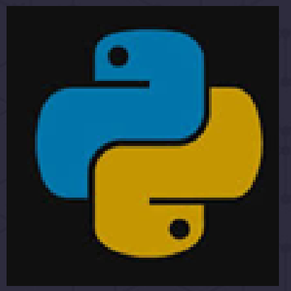
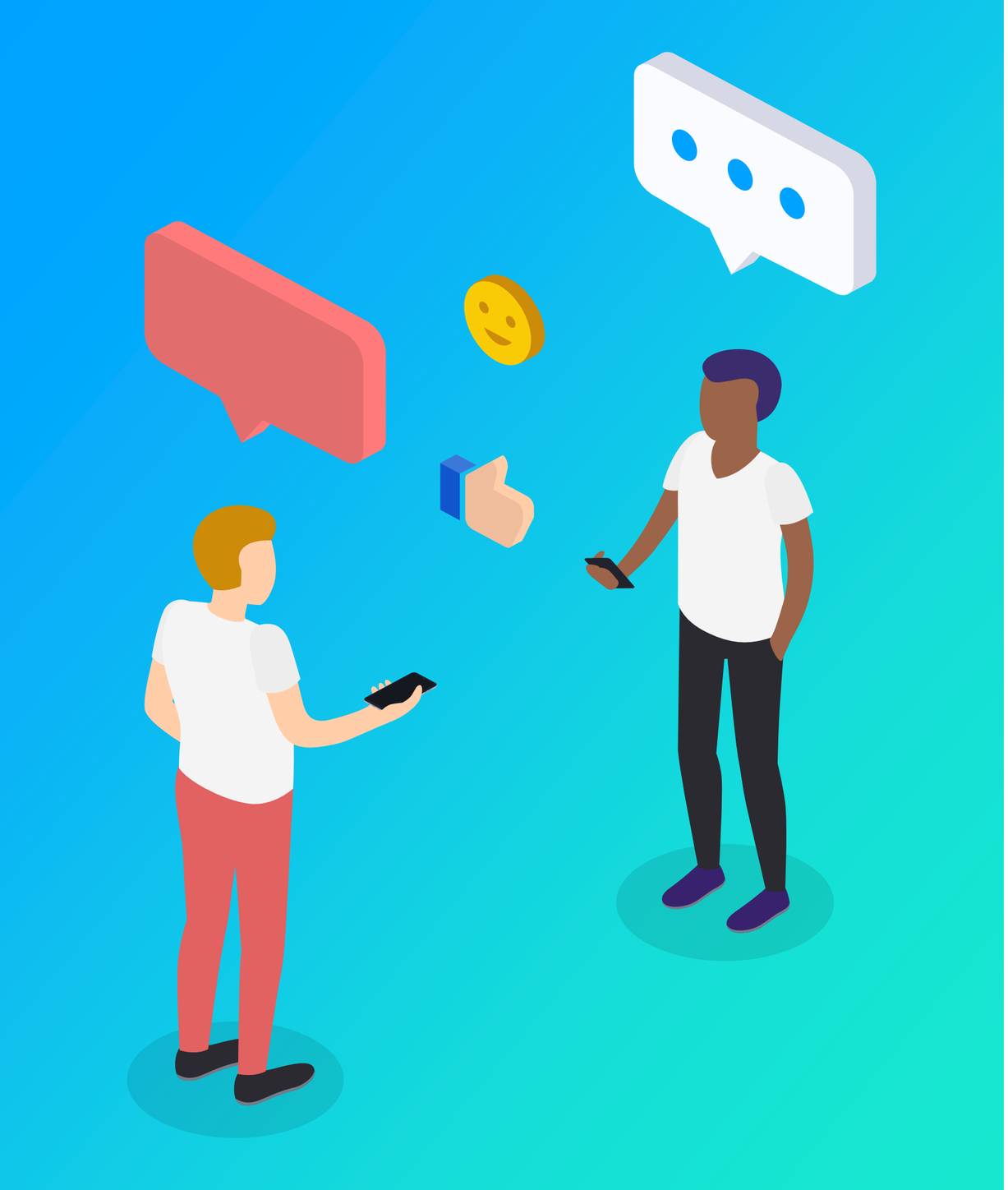
PREGRADO

UNIDAD 1 | SEMANA 1

EL LENGUAJE PYTHON



1ACC0201| Programación Orientada a Objetos



Al finalizar la Unidad, el estudiante entiende los conceptos básicos del lenguaje de programación Python

AGENDA

La filosofía Python

Conceptos básicos:

- Tipos de datos
- Variables
- Función print()
- Operadores matemáticos
- Conversiones entre tipo de datos
- Función input()
- Excepciones

¿QUÉ ES PYTHON?

