



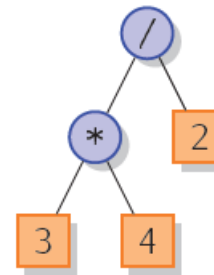
UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD DE
INGENIERÍA

Cursos Básicos Primer Ciclo 2024

Fundamentos de Programación



$3 * 4 / 2$



Clase Pasada

- Programa Almacenado
- Arquitectura de Von Neumann
- Algoritmo
- Lenguaje de Programación
- Expresiones, Variables, Valores
- Operadores y su precedencia

¿Qué veremos hoy?

- Introducción a Python, Historia, IDE
- Primer programa en Python
- Tipos de datos básicos, conversiones
- Variables
- Operadores
- Tipos de Errores
- Entrada/salida
- Cadenas
- Ejercicios

Algoritmos y Lenguajes de Programación

- Seudocódigo
(semi hablado)

Ejemplo: Suma de dos números

Entrada: $a, b \rightarrow$ números

Salida: s , la suma de a y b

inicio

leer (a, b)

$s = a + b$

escribir (s)

fin

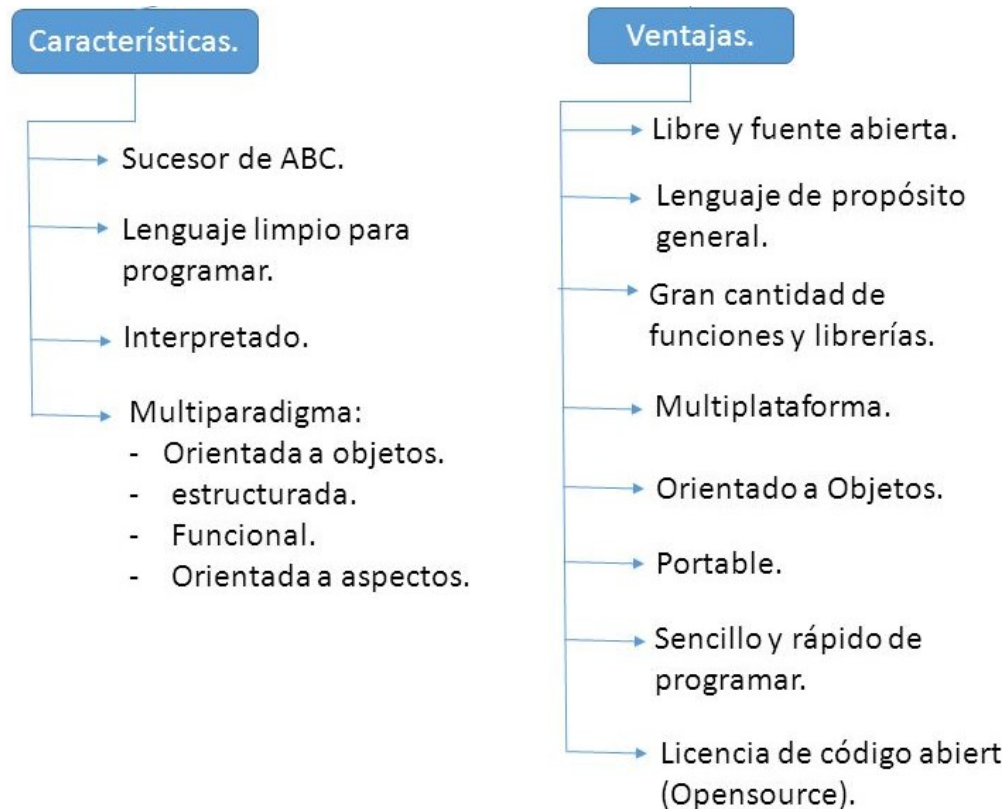
- Lenguaje de Programación
(comunicación con la computadora)

Ejemplo: Suma de dos números en **Python**
Archivo de texto, con extensión .py

```
a=int(input())  
b=int(input())  
print(a+b)
```

