Curso de introducción a Python

Fundamentos de Python

Agenda

- Introducción.
  a. Nuestro equipo.
  b. Presentación del grupo.

- Programa de curso.
  a. Contenidos.
  b. Metodología y Evaluación.
  c. Uso del tecDigital.

- Clase #1: Fundamentos de Python.
  a. ¿Qué es Python?
  b. Instalación de Python.
  c. Objetos y tipos.
  d. Operaciones básicas.
  e. Estructuras.
  f. Ciclo For.
  g. Ciclo While.
  h. Funciones.

Nuestro Equipo

María Auxiliadora Mora Cross
Profesora del Instituto Tecnológico de Costa Rica. | Investigadora en Informática de la Biodiversidad.

Te Chen Huang
AI Engineer at Procter & Gamble. | Software Engineer

María Biarreta Portillo
Estudiante Ing. Computación
TEC

Josué Castro Ramírez
Estudiante Ing. Computación
TEC

Presentación del grupo

- ¿Cómo te llamas?
- ¿Dónde vives?
- ¿Qué estudias / estudiaste?
- ¿Qué disfrutas hacer en tu tiempo libre?
- ¿Cuál es tu expectativa de este curso?

Programa del Curso

Enlace al *documento*

¿Qué es Python?

Python es un lenguaje de programación de alto nivel de programación multiparadigma.

Goza de muchas librerías estándar para diferentes tipos de sistemas.

Desde el 2020 se utiliza python3 por defecto, y es la versión que se sigue actualizando.

Instalación de Python

Ingresar a la página y seguir los pasos indicados para su sistema operativo (Windows, Linux, MacOS):

https://elpythonista.com/como-instalar-python

Para efectos del curso, se utilizarán Cuadernos de Jupyter. Estos se pueden ejecutar de forma local utilizando una herramienta para crear ambientes de ejecución como Anaconda, o en la nube utilizando Google Colaboratory (https://colab.research.google.com).

Objetos y tipos

Cuaderno de Jupyter con el material:

https://colab.research.google.com/drive/1-5B2G_iql356efetod2UcZwqZmH9cO4L?usp=sharing

Tipo de objeto	Descripción	Ejemplo
int	Entero	10 / -7 / 235 / -100020
float	Flotante	0.003 / -5.32 / -89.85555
str	Cadena de caracteres	‘HOLA’ / ‘hola’ / "Hola"
bool	Valor de verdad	True / False
NoneType	Ausencia de valor	None

Operaciones básicas

- Operadores aritméticos
  - **+** (addition)
  - **-** (subtraction)
  - ***** (multiplication)
  - **/** (división)
  - ****** (exponent)

- Operadores de asignación
  - **=** (assign a value)
  - **+=** (add and re-assign; increment)
  - **-=** (subtract and re-assign; decrement)
  - ***=** (multiply and re-assign)

- Operadores de comparación (return either `True` or `False`)
  - **==** (equal to)
  - **!=** (not equal to)
  - **<** (less than)
  - **<=** (less than or equal to)
  - **>** (greater than)
  - **>=** (greater than or equal to)

2^3 ⟶ (2**3)

$\left(\frac{1}{2} - 3\right)^2$ ⟶ ((1/2 - 3)**2)

$\left(\frac{1}{2} - 3 \cdot x\right)$ ⟶ (1/2 - (3*x))

Estructuras

Estructura	Descripción	Ejemplo
String - str	Inmutable, ordenado	"Hola mundo!"
Lista - list	Mutable, ordenado	[1, 2, -3, "cuatro"]
Tupla - tuple	Inmutable, ordenado	(2, 80, True)
Set - set	Mutable, no ordenado	(0.003, False, "set")
Diccionario - dict	Mutable, no ordenado	{"nombre" = "Ana", "Apellido" = "Campos"}

Listas

"Las listas en Python son uno de los tipos o estructuras de datos más versátiles del lenguaje, ya que permiten almacenar un conjunto arbitrario de datos. Es decir, podemos guardar en ellas prácticamente lo que sea."

Una lista sea crea con **[ ]** separando sus elementos con comas **","**.

```
lista1 = [ 1, 2, -3, "cuatro"  ]

animales = [ "delfín", "murciélago", "elefante" ]
```

Se puede acceder a sus elementos mediante un índice **[i]**.

```
animales = [ "delfín", "murciélago", "elefante" ]
animales[0]
'delfín'
```

```
frutas = [ "manzana", "banano", "sandía" ]
frutas[2]
'sandía'
```

