



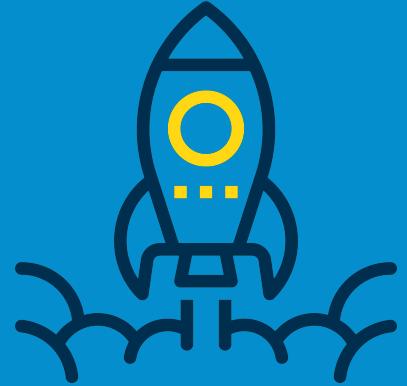
Curso Básico de Python

Facundo García Martoni

```
31     def __init__(self, path=None, debug=False):
32         self.file = None
33         self.fingerprints = set()
34         self.logduplicates = True
35         self.debug = debug
36         self.logger = logging.getLogger(__name__)
37         if path:
38             self.file = open(os.path.join(settings['LOG_DIR'], 'request.log'), 'a')
39             self.file.seek(0)
40             self.fingerprints.update(fp.read().splitlines())
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('SUPERVISED_DEBUG')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```

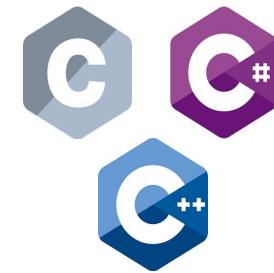






¡Vamos!

¿Por qué Python?



Frontend

IOT

Inteligencia Artificial

Backend

DevOps

Data Science

Videojuegos

Desarrollo móvil

Frontend

IOT

Inteligencia Artificial

Backend

DevOps

Data Science

Videojuegos

Desarrollo móvil



Google



NETFLIX

Uber

Ventajas

- Fácil
- Elegante
- Buenas prácticas





Desafío

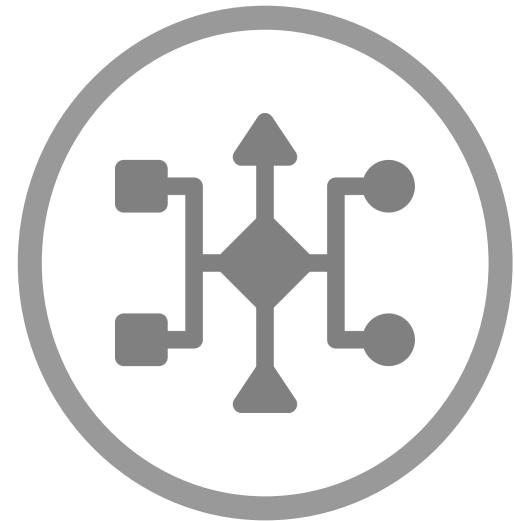
Algoritmo

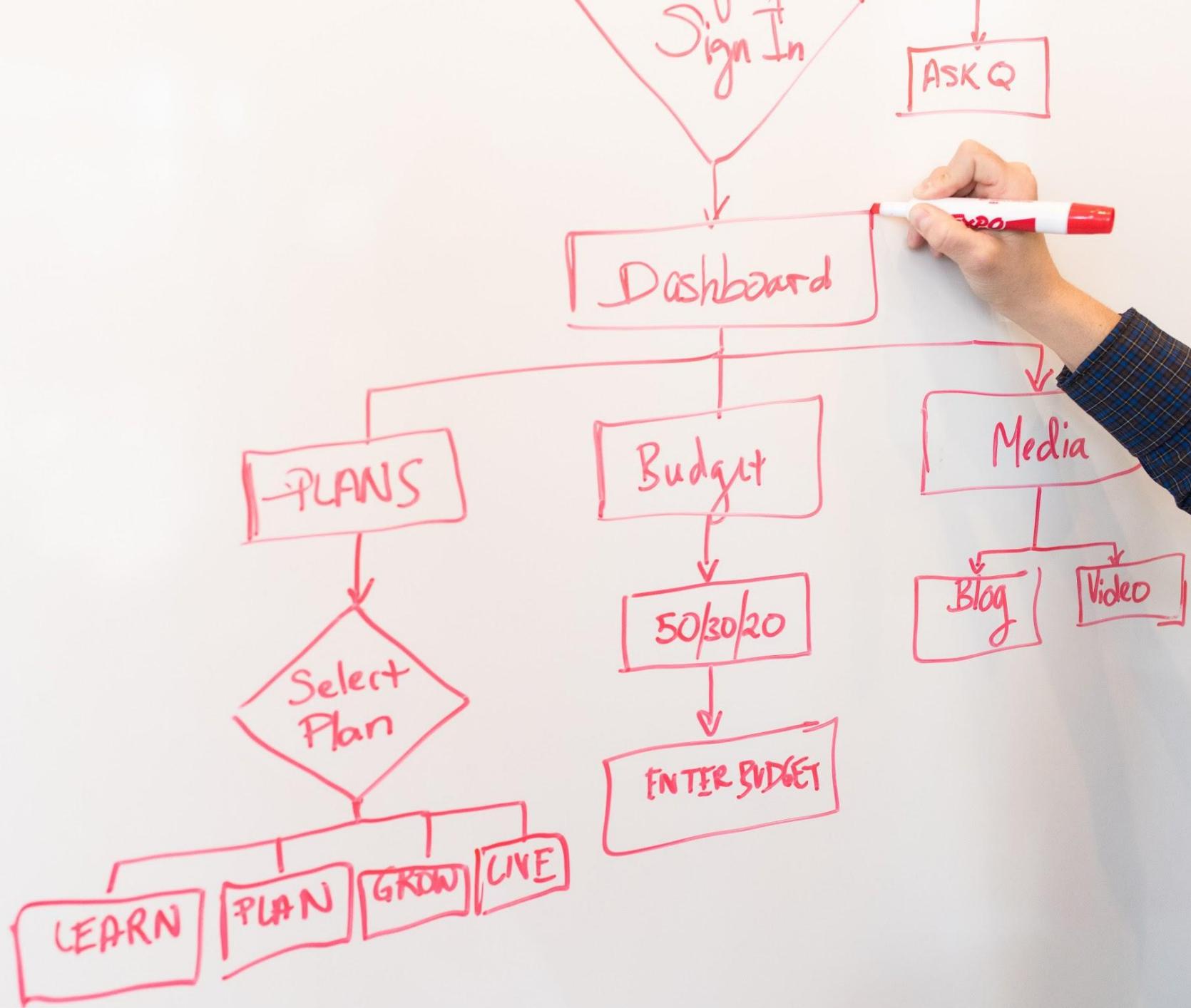




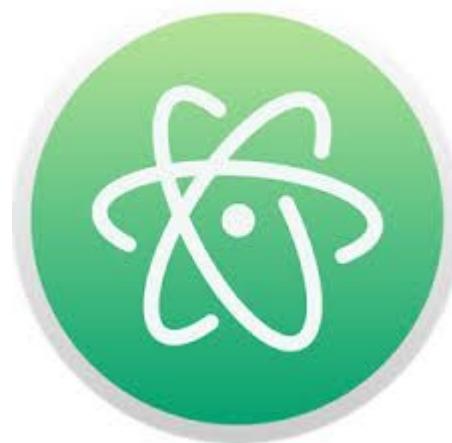
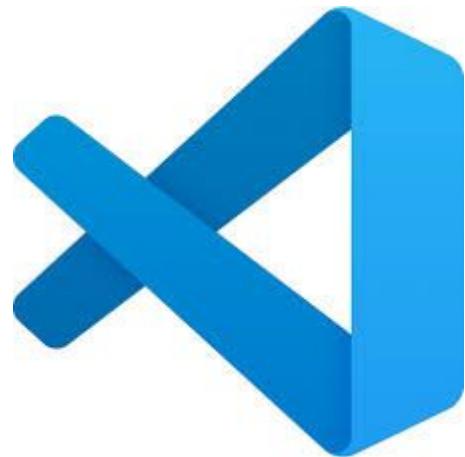
¿Qué es un algoritmo?