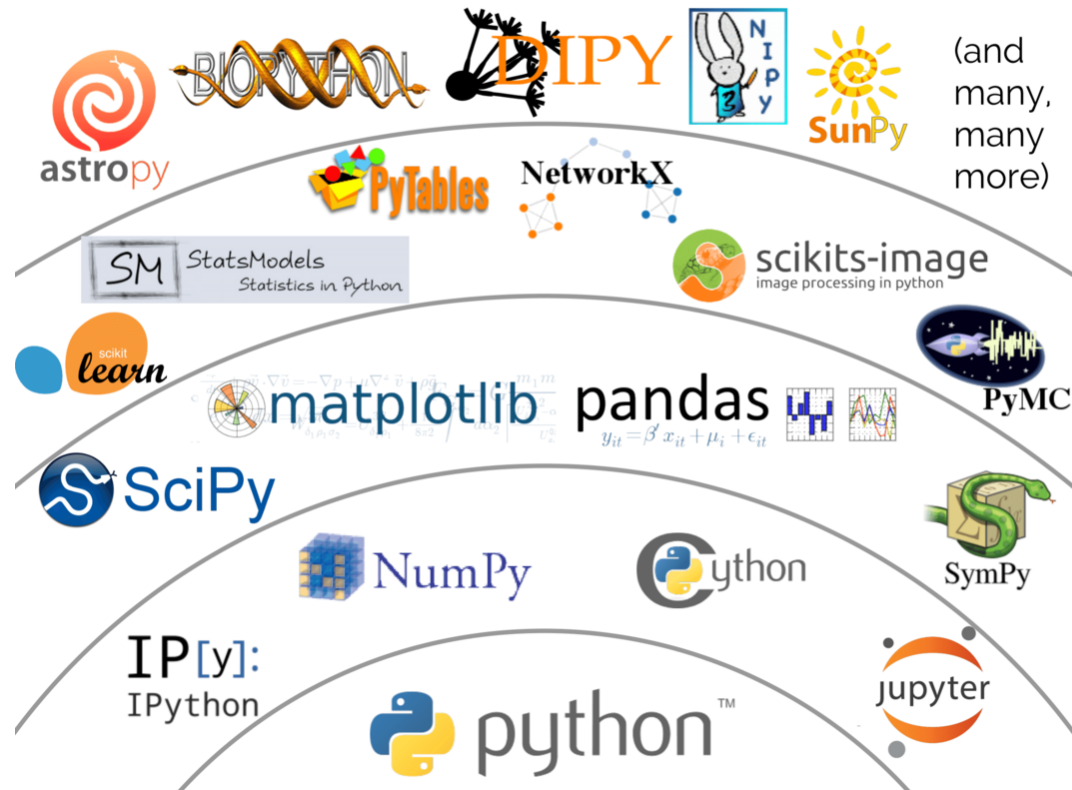
Lenguaje Python

Python 3.12 es la última versión estable del lenguaje de programación Python, con una combinación de cambios en el lenguaje y la biblioteca estándar.



<https://docs.python.org/es/3/library/index.html>

Bibliotecas para aplicación



Existen muchísimos proyectos (como librerías) además de las librerías que son estándar. Actualmente más de 380K proyectos (<https://pypi.org/>) para todas las áreas.

Su historia



Guido Van Rossum, un programador de computación de los Países Bajos, creó Python. Python comenzó en 1989 en el Centrum Wiskunde & Informatica (CWI), en principio como un proyecto de afición para mantenerse ocupado durante las vacaciones de Navidad.

El nombre del lenguaje se inspiró en el programa de televisión de la BBC “Monty Python’s Flying Circus” debido a que Guido Van Rossum era un gran aficionado del programa.