- Es un lenguaje de programación interpretado cuya filosofía hace hincapié en la **legibilidad de su código.**

- La historia del lenguaje de programación Python se remonta hacia finales de los 80s y principio de los 90s.

- Su implementación comenzó en diciembre de 1989 cuando en Navidad **Guido Van Rossum** decide darle continuar al proyecto “ABC” que se encontraba en laboratorio.

El Zen Python

- **La Filosofía Python**
  - A la hora de programar, existen una serie de principios que hacen especial a Python. Se dice que el código que siga el **Zen Python** (llamados los principios de Python, escrito por uno de sus desarrolladores: Tim Peters) es **“pythonico”.**
  - El Zen de Python de Tim Peters son 20 pautas para el diseño del lenguaje Python.
  - Su código Python no necesariamente tiene que seguir estas pautas, pero es bueno tenerlas en cuenta
  - Sin embargo, sólo 19 de las pautas están escritas. Según Guido van Rosum el número 20 que falta es "una extraña broma interna de Tim Peters".
  - Los 19 principios son los siguientes:

El Zen Python

- El Zen de Python, de Tim Peters

1. Bello es mejor que feo.
2. Explícito es mejor que implícito.
3. Simple es mejor que complejo.
4. Complejo es mejor que complicado.
5. Plano es mejor que anidado.
6. Disperso es mejor que denso.
7. La legibilidad cuenta.
8. Los casos especiales no son tan especiales como para quebrantar las reglas.
9. Lo práctico gana a lo puro.
10. Los errores nunca deberían dejarse pasar silenciosamente.
11. A menos que hayan sido silenciados explícitamente.
12. Frente a la ambigüedad, rechaza la tentación de adivinar.

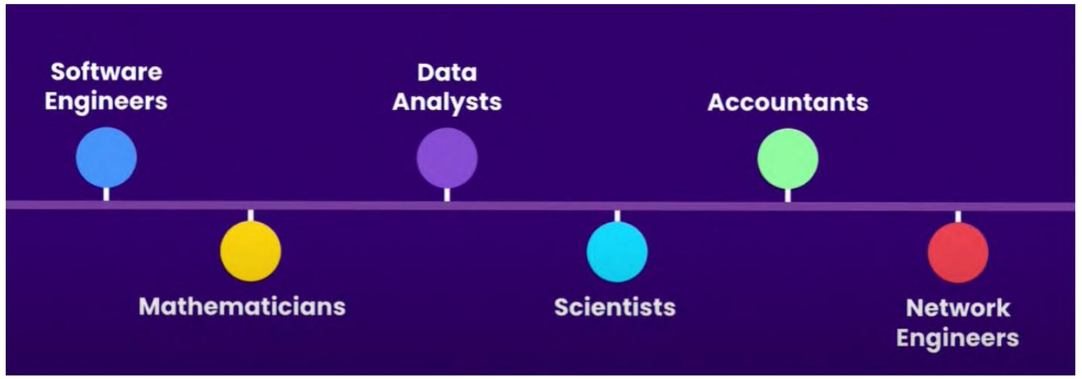
El Zen Python

13. Debería haber una —y preferiblemente sólo una— manera obvia de hacerlo.
14. Aunque esa manera puede no ser obvia al principio, a menos que usted sea holandés.
15. Ahora es mejor que nunca.
16. Aunque nunca es a menudo mejor que ya mismo.
17. Si la implementación es difícil de explicar, es una mala idea.
18. Si la implementación es fácil de explicar, puede que sea una buena idea.
19. Los espacios de nombres (namespaces) son una gran idea ¡Hagamos más de esas cosas!

