Università degli Studi di Verona
Dipartimento di Informatica
A.A. 2019-2020

APPUNTI DI “PROGRAMMAZIONE PYTHON”

Creato da *Davide Zampieri*

INPUT

```
n = int(input("Number: "))
```

STAMPA

```
print(i, end=" ")
```

- *,* --> concatenazione (con spazio finale)
- *end* --> specifica quale carattere termina la riga

```
print(str(a) + "^" + str(b) + " = " + str(r))
```

- *str(a)* --> converte *a* in stringa
- *+* --> concatenazione (senza spazio finale)

CICLO FOR

```
for i in range(a, b, n):
```

- *range(0, 100, n)* --> conta da 0 (incluso) a 100 (escluso) con step pari a *n*
- *range(10, 0, -1)* --> countdown da 10 (incluso) a 0 (escluso)

CICLO WHILE

```
while condizione:
      istruzioni
```

IF-ELIF-ELSE

```
if condizione:
      istruzioni
elif condizione:
      istruzioni
else:
      istruzioni
```

ESPRESSIONI DI CONFRONTO

- `<`
- `>`
- `<=`
- `>=`
- `==`
- `!=`

ESPRESSIONI BOOLEANE

- `not`
- `and`
- `or`

FUNZIONI

```
def fun(param):
      """
      Docstring
      """
      istruzioni
```

NUMERI PRIMI

```
def is_prime(n):
    for i in range(2, n // 2):
        if n % i == 0:
            return False
    return True
```

NUMERI TRIANGOLARI

```
def is_triangular_number(n):
    _sum = 0
    i = 1
    while _sum <= n:
        _sum += i
        if _sum == n:
            return True
        i += 1
    return False
```

FATTORIALE (ITERATIVO)

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result = result * i
    return result
```

FATTORIALE (RICORSIVO)

```
def factorial_r(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n *
            factorial_r(n - 1)
```

FIBONACCI (ITERATIVO)

```
def fibonacci_i(n):
    a = 0; b = 1; i = 0
    while i <= n:
        if i == n:
            return a
        tmp = a
        a = b
        b = tmp + b
        i += 1
```

FIBONACCI (RICORSIVO)

```
def fibonacci_r(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_r(n-1)
             + fibonacci_r(n-2)
```

POTENZA (ITERATIVO)

```
def power_of(base, exp):
    result = 1
    for i in range(exp):
        result = result * base
    return result
```

POTENZA (RICORSIVO)

```
def power_of_r(b, e):
    if e == 0:
        return 1
    else:
        return b *
            power_of_r(b, e-1)
```

POTENZA (OPERATORE BINARIO)

```
result = base ** exp
```

POTENZA (LIBRERIA)

```
result = pow(base, exp)
```

RADICE INTERA (ITERATIVO)

```
def sqrt_i(n):
    count = 0
    while n >= count * count:
        count += 1
    return count - 1
```

RADICE INTERA (RICORSIVO)

```
def sqrt_r(n, counter):
   if n < count * count:
       return count - 1
   else:
       return sqrt_r(n,count+1)
```

RANDOM

```
import random
random.randint(a, b)
random.choice(arr)
```

- *randint(a, b)* --> restituisce un intero casuale N tale che *a* <= N <= *b*.
- *choice(arr)* --> restituisce un elemento casuale della sequenza non vuota *arr*

DOCTEST

```
def fun(param):
      """
      >>> fun(0)
      0
      >>> fun(1)
      1
      >>> fun(2)
      1
      """
      istruzioni

if __name__ == '__main__':
      import doctest
      doctest.testmod(verbose=True)
```

STRINGHE

```
import string
string.ascii_uppercase
string.ascii_lowercase
s.lower()
```

- *ascii_uppercase* --> stringa con tutte le lettere maiuscole
- *ascii_lowercase* --> stringa con tutte le lettere minuscole
- *lower()* --> converte la stringa su cui viene chiamata in minuscolo

```
for c in s:
      istruzioni
```

- scorre ogni carattere *c* della stringa *s*

```
value = ord(c)
s = chr(value)
```

- *ord(c)* --> restituisce il codice Unicode del carattere *c*
- *chr(value)* --> restituisce il carattere rappresentato dal codice Unicode *value*

DIZIONARI

```
frequencies = {"a":0, "e":0, "o":0, "i":0, "u":0}
if char in frequencies:
      frequencies[char] += 1
for k in frequencies.keys():
      frequencies[k] = operazione
for k, v in frequencies.items():
      print("{}: {}".format(k, v))
```

ARRAY E MATRICI