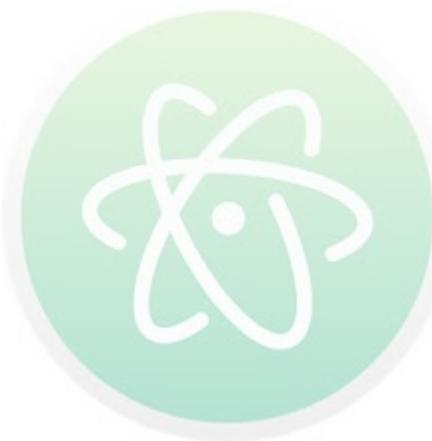
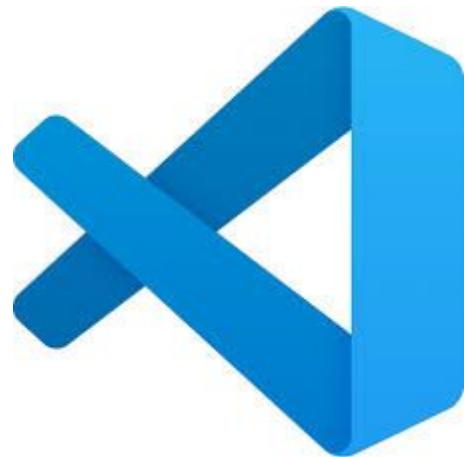
- Serie de pasos ordenados para resolver un problema
- Finito
- No ambiguo

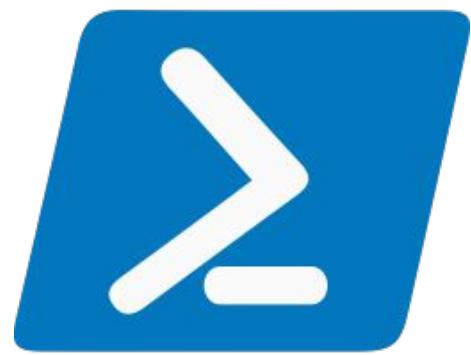


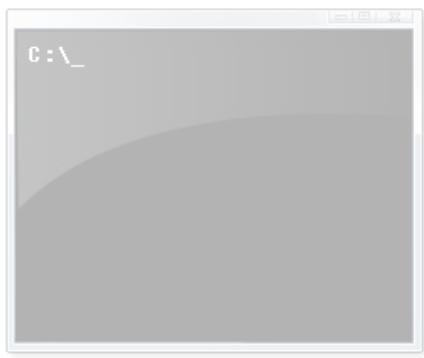


Instalación de nuestras herramientas

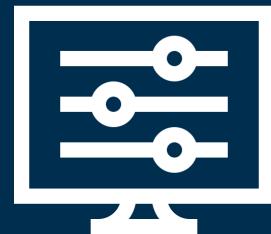








Tu mejor herramienta: la consola



¡En síntesis!

- El comando `cd` sirve para moverse de carpeta en carpeta
- El comando `ls` muestra el contenido de una carpeta
- El comando `mkdir` crea carpetas
- El comando `touch` sirve para crear archivos

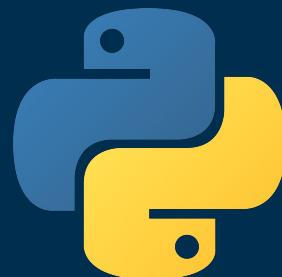
Explorando Python: operadores aritméticos



¡En síntesis!

- `+` → suma
- `-` → resta
- `*` → multiplicación
- `/` → división
- `//` → división euclíadiana (cociente)
- `%` → módulo (resto de la división)
- `**` → potencia

¿Qué es una variable?





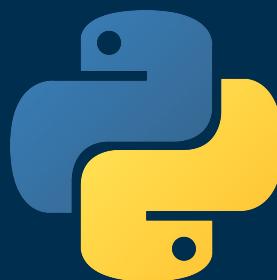




¡En síntesis!

- Una **variable** es una “caja” o lugar en donde puedo guardar objetos: números, texto, etc.
- El **identificador** de mi variable no puede comenzar con un número y debe estar en minúsculas. Las palabras dentro del mismo se separan con guión bajo

Los primitivos: tipos de datos sencillos



Números
enteros

Números
de punto
flotante

Texto
(cadenas de
caracteres)

Booleanos
(verdadero
y falso)

¡En síntesis!

- `int`: 5, 6, 1, 3, -6, -10
- `float`: 1.33, 4.5, 6.2
- `bool`: True, False
- `str`: “Facundo”, “María José”

Convertir un dato a un tipo diferente



¡En síntesis!