Historial de lanzamientos de Python

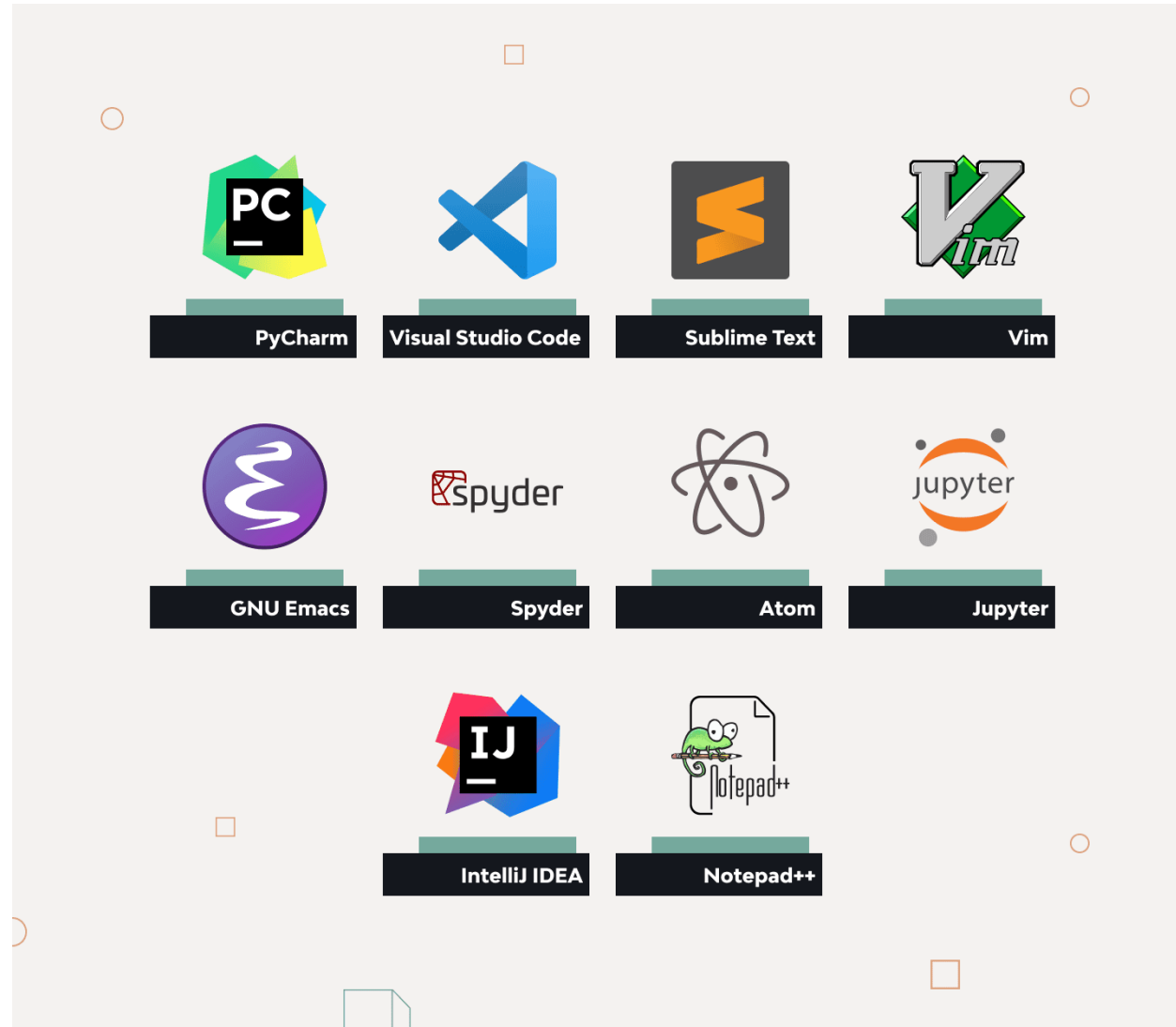
- Guido Van Rossum publicó la primera versión del código Python (versión 0.9.0) en 1991. Dicha versión ya incluía buenas características, como algunos tipos de datos y funciones para la gestión de errores.
- Python 1.0 se lanzó en 1994 con nuevas funciones para procesar fácilmente una lista de datos, como la asignación, el filtrado y la reducción.
- Python 2.0 se lanzó el 16 de octubre de 2000, con nuevas características útiles para los programadores, como la compatibilidad con los caracteres Unicode y una forma más corta de recorrer una lista.
- El 3 de diciembre de 2008, se lanzó Python 3.0. Incluía características como la función de impresión y más soporte para la división de números y la gestión de errores.