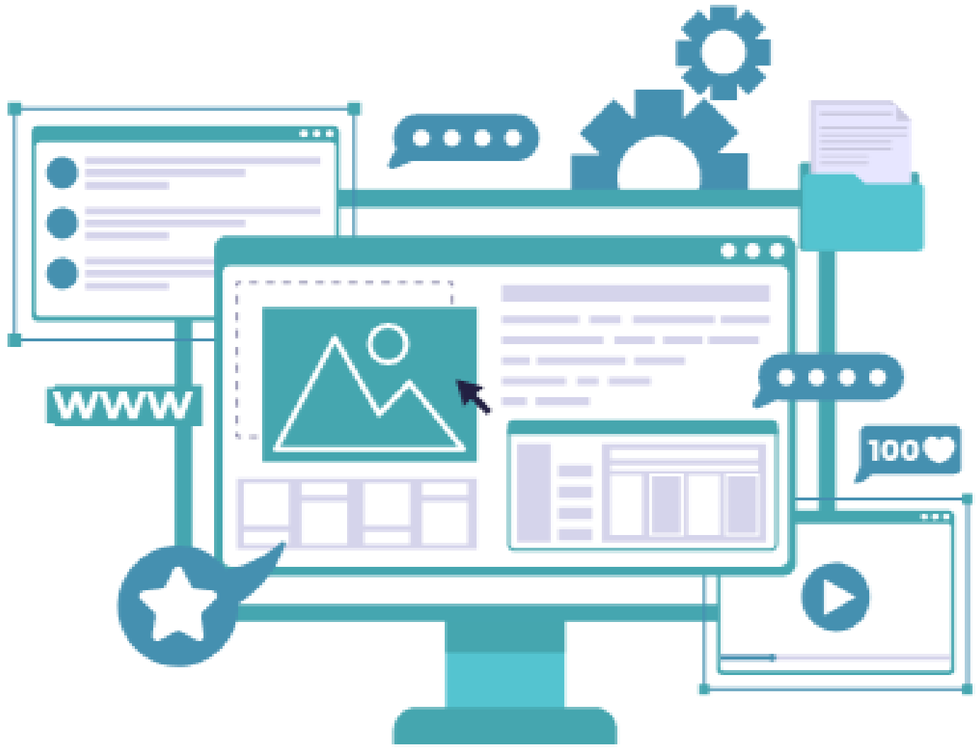
Para una mayor explicación de estos principios puede consultar los siguientes enlaces
https://platzi.com/clases/2255-python-intermedio/36456-el-zen-de-python/
https://www.youtube.com/watch?v=QfbLtQ8Kiv4
https://www.youtube.com/watch?v=l5hr19b2o98
https://www.youtube.com/watch?v=7yhyWCrcxQA

¿Por qué es tan popular Python y por qué aprenderlo?

Python es muy popular debido a su uso en diferentes disciplinas y su fácil aprendizaje.



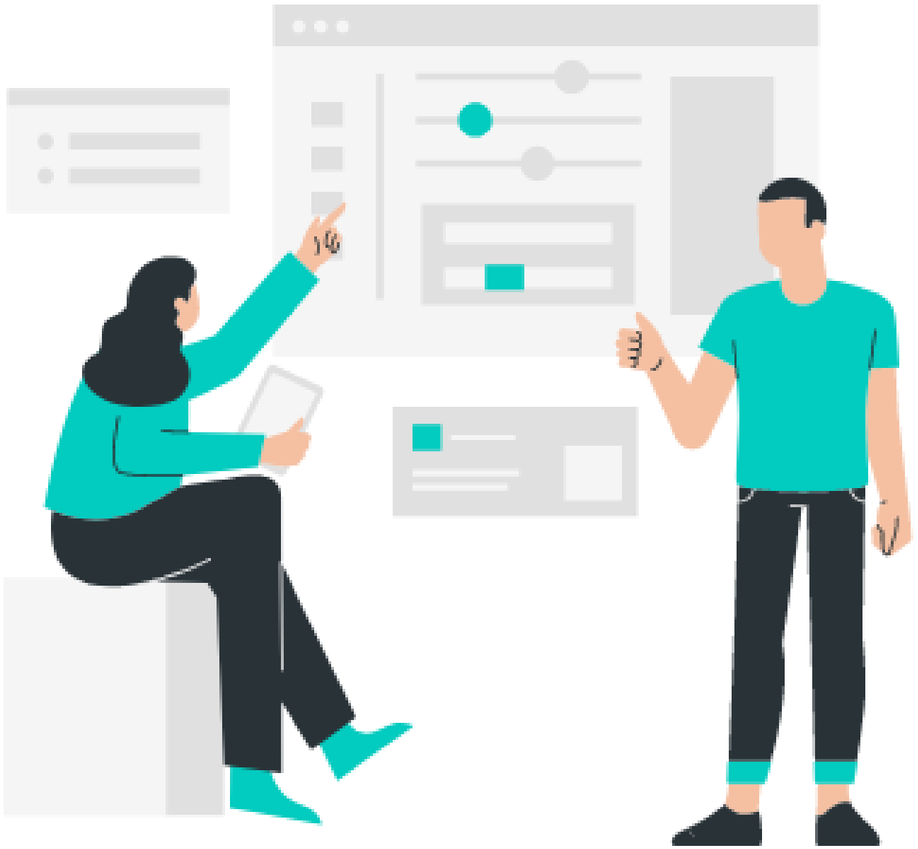
¿QUÉ EMPRESAS USAN PYTHON?



