ore.
- Scrivere un programma che calcola e stampa quanti giorni sono trascorsi dal 1 gennaio 2023 ad oggi.

**ci sono tante diverse soluzioni,  
provate a confrontarle**