Programa simple

```
#Primer programa  
print("Hola mundo!!")
```

```
#Segundo programa  
a="Hola "  
b="mundo!! "  
print(a+b)
```


Herramientas o IDE



Primer programa en Python

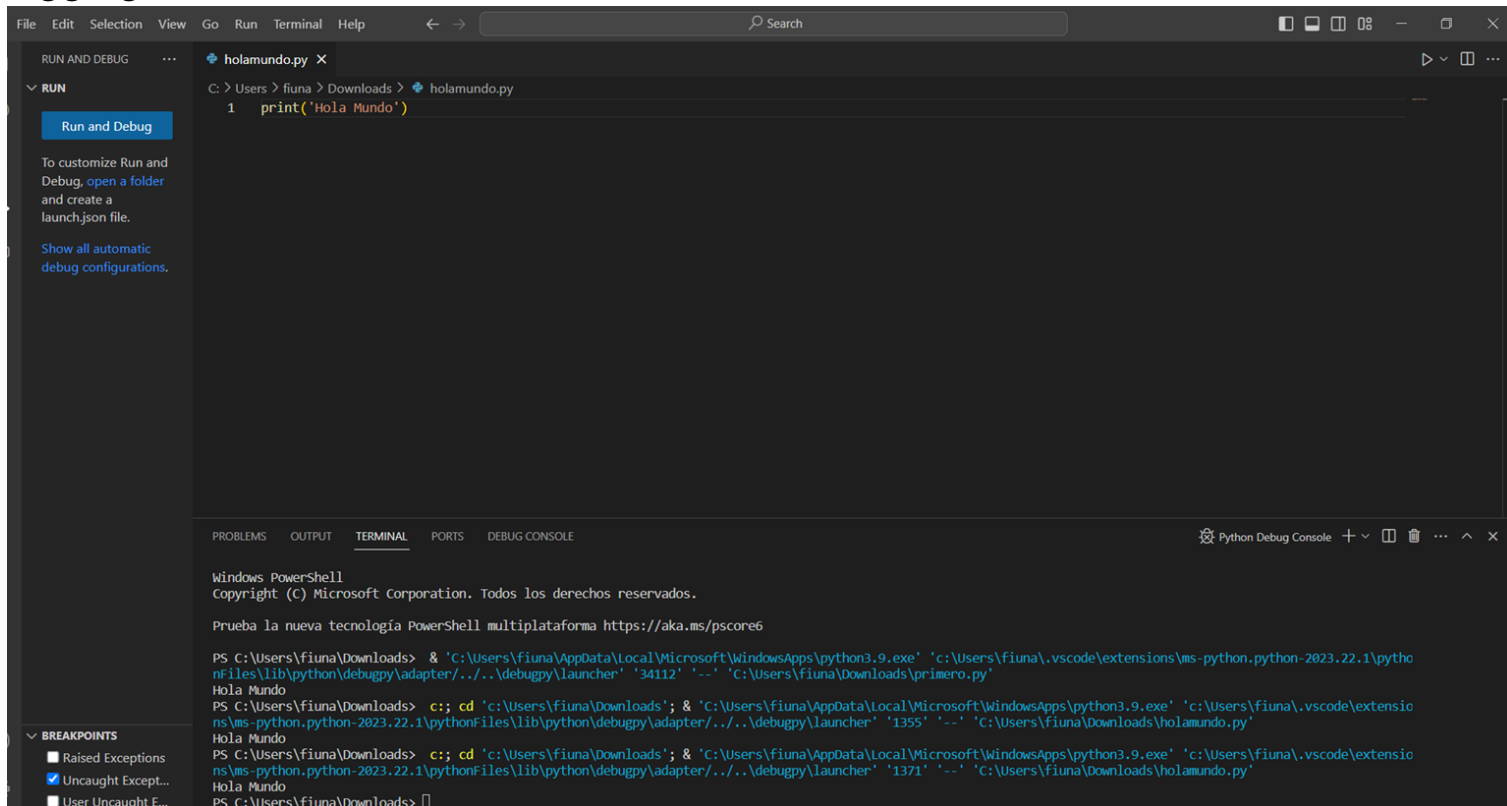
Se crea un programa nuevo: File->New File

Se escribe el programa: Según se muestra en la impresión de pantalla.