Características de Python

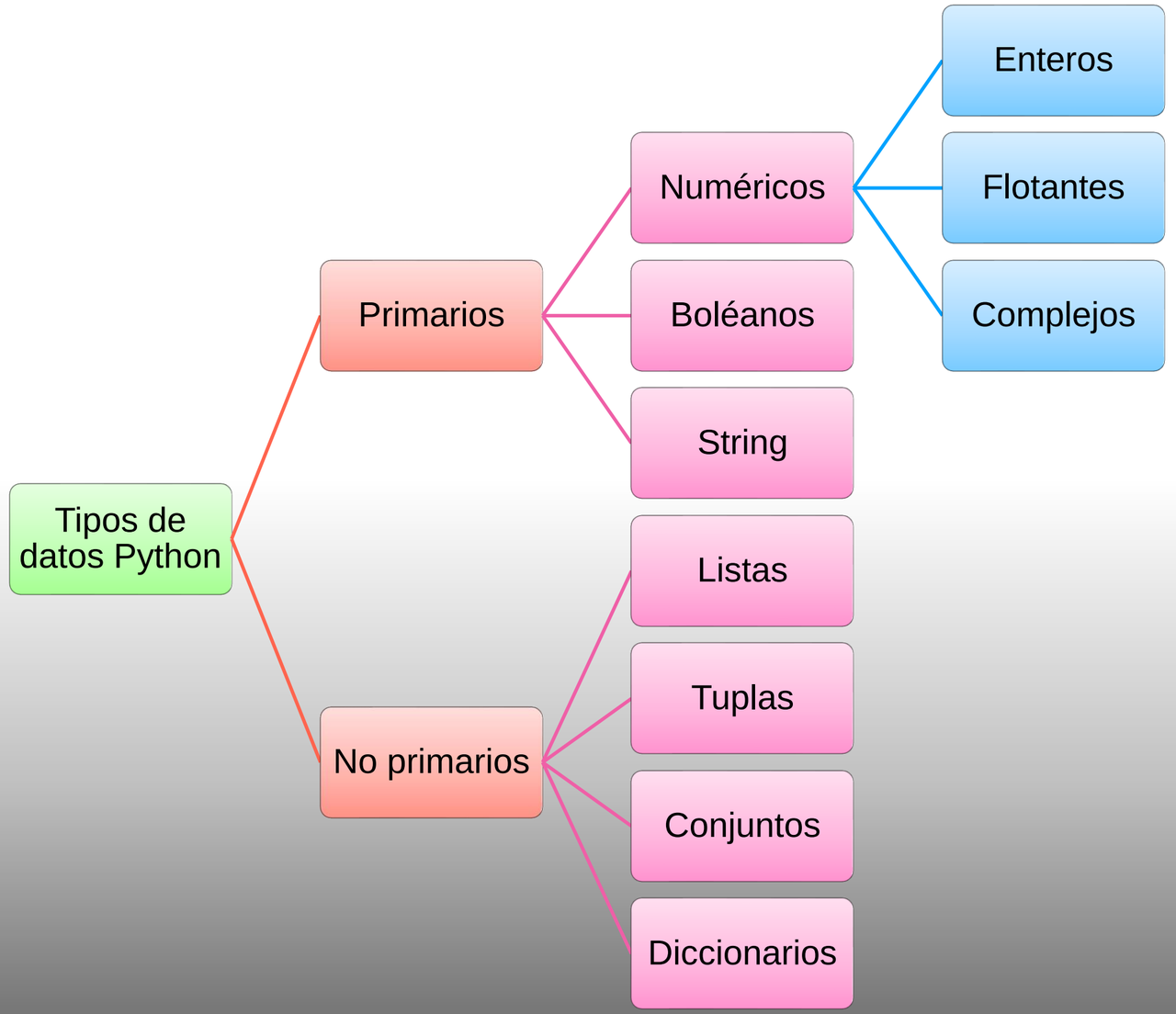
- **Orientado a objetos**
  - El lenguaje Python de programación está **orientado a objetos**. Con Python **todo es un objeto** que cuenta con sus propiedades. Ajustando las propiedades y relacionando el comportamiento entre objetos se puede crear cualquier tipo de programa en Python.
- **Tipado dinámico**
  - **Con Python no hace falta declarar las variables** al principio, ya que es un lenguaje de tipado dinámico. La variable se declara en el mismo momento en el que va a ser utilizada, con el objetivo de facilitar la creación de software.