- El **casting** es la técnica que sirve para convertir un dato de un tipo a un tipo diferente. Ejemplos:
- **int** a **str**: str(45)
- **str** a **int**: int("123")
- **float** a **int**: int(3.5)

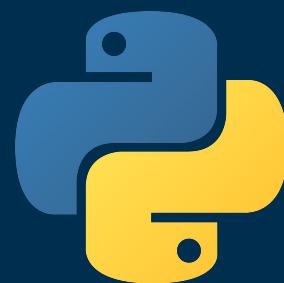
Operadores lógicos y de comparación



¡En síntesis!

- Una **expresión** es la combinación de variables, datos y operadores, de la cual se obtiene un valor
- Operadores lógicos: **and**, **or**, **not**
- Operadores de comparación: **==**, **!=**, **<**, **>**,
>=, **<=**

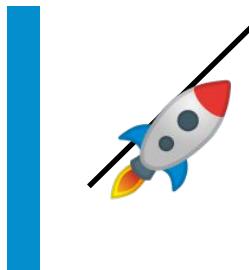
Tu primer programa: conversor de monedas



Pesos
colombianos



Dólares



Reto

- Haz el mismo programa, pero con la moneda de tu país
- Crea un programa que hace lo inverso: pasa una cantidad de dólares a la moneda de tu país



Construyendo el camino de un programa con condicionales





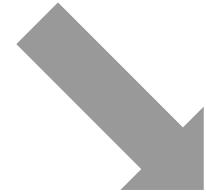
¡En síntesis!

- Las palabras clave `if`, `elif` y `else` te permiten dirigir el camino por el que avanza tu programa dependiendo de una o varias condiciones