Listas anidadas

Una de las características de las listas, es que se pueden anidar. Esto quiere decir que uno o varios de los valores dentro de una lista, puede ser otra lista. Una propiedad muy útil a la hora de crear estructuras tipo matrices.

```
lista1 = [ [1, 2, 3],
           [4, 5, 6],
           [7, 8, 9] ]
```

lista1

1	2	3
4	5	6
7	8	9

Los valores dentro de una matriz se pueden acceder de forma **[i][j]**. En el cual **i** representa el valor de la fila y **j** representa el valor de la columna.

```
lista1 = [ [1, 2, 3],
           [4, 5, 6],
           [7, 8, 9] ]

lista1[1][2]
```

```
6
```

Diccionarios

Los diccionarios en Python son una estructura de datos que permite almacenar su contenido en forma de llave y valor. Los diccionarios se pueden crear con paréntesis {} separando con una coma cada par **key: value**.

```
p1 = { "nombre" : "Ana",
       "apellido" : "Campos",
       "edad" : 23,
       "mascotas" : ["michi", "princesa"] }
```

p1				
Llave	Nombre	Apellido	Edad	Mascotas
Valor	Ana	Campos	23	Michi
				Princesa

Características de los diccionarios

- Son **dinámicos**, pueden crecer o decrecer, se pueden añadir o eliminar elementos.

- Son **indexados**, los elementos del diccionario son accesibles a través del *key*.

```
p1["nombre"]
```

```
'Ana'
```

```
p1["mascotas"]
```

```
['michi', 'princesa']
```

- Y son **anidados**, un diccionario puede contener a otro diccionario en su campo *value*.

Ciclo For

- Iteración a través de una estructura.
- Se establece una número limitado de iteraciones.
- Se itera por una estructura ordenada.
- Se debe tomar en cuenta la indentación.
- Tiene la forma:

for _ in estructura:

```
sum = 0
for i in range(10):
    sum = sum + i
    print sum
```

```
: #iterate through a list
  colors = ["red","green","yellow","black"]
  for x in colors:
      print(x)

  #iterate through a string
  s = "red"
  for x in s:
      print(x)
```

```
red
green
yellow
black
r
e
d
```

Ciclo While

```
while i != 5:
  i+=1
```

- Iteraciones mientras la condición se cumpla.
- Se debe tomar en cuenta la indentación.
- Tiene la forma:

while condición:

Continue: Comienza la siguiente iteración.

Break: Sale del ciclo.

```
i = 0

while i < 10:
  i+=1

  if i == 3:
    continue

  if i == 6:
    break

  print("iteración: ", i)
```

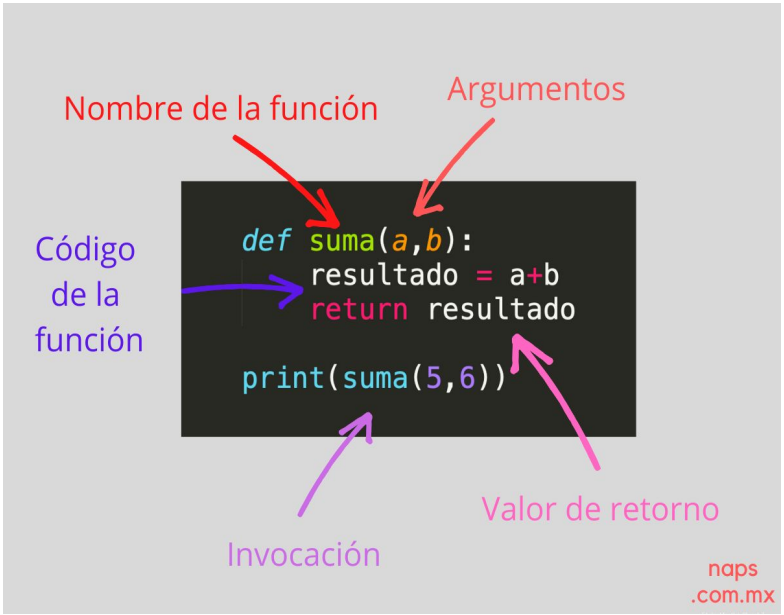
```
iteración:  1
iteración:  2
iteración:  4
iteración:  5
```

Funciones

- Se definen con la palabra *def* de forma:

  def [nombre_funcion](arg1…):

- Se retorna con la palabra *return*.
- Se debe tomar en cuenta la indentación.
- Pueden retornar múltiples valores.
- No son tipadas.



Bibliografía

- Beginner's Guide (2022). Recuperado de https://wiki.python.org/moin/BeginnersGuide/Overview
- Learning Python 3 (2019). Recuperado de https://gist.github.com/kenjyco/69eeb503125035f21a9d
- Funciones en Python: argumentos y retorno de múltiples valores (2020). Recuperado de https://naps.com.mx/blog/funciones-en-python-argumentos-y-retorno/
- El Libro De Python. Listas en Python (2023). Recuperado de https://ellibrodepython.com/listas-en-python
- El Libro De Python. Diccionarios en Python (2023). Recuperado de https://ellibrodepython.com/diccionarios-en-python