Características de Python



- **Llamadas a librerías**
  - En Python existen infinidad de **librerías** que pueden ser llamadas y que contienen funciones y tareas ya programadas que facilitan la codificación y el desarrollo de aplicaciones. Gracias a estas librerías no hay que perder tiempo en codificar muchos aspectos de un programa pues ya se encuentra desarrollado en la propia librería, y solo es necesario realizar la llamada correspondiente y ajustar los atributos necesarios para utilizarlos.
- **Interpretado**
  - Al desarrollar en Python no es necesario compilar el programa antes de ejecutarlo para ver su funcionamiento, Al ser interpretado, un programa externo permite ejecutarlo de forma automática.

Tipos de datos Python



Tipos de dato

Nombre	Tipo	Ejemplos
Booleano	bool	True, False, 1 (se interpreta como True), 0 (se interpreta como False)
Entero	int	21, 34500
Flotante	float	3.14, 1.5e3
Complejo	complex	2j, 3 + 5j
Cadena	string	“OOP”, ‘Lenguaje Python’, ‘’’Tipos de datos Python’’’
Tupla	tuple	(1, 3, 5)
Lista	list	[“C++”, “Python”]
Conjunto	set	{2, 4, 6}
Diccionario	dict	{“Chrome”: “v79”, “Firefox”: “v71”}

Variables

- A diferencia de otros lenguajes, **no es necesario especificar el tipo de dato** que almacenará
- A través de la asignación de valores, Python determina el tipo de variable primario
  - `n = 5` `(entero)`
  - `pi = 3.14159` `(punto flotante)`
  - `curso = “Programación Orientada a Objetos”` `(string)`
  - `z = True` `(booleano)`
- Las variables son **“case sensitive”,** es decir la variable **nota** es diferente de la variable **Nota**
- Los nombres de las variables siguen las mismas reglas que otros lenguajes como C++
  - Debe comenzar con una letra o un guion bajo _
  - Debe consistir en letras, números y guiones bajos
  - No puede tener caracteres especiales como por ejemplo /*+-

La función print()

Sintaxis: print(objetos, sep=“ “, end=“\n”)

- La función print() permite poner diferentes parámetros o ninguno
- print() (sin parámetros), da como resultado una línea en blanco
- Se puede pasar una cadena directamente a print

```
print("Por favor, espere mientras se carga el programa...")
```

Por favor, espere mientras se carga el programa...

- Se puede usar una variable

```
mensaje = "Por favor, espere mientras se carga el programa...“
print(mensaje)
```

Por favor, espere mientras se carga el programa...

La función print()

Sintaxis: print(objetos, sep=“ “, end=“\n”)

- Se puede imprimir números y variables numéricas

```
print(3.14)
x = 45
print(x)
```

3.14
45

```
s1 = "Curso“
s2 = "Programación Orientada a Objetos“
print(s1, s2)
```

Curso Programación Orientada a Objetos

Caracteres de escape

A continuación, se muestran los principales carácteres de escape y su significado

Carácter	Significado
\n	Nueva línea
\'	Insertar apóstrofes o comillas simples
\"	Insertar comillas dobles
\\	Poner un backslash en el texto
\t	Poner tab en el texto

La función print(): Usando end y sep

Analizar los resultados de los códigos a continuación y debatir sobre las variaciones encontradas.

```
print("Nombre:")
print("Luis Alberto")
print("Apellido:")
print("Rodriguez Vargas")
```

```
print("Nombre:", end=" ")
print("Luis Alberto")
print("Apellido:", end=" ")
print("Rodriguez Vargas")
```

La función print(): Usando end y sep

Ejecuta las siguientes instrucciones y analiza los resultados obtenidos.

```
print('Mercurio', 'Venus', 'Tierra', sep=', ', end=', ')
print('Marte', 'Jupiter', 'Saturno', sep=', ', end=', ')
print('Urano', 'Neptuno', 'Pluton', sep=', ')
```

```
print('Impresión en pocas palabras', end='\n * ')
print('Llamar a Imprimir', end='\n * ')
print('Separar varios argumentos', end='\n * ')
print('Prevención de saltos de línea')
```