Print: Orden de Python que indica que se debe imprimir algo en pantalla (mensaje o variable, cuando es mensaje se coloca este entre comillas “)

Una vez escrito el programa, se guarda: File-> Save as...

Para verificar errores y correr el programa si no hay errores: Run->Start debugging



```
File Edit Selection View Go Run Terminal Help
holamundo.py X
C:\Users\fiuna\Downloads\holamundo.py
1 print('Hola Mundo')

Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
Python Debug Console
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\fiuna\Downloads> & 'C:\Users\fiuna\AppData\Local\Microsoft\WindowsApps\python3.9.exe' 'c:\Users\fiuna\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '34112' '--' 'c:\Users\fiuna\Downloads\primero.py'
Hola Mundo
PS C:\Users\fiuna\Downloads> cd 'c:\Users\fiuna\Downloads'; & 'C:\Users\fiuna\AppData\Local\Microsoft\WindowsApps\python3.9.exe' 'c:\Users\fiuna\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '1355' '--' 'c:\Users\fiuna\Downloads\holamundo.py'
Hola Mundo
PS C:\Users\fiuna\Downloads> cd 'c:\Users\fiuna\Downloads'; & 'C:\Users\fiuna\AppData\Local\Microsoft\WindowsApps\python3.9.exe' 'c:\Users\fiuna\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '1371' '--' 'c:\Users\fiuna\Downloads\holamundo.py'
Hola Mundo
PS C:\Users\fiuna\Downloads>
```

Estructura de un programa

<i>Comentarios</i>
import numpy as np
funciones

Imprimir Pi

"""

comentario

largo

"""

import math

x = math.pi

print(x)

Los comentarios son ignorados por el intérprete, pero son útiles para los programadores, pues pueden proporcionar información sobre el código.

Creando variables

- No se declaran variables en python.
- La variable se crea en la primera asignación

```
x = 5
y = "Juan Perez"
print(x)
print(y)
```

- La variable puede cambiar de tipo

```
x = 4      # x es entero int
x = "Juan Perez" # x ahora es cadena str
print(x)
```

Atención

== sirve para comparar si es igual

Operador de asignación (=)

Crea un espacio de memoria de acuerdo al tipo de la expresión y lo asocia al identificador de la derecha. El identificador corresponde a una variable.

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Creando variables

- No se declaran variables en python.
- La variable se crea en la primera asignación

```
x = 5
```

```
y = "Juan Perez"
```

```
print(x)
```

```
print(y)
```

Con Comilla simple es equivalente

```
y = 'Juan Perez'
```

- La variable puede cambiar de tipo

```
x = 4      # x es entero int
```

```
x = "Juan Perez" # x ahora es cadena
```

```
str
```

```
print(x)
```

Tipos de dato y asignación



Python todas las variables tienen tipo. Existe un caso especial que es **None** (como ausencia de valor)

- Para determinar el tipo se utiliza la función **type(<var>)**

Ejemplos de tipos de datos

<code>x = "Hello World"</code>	<code>str</code>
<code>x = 20</code>	<code>int</code>
<code>x = 20.5</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["apple", "banana", "cherry"]</code>	<code>list</code>
<code>x = ("apple", "banana", "cherry")</code>	<code>tuple</code>
<code>x = range(6)</code>	<code>range</code>
<code>x = {"name" : "John", "age" : 36}</code>	<code>dict</code>
<code>x = {"apple", "banana", "cherry"}</code>	<code>set</code>
<code>x = frozenset({"apple", "banana", "cherry"})</code>	<code>frozenset</code>
<code>x = True</code>	<code>bool</code>
<code>x = b"Hello"</code>	<code>bytes</code>
<code>x = bytearray(5)</code>	<code>bytearray</code>
<code>x = memoryview(bytes(5))</code>	<code>memoryview</code>
<code>x = None</code>	<code>NoneType</code>