```
m = []
for i in range(n_rows):
      m.append([0] * n_cols)
```

- *append(e)* --> aggiunge l'elemento *e* in fondo all'array
- *[e] * n* --> crea un array lungo *n* contenente tutti elementi *e*

FILE DI TESTO

```
for line in open(filename):
      for word in line.split():
            for char in string.punctuation:
                  word = word.replace(char, "")
            istruzioni
```

- scorro tutte le righe del file di testo
- scorro tutte le parole di una riga
- tolgo la punteggiatura

FILE CSV

```
for line in open(filename):
      line = line[:-1]
      x, y, z = line.split(";")
      x = line.split(";")[0]
      istruzioni
```

- tolgo il carattere finale (a capo)
- ottengo i valori nelle colonne spezzando la stringa usando il punto e virgola

MASSIMO	**MEDIA**

`def massimo(valori):`
`  m = valori[0]`
`  for n in valori:`
`  if n > m:`
`  m = n`
`  return m`
	`def media(valori):`
	`  somma = 0`
	`  for n in valori:`
	`  somma += n`
	`  return somma / len(valori)`

MASSIMO (LIBRERIA)	**MEDIA (LIBRERIA)**
`max(valori)`	`sum(valori) / len(valori)`

ANAGRAMMA

```
def is_anagramma(s1, s2):
      s2 = s2.upper()
      for char in s1.upper():
            if char in s2:
                  s2 = s2.replace(char, "", 1)
      for char in string.punctuation:
            s2 = s2.replace(char, "")
      return s2.strip() == ""
```

- se *char* (di *s1*) è presente in *s2,* lo sostituisco con la stringa vuota (solo quel carattere)
- sostituisco con la stringa vuota tutti i segni di punteggiatura
- elimino gli spazi con *strip()* e controllo se ho tolto tutti i caratteri

ECCEZIONI

```
try:
      n = int(input("Number? "))
      # Method 1
      assert n > 0
      # Method 2
      if n < 0:
            raise Exception()
      istruzioni
except:
      print("Input not valid, try again!")
```

- se nel blocco *try* sollevo una eccezione (con uno dei due metodi) salto al blocco *except*

IMMAGINI

```
from PIL import Image
image = Image.open(filename)
w = image.width
h = image.height
pixels = list(image.getdata())
# pixels[i] = (r, g, b)
image.putdata(pixels)
image.save(filename)
```

- *list* --> ritorna la lista dei pixel dell'immagine (array di triple)
- *putdata* --> riversa i valori della lista nell'immagine in memoria
- *save* --> salva l'immagine su file

TURTLE

```
import turtle
turtle.title(t)
turtle.setup(w, h, 0, 0)
turtle.tracer(0)

ninja = turtle.Turtle()
ninja.shape("turtle")
ninja.shapesize(w, h)
ninja.penup()
ninja.hideturtle()
ninja.goto(x, y)
ninja.pendown()
ninja.fd(n)
ninja.left(a)
ninja.color(c)
ninja.stamp()
ninja.write(str, font=("Arial", 40, "normal"), align="center")

turtle.fillcolor((r, g, b))
turtle.begin_fill()
disegno
turtle.end_fill()
```

- *fd(n)* --> va avanti (forward) di *n* passi
- *left(a)* --> ruota a sinistra di *a* gradi
- *shape(s)* --> cambia la forma della tartaruga
- *shapesize(w, h)* --> cambia le dimensioni della forma della tartaruga impostando come larghezza *w* e come altezza *h*
- *penup()* --> alza la penna (non disegna mentre la tartaruga si muove)
- *pendown()* --> abbassa la penna
- *hideturtle()* --> nasconde la forma della tartaruga
- *stamp()* --> stampa una copia della forma della tartaruga corrente
- *color(c)* --> cambia il colore della penna
- *write(str, …)* --> scrive la stringa *str* formattata secondo i parametri successivi
- *title(t)* --> cambia il titolo della finestra
- *setup(w, h, 0, 0)* --> crea una finestra di larghezza *w* e altezza *h* e la posiziona al centro
- *tracer(0)* --> massima velocità di disegno (nessuna animazione)
- *goto(x, y)* --> si sposta nel punto di coordinate *x*, *y* (per chiarezza è meglio se compreso tra *-w//2* e *w//2* di larghezza e tra *-h//2* e *h//2* di altezza)
- *begin_fill()* e *end_fill()* --> delimitano la zona da colorare del colore impostato da *fillcolor*