Operadores matemáticos en Python

Operador	Descripción	Ejemplo	Resultado	Prioridad
**	Exponenciación	2**3	8	1
%	Módulo	9 % 4	1	2
//	División entera	9 // 2	4	2
/	División flotante	9 / 2	4.5	2
*	Multiplicación	2 * 8	16	2
+	Suma	5 + 4	9	3
-	Resta	6 - 3	3	3

Formas compactas de operaciones en Python

Operador	Ejemplo	Equivalente
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Conversiones entre tipos de datos

Función	Descripción	Aplica
type(objeto)	Devuelve el tipo de objeto	A cualquier tipo de objeto
Id(objeto)	Devuelve una identificación única para el objeto especificado	A cualquier tipo de objeto
str(objeto)	Convierte el valor especificado en una cadena	Int, float (aplica a otros, pero se verá más adelante)
int(objeto)	Convierte un valor a un entero. Si es un float, elimina la parte decimal, si es un string, lo convierte si el string es un número sin punto decimal	string, float
float	convierte un valor a un punto flotante. Si el valor es un int lo convierte a flotante. Si es un string lo convierte a flotante incluso si hay puntos decimales en el string	string, int

Conversiones entre tipos de datos – ejemplos

```
x = 3.5
print(x)
print(type(x))
x = int(x)
print(type(x))
print(x)
```

3.5

<class 'float'>

<class 'int'>

3

Conversiones entre tipos de datos - ejemplos

```
x = "3"
print(x)
print(type(x))
x = int(x)
print(type(x))
print(x)
```

3.5

<class 'float'>

<class 'int'>

3

Conversiones entre tipos de datos - ejemplos

```
x = "3.5"
print(x)
print(type(x))
x = int(x)
print(type(x))
print(x)
```

3.5

<class 'str’>

line 4

ValueError: invalid literal for int() with base 10: '3.5'

Conversiones entre tipos de datos - ejemplos

```
x = 3
x = float(x)
print(type(x))
print(x)

x = "3.5"
x = float(x)
print(type(x))
print(x)
```

<class 'float'>

3.0

<class 'float'>

3.5

La función input()

- La función input() permite la entrada de datos por parte del usuario.
- Sintaxis: **input(*prompt*)**
- Donde prompt es a string, que representa un mensaje que es mostrado antes de la entrada.
- La función input **siempre recibe los datos como tipo string**, aun cuando lo que se ingresen sean números

```
x = input('Ingresa tu nombre:')
print('Hola, ' + x)

x = input('Ingresa tu edad:')
print(x)
print(type(x))
```

Ingresa tu nombre:Alberto

Hola, Alberto

Ingresa tu edad:19

19

<class 'str'>

La función input()

Para que el dato ingresado por el usuario sea tratado como un número, es necesario transformar el string a int o float utilizando las funciones int() y float().

```
x = float(input('Ingresa tu estatura:'))
print(x)
print(type(x))
```

Ingresa tu estatura:1.85

1.85

<class 'float'>

Manejando excepciones - Bloques try y except

- Hay situaciones, como por ejemplo el ingreso de datos, que pueden producir errores y hacer que el programa colapse
- Por ejemplo, si tenemos el siguiente programa

```
num = int(input("ingrese un número entero "))
print(num)
```

- Si por ejemplo se ingresa el número 3.5 se produciría lo siguiente:

```
Traceback (most recent call last):
line 1
num = int(input("ingrese un número entero "))
          ~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: '3.5'
```

Manejando excepciones - Bloques try y except

- Para prevenir el error, debemos poner el código propenso a error en un bloque try y luego encadenaremos un bloque except para tratar la excepción
- Si modificamos el ejemplo anterior quedaría de la siguiente manera

```
try:
    num = int(input("ingrese un número entero "))
    print(num)
except ValueError:
    print("El número ingresado no es un entero")
```

- Si ahora se ingresara el número 3.5, el resultado sería

```
ingrese un número entero  3.5
El número ingresado no es un entero
```

PREGRADO

Ingeniería de Sistemas de Información

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco

Lima 33 - Perú

T 511 313 3333

https://www.upc.edu.pe

exígete, innova

UPC

LAUREATE
INTERNATIONAL
UNIVERSITIES