- Luego de los dos puntos (`:`), dejamos cuatro espacios de sangría en la siguiente línea

Varios países en mi conversor de monedas



Pesos argentinos

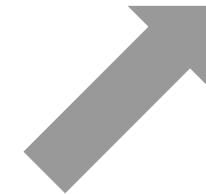


Pesos
colombianos



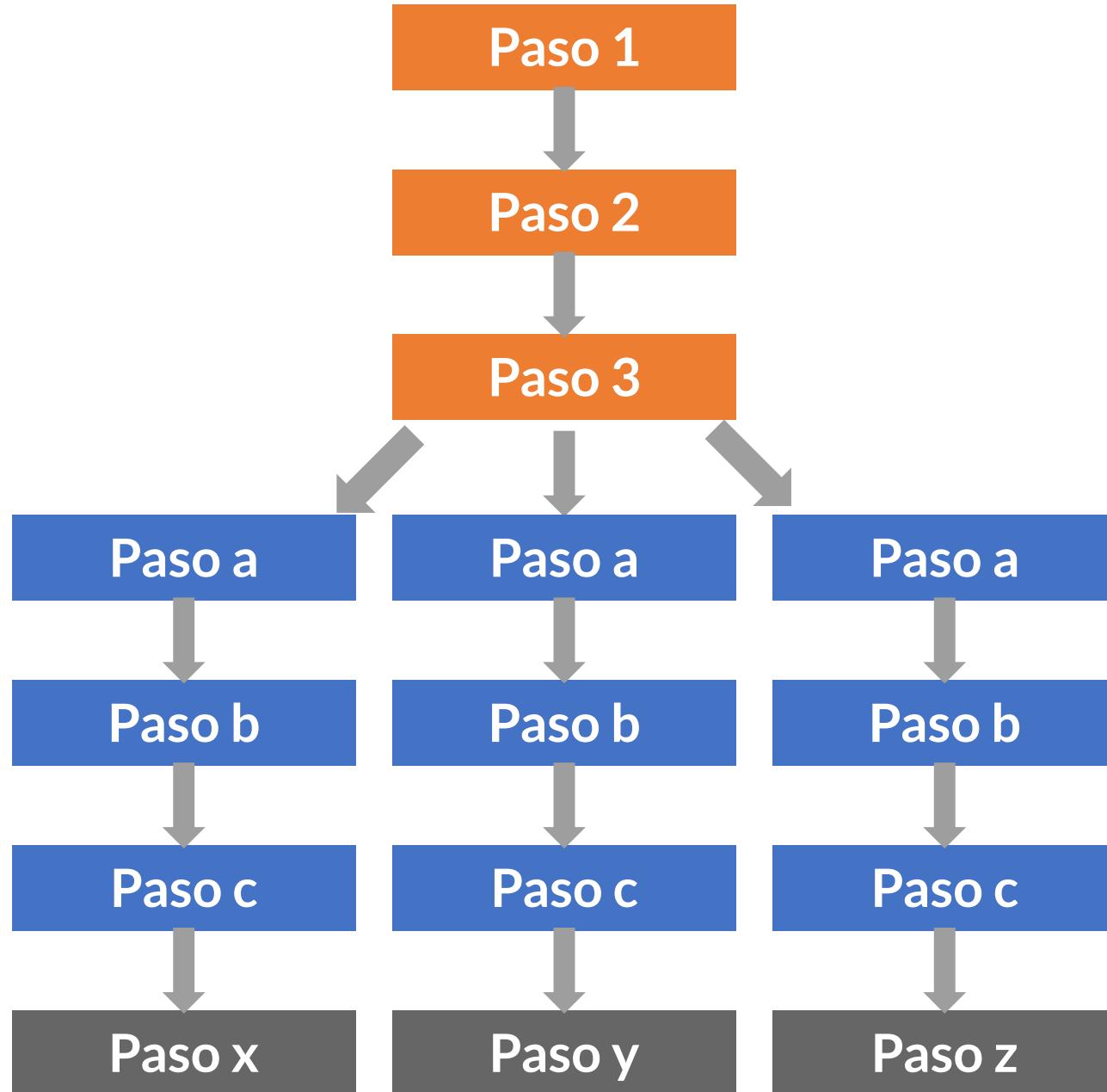
Dólares

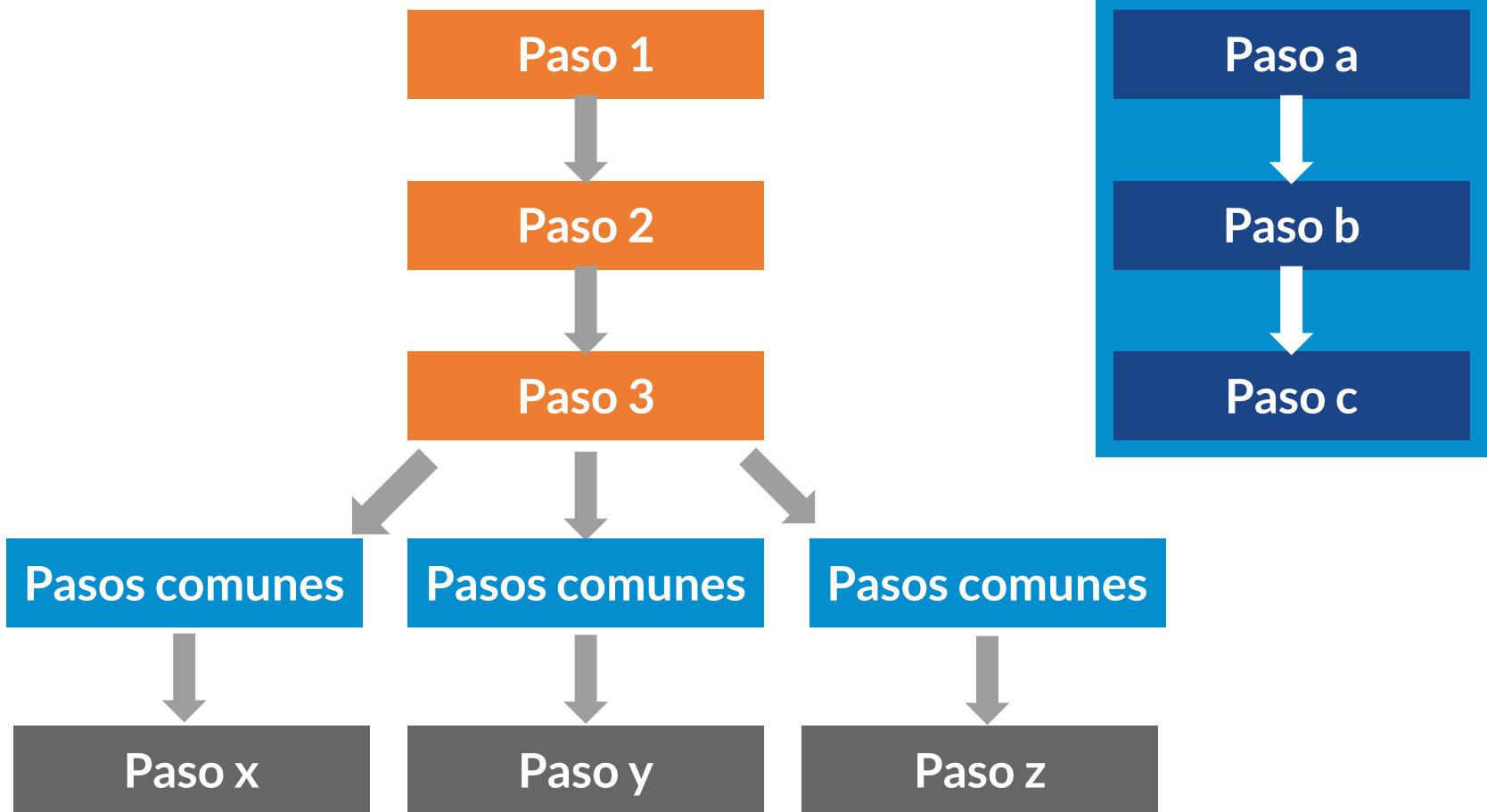
Pesos mexicanos



Aprendiendo a no repetir código con funciones







¡En síntesis!

- Una **función** es una pieza de código que se puede **invocar** varias veces. Se definen con la palabra clave **def**
- Las funciones pueden recibir opcionalmente **parámetros**, que son, en palabras simples, los datos que cambian en cada **llamada** o **invocación**

Modularizando nuestro conversor de monedas



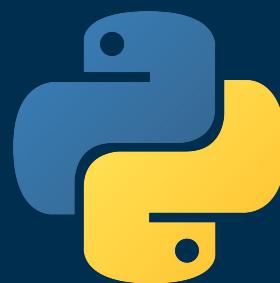
Trabajando con texto: cadenas de caracteres



¡En síntesis!

- Para **concatenar** cadenas de caracteres debo sumarlas con el operador **+**
- Si multiplico una cadena por un entero **n** con el operador *****, esta **se repite n veces**
- El método **strip** elimina espacios basura al inicio y al final de una cadena
- El método **replace** reemplaza una subcadena por otra

Trabajando con texto: modificando las cadenas



¡En síntesis!

- El método `upper` convierte a mayúsculas una cadena
- El método `lower` convierte a minúsculas una cadena
- El método `capitalize` coloca la primera letra de una cadena en mayúsculas
- La función `len` devuelve la longitud de una cadena
- Los `índices` permiten acceder a caracteres.
Se colocan entre `[]`

Trabajando con texto: slices





Platzi

Pla|tzi

P latzi

¡En síntesis!

- Las *slices* o rebanadas permiten acceder a subcadenas de una cadena original
- Se colocan entre corchetes con la siguiente sintaxis: [índice_inicial:índice_final:pasos]
- Tanto los índices como los pasos son opcionales

Proyecto: palíndromo



Luz azul

¡En síntesis!

- Un **palíndromo** es una palabra o frase que puede leerse igual al derecho y al revés.

Ejemplos:

- No deseo ese don
- Yo hago yoga hoy
- Luz azul

Creando nuestra función palíndromo



¡En síntesis!