Convirtiendo tipos(casting)

- Se puede verificar el tipo de variable con la función `type`

```
x = 5
y = " Juan"
print(type(x))
print(type(y))
```

- Los identificadores son sensibles a mayúsculas y minúsculas

```
a = 4
A = " Juan"
#A no va sobre escribir a
```


Datos Numéricos

x = 1 # int

y = 2.8 # float

z = 1j # complex

#Enteros

x = 1

y = 35656222554887711

z = -3255522

#Reales

x = 1.10

y = 1.0

z = -35.59

Notación científica

x = 35e3

y = 12E4

z = -87.7e100

#Complejos

x = 3+5j

y = 5j

z = -5j

Binarios(literales enteros)

x = 0b001

y = 0b010

z = x+y

Datos Booleanos

En programación, a menudo es necesario saber si una expresión es Verdadera (True) o Falsa(False).

Se puede evaluar cualquier expresión en Python y obtener una de dos respuestas: Verdadero o Falso.

Cuando comparas dos valores, la expresión se evalúa y Python devuelve la respuesta booleana.

```
print(10 > 9)
```

```
print(10 == 9)
```

```
print(10 < 9)
```

Nombres de variables

- Una variable puede tener un nombre corto (como key) o un nombre más descriptivo (edad, nombre_auto, volumen_total). Reglas para variables de Python:
- El nombre de una variable debe comenzar con una letra o _.
- El nombre de una variable no puede comenzar con un número
- El nombre de una variable solo puede contener caracteres alfanuméricos y guiones bajos (A-z, 0-9 y _).
- Los nombres de las variables distinguen entre mayúsculas y minúsculas (edad, Edad y EDAD son tres variables diferentes)
- Un nombre de variable no puede ser ninguna de las palabras clave de Python.

```
mivar = "compu"  
mi-var = "compu " < Incorrecto  
mi_var = "compu "  
_mi_var = "compu "  
2mivar = "compu " < Incorrecto  
miVar = "compu "  
MiVAR = "compu "  
mi var = "compu " < Incorrecto  
mivar2 = "compu "
```

Expresiones y Operadores

Las expresiones son una combinación válida de símbolos de operaciones, constantes literales, variables, paréntesis y llamada a funciones.

```
apellido + " " + nombre  
y + (z + 3) + x**2  
(a-2) < (b+4)  
a and b  
4%2  
4%3  
3/2  
3//2
```

```
-5 % 3  
5 % -3  
-10 % 3  
10 % 3.0
```

Los operadores son símbolos que se usan para indicar operaciones sobre los datos u operandos y tienen significado de acuerdo al tipo del dato.

Operadores

Operación	Operador	Aridad	Asociatividad	Precedencia
Exponenciación	**	Binario	Por la derecha	1
Identidad	+	Unario	—	2
Cambio de signo	-	Unario	—	2
Multiplicación	*	Binario	Por la izquierda	3
División	/	Binario	Por la izquierda	3
División entera	//	Binario	Por la izquierda	3
Módulo (o resto)	%	Binario	Por la izquierda	3
Suma	+	Binario	Por la izquierda	4
Resta	-	Binario	Por la izquierda	4

