

# Taller de Python

## Clase I

---

Comisión de Talleres

Centro de Estudiantes Tecnológicos

# Table of contents

1. Introducción
2. Funciones
3. Variables
4. Controladores de flujo
5. Repaso

# Introducción

---

Para llevar a cabo las actividades propuestas, recomendamos descargar **Anaconda** de Continuum siguiendo el enlace que ofrecemos a continuación.

`https://www.continuum.io/downloads`

# ¿Qué es Python?

Python es un **lenguaje de programación** con las siguientes características:

# ¿Qué es Python?

Python es un **lenguaje de programación** con las siguientes características:

- Es un lenguaje de propósito general,

# ¿Qué es Python?

Python es un **lenguaje de programación** con las siguientes características:

- Es un lenguaje de propósito general,
- interpretado,

# ¿Qué es Python?

Python es un **lenguaje de programación** con las siguientes características:

- Es un lenguaje de propósito general,
- interpretado,
- multiparadigma,



# ¿Qué es Python?

Python es un **lenguaje de programación** con las siguientes características:

- Es un lenguaje de propósito general,
- interpretado,
- multiparadigma,
- fuertemente tipado.

¡También es muy sencillo!

# Funciones

---

Una función es un fragmento de código con un nombre asociado, reutilizable y que permite llevar a cabo una tarea en particular.

El interprete de Python tiene un número de funciones y tipos siempre disponibles.

`https://docs.python.org/3/library/functions.html`

# Funciones

Para definir nuestras propias funciones, usamos el comando **def**.

```
def F(x,y):  
    return x + y**2  
  
def G():  
    print('Hello world!')
```

- La línea que contiene al comando **def** termina siempre con ':'.  
• Todas las sentencias que formen parte de la definición de la función deben estar correctamente indentadas.

¿Qué diferencia existe entre **return** y **print**?

# Variables

---

Python tiene cinco tipos de datos estándares:

1. Números
2. *Strings*
3. Listas
4. Tuplas
5. Diccionarios

*'''A las variables a, b y c se le asignan distintos valores, como se muestra a continuación.'''*

```
a = 10
```

```
b = 10.0
```

```
c = 10 + 10j
```

*# Utilice la función type() para saber de qué tipo de  
# dato se trata*

¿Qué diferencias existen entre los distintos tipos? ¿es posible convertir de un tipo a otro?

# Strings

Los *Strings* en Python se identifican como un conjunto de caracteres contiguos encerrados entre comillas.

```
'Mi nombre es...'
```

El texto entre comillas es un tipo de dato y es por lo tanto posible operar con él.

```
a = 'nombre'  
type(a)  
len(a)  
b = a[0]  
2 * a
```

¿Qué resultados se obtienen al ejecutar las acciones sugeridas?



**Listas** Contienen elementos separados por comas escritos entre corchetes.

**Tuplas** Son semejantes a las listas pero sus elementos están escritos entre paréntesis.

**Diccionarios** Sus elementos consisten en pares *key-value* separados por comas y escritos entre llaves.

¿Qué otras diferencias existen entre estos tipos de datos?

# Listas, tuplas y diccionarios

*# Lista*

```
list = ['a', 'b', 3, 5, 7]
```

*# Tupla*

```
tuple = ('a', 'b', 3, 5, 7)
```

*# Diccionario*

```
dict = {1:'Ana', 2:'Bruno', 3:'Carlos'}
```

```
list.append(9) # Intente hacer lo mismo con la tupla
```

```
list
```

```
list[0]
```

```
list[2:3]
```

# Asignación de variables

**Softcoding** es un término que en programación hace referencia al hecho de obtener un valor o función desde una fuente externa. Es lo opuesto de **hardcoding**, término que hace referencia a programar valores y funciones en el código fuente.

Evitar el hard-coding de valores comúnmente modificados es una buena práctica en programación.

`input()` es una función que permite que el usuario ingrese datos y asignarlos a variables definidas en el código del programa.

```
Te = input('Ingrese la temperatura inicial en Kelvin: ')\nTe = float(Te)
```

¿Por qué necesitamos la segunda línea?

# Controladores de flujo

---

# Condicionales

```
number = 23
guess = int(input('Enter an integer : '))

if guess == number:
    print('Congratulations, you guessed it.')
    print('(but you do not win any prizes!)')
elif guess < number:
    print('No, it is a little higher than that')
else:
    print('No, it is a little lower than that')

print('Done')
```

Analice el código presentado prestando especial atención a la *indentación*.

# Condicionales

```
number = 23
running = True # boolean

while running:
    guess = int(input('Enter an integer : '))
    if guess == number:
        print('Congratulations, you guessed it.')
        running = False
    elif guess < number:
        print('No, it is higher than that.')
    else:
        print('No, it is lower than that.')
print('Done')
```

¿Qué diferencia(s) hay entre los dos códigos presentados?

```
for i in range(1, 5):  
    print(i)  
else:  
    print('The for loop is over')
```

¿Qué diferencia(s) supone ocupar **for** en lugar de **while**?

¿Preguntas?



# Repaso

---

**Tabla 1:** Operadores aritméticos

Símbolo	Interpretación
=	Igualdad
+	Suma
-	Resta
*	Multiplicación
**	Potencia
/	División
//	Parte entera del cociente
%	Resto de la división

**Tabla 2:** Operadores de relación

Símbolo	Interpretación
$==$	Igualdad
$!=$	Desigualdad
$<$	Menor a
$>$	Mayor a
$<=$	Menor o igual
$>=$	Mayor o igual

**Tabla 3:** Controladores de flujo.

Controlador	Intepretación
if...:	Si se cumple tal cosa, hacer...
elif...:	Si en cambio se cumple otra cosa, hacer...
else:	Para todos los demás casos, hacer...
for ... in ...:	Para los elementos en una lista, hacer...
while...:	Mientras se cumpla tal cosa, hacer...

# Enlaces de interés

- Comunidad Python Argentina  
<http://www.python.org.ar>
- AeroPython  
<https://github.com/AeroPython>
- Tutorialspoint  
<https://www.tutorialspoint.com/index.htm>
- Byte of Python  
[https://python.swaroopch.com/control\\_flow.html](https://python.swaroopch.com/control_flow.html)
- Python programming  
<https://pythonprogramming.net>

**Fin de la primera clase**

---