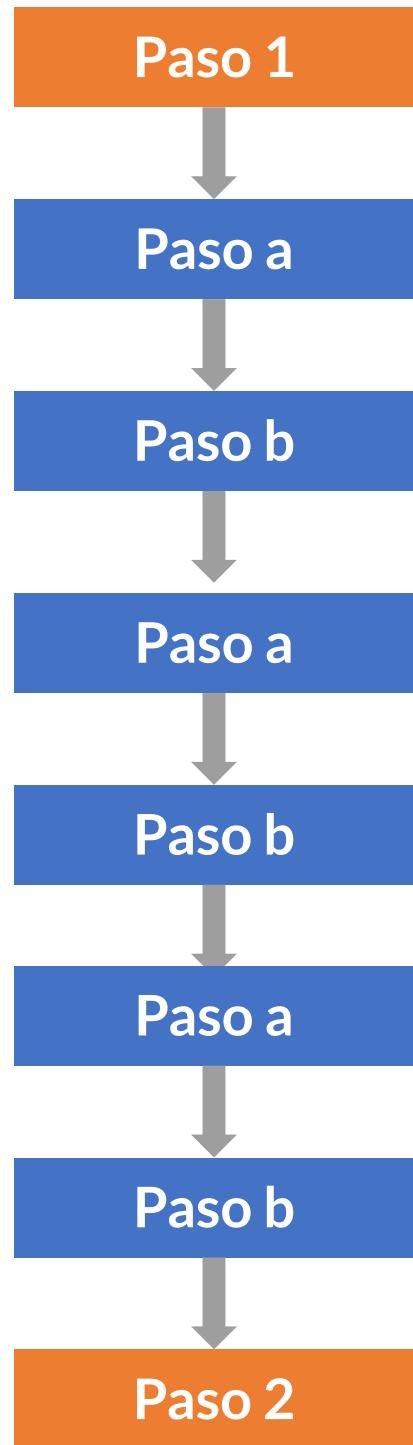
- Un **palíndromo** es una palabra o frase que puede leerse igual al derecho y al revés.

Ejemplos:

- No deseo ese don
- Yo hago yoga hoy
- Luz azul

Aprendiendo a ejecutar algo varias veces con bucles





Paso 1



Repetir: 3 veces

Paso a



Paso b



Paso 2

¡En síntesis!

- Un **bucle** o **ciclo** es una herramienta de la programación que permite repetir la ejecución de un bloque de código una cantidad determinada de veces

El ciclo while



¡En síntesis!

- El ciclo `while` permite ejecutar un bloque de código mientras una condición se cumpla. La sintaxis es la siguiente:

`while` condición:

código

Explorando un bucle diferente: el ciclo for



¡En síntesis!

- El ciclo **for** permite ejecutar un bloque de código mientras se recorre un rango determinado, teniendo disponible en cada vuelta del ciclo a un elemento de ese rango. La sintaxis es la siguiente:

for elemento **in** rango:

código

Recorriendo un string con for



¡En síntesis!

- El ciclo **for** permite operar caracter por caracter en una determinada cadena. Por ejemplo, para la cadena “Platzi”, yo podría acceder a cada letra en cada vuelta del bucle. Sintaxis:

for caracter **in** cadena:

código

Interrumpiendo ciclos con break y continue





Desafío

¡En síntesis!

- La palabra clave `break` termina la ejecución de un ciclo
- La palabra clave `continue` permite pasar a la siguiente vuelta del ciclo sin tener en cuenta el código que falta por ejecutar en la vuelta actual

Proyecto: prueba de primalidad



1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

¡En síntesis!

- Un **número primo** es aquel que solo puede dividirse de forma exacta por sí mismo y por 1
- Ejemplos: 1, 13, 19, 41, 43, 97
- **Reto:** investiga diferentes formas de averiguar si un número es primo o no. Intenta programarlas, y deja tu código en el sistema de discusiones

Proyecto: videojuego



¡En síntesis!