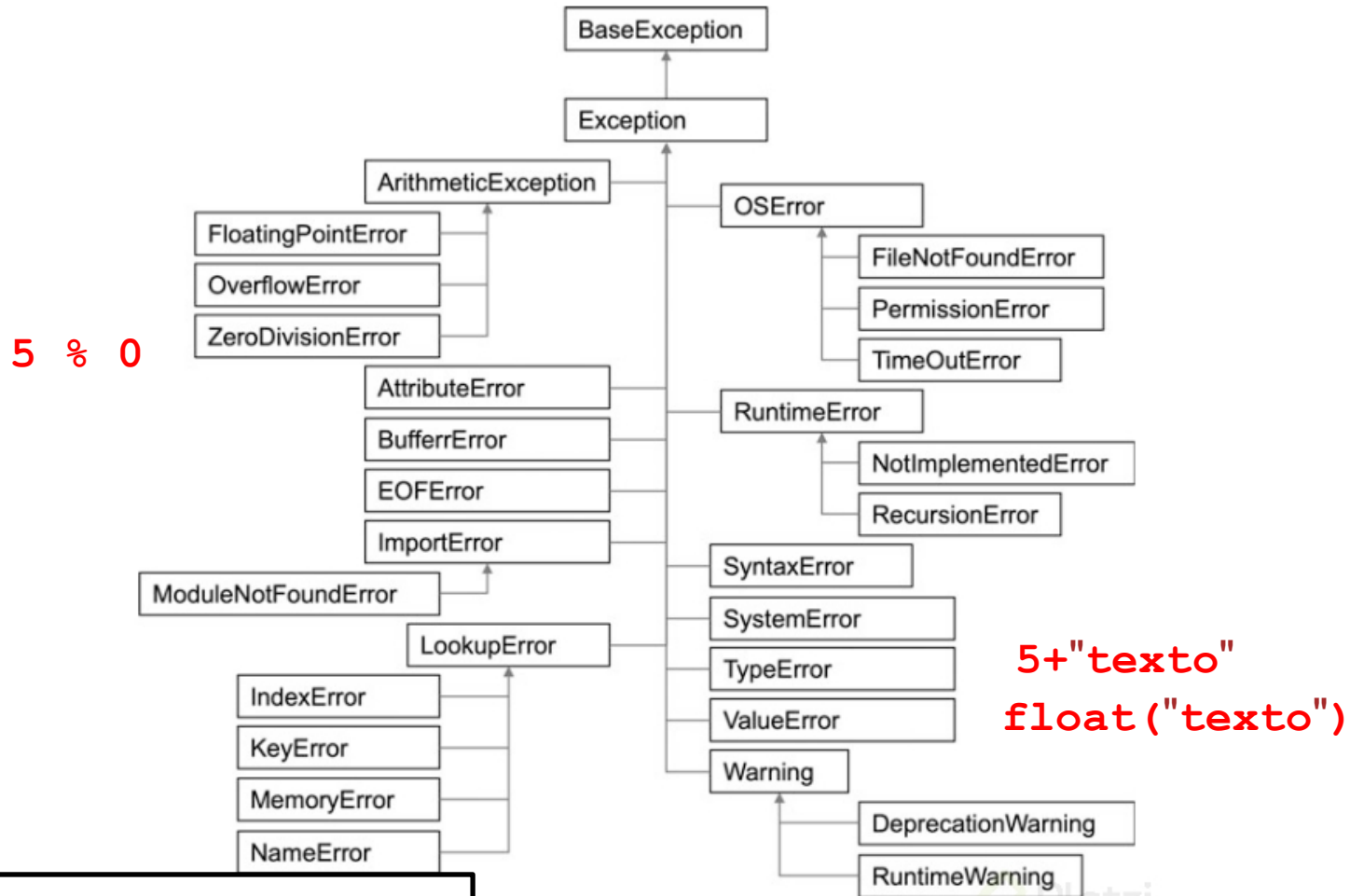
Operadores

Operator	Example	Same As
=	x = 5	x = 5
+=	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
&=	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