- El módulo `random` permite trabajar con aleatoriedad en Python
- La función `randint` devuelve un número entero aleatorio en un determinado rango

Almacenar varios valores en una variable: listas

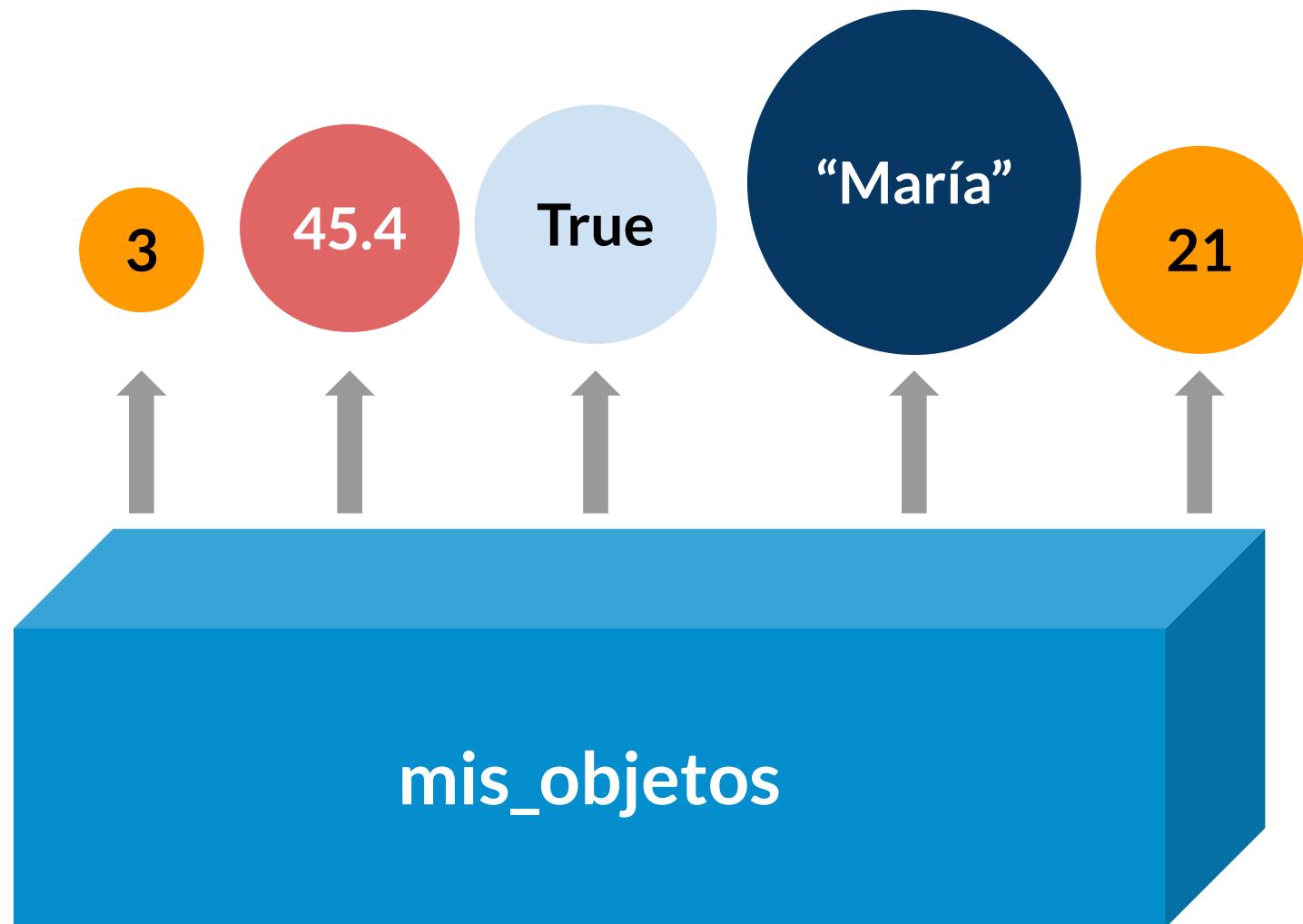


14



mi_numero





¡En síntesis!

- Una **lista** es una estructura de datos que permite almacenar varios valores, que pueden tener o no el mismo tipo, en una variable
- Al igual que con los strings, se utilizan **índices** y **slices** para acceder a los elementos
- El método **append** permite agregar un elemento. El método **pop** permite borrar un elemento

Entendiendo cómo funcionan las tuplas



¡En síntesis!

- Una **tupla** funciona igual que una lista, pero es **inmutable**: no puedo agregar ni borrar valores una vez creada
- Operar con tuplas es **más rápido** que operar con listas

¿Qué son los diccionarios?



¡En síntesis!

- Un **diccionario** es una estructura de datos formada por pares **llave:valor** que permite almacenar varios tipos de valores de una forma organizada
- Puedo recorrer un diccionario con el ciclo `for`: el método **keys** devuelve las llaves, el método **values** devuelve los valores y el método **items** devuelve tanto llaves como valores

Proyecto: generador de contraseñas



¡En síntesis!

- Dentro del módulo random, la función `choice` permite elegir un elemento aleatorio de una lista
- El método `join` permite transformar una lista en un string

Sigue aprendiendo



¿Quieres preguntarme algo?



@facmartoni

facundonicolas.com