Funciones de python

Function	Description
<u>abs()</u>	Returns the absolute value of a number
<u>all()</u>	Returns True if all items in an iterable object are true
<u>any()</u>	Returns True if any item in an iterable object is true
<u>ascii()</u>	Returns a readable version of an object. Replaces none-ascii characters with escape character
<u>bin()</u>	Returns the binary version of a number
<u>bool()</u>	Returns the boolean value of the specified object
<u>bytearray()</u>	Returns an array of bytes
<u>bytes()</u>	Returns a bytes object
<u>callable()</u>	Returns True if the specified object is callable, otherwise False
<u>chr()</u>	Returns a character from the specified Unicode code.
<u>classmethod()</u>	Converts a method into a class method
<u>compile()</u>	Returns the specified source as an object, ready to be executed
<u>complex()</u>	Returns a complex number
<u>delattr()</u>	Deletes the specified attribute (property or method) from the specified object

Tipos de errores



```
try:
    print(x)
except NameError:
    print("La variable no está definida")
except:
    print("Otro tipo de error")
```


Depuración

The image shows the VS Code Python Debug Console interface with several annotations in green text and arrows:

- Breakpoint**: An arrow points to a yellow breakpoint icon on line 9 of `app.py`.
- Variables Panel**: The `VARIABLES` panel on the left shows the `Locals` scope with variables like `name`, `today`, `day`, `max`, `min`, `month`, `resolution`, and `year`.
- WATCH Panel**: The `WATCH` panel shows the variable `name` with its value `'Sebastian'`. An annotation says: "You can define some variables that you want to watch all the time".
- CALL STACK Panel**: The `CALL STACK` panel shows the current frame `home` in `app.py` at line 9.
- DEBUG CONSOLE Panel**: The `DEBUG CONSOLE` panel at the bottom shows the output of the debug session, including the command to run the app and the Flask server output.
- Code Editor**: The `app.py` file is open, showing the `home` function. A green arrow points to the `formatted_today` variable in the function, with an annotation: "If you hover your mouse over a variable, you will see its current value".

Annotations in the image:

- Breakpoint**: Points to the breakpoint on line 9 of `app.py`.
- List of all available variables in the current scope**: Points to the `VARIABLES` panel.
- You can define some variables that you want to watch all the time**: Points to the `WATCH` panel.
- You can execute any Python code in the DEBUG CONSOLE tab**: Points to the `DEBUG CONSOLE` panel.
- If you hover your mouse over a variable, you will see its current value**: Points to the `formatted_today` variable in the code editor.

Entrada/Salida

```
x = input()
print(' Hola, ' + x)
```

```
x = input('Ingrese su nombre:')
print('Hola, ' + x)
```

```
x = int(input('Ingrese su edad:'))
print(' Tienes ' + x + ' años')
```

```
peso = float(input("Dígame su peso en kg: "))
print(f"Su peso es {peso} kg")
```

```
peso = float(input("Dígame su peso en kg: "))
print("Su peso es" + str(peso) + " kg")
```

```
peso = float(input("Dígame su peso en kg: "))
print("Su peso es ", peso, " kg", sep=None)
```

```
peso = float(input("Dígame su peso en kg: "))
print("Su peso {:.10.4f} es kg".format(peso))
```

Cadenas o Texto(str)

- Es un tipo secuencial en Python (elementos accesibles por un índice)
- Es inmutable y esta delimitado por comillas simples o dobles
- Cada elemento es de tipo UniCode(codificación de caracteres)
- Son comparables.

x= "Juan Perez"

y = 'Juan Perez'

z = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""

z= " La Facultad de Ingeniería es la "mejor " facultad "

z= " La Facultad de Ingeniería es la \"mejor \ " facultad "

x*5 # que imprimiría? 😊 **tema de examen**

Caracteres

`ord(a)`

97

`chr(65)`

A

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	\$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	
29	1d	035	GS	61	3d	075	=	93	5d	135]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.overcoded.net

```
print('\u0061\u0062\u0063')
```

<https://home.unicode.org/>

`"\N{GREEK CAPITAL LETTER DELTA}"` # También se puede usar el nombre del caracter



Cadenas o Texto(str)

Es posible acceder a porciones de la cadena, de la forma

<cadena>[<inicio>,<fin>,<paso>] donde <fin>y <paso>son opcionales.

```
x= " computación"  
print(x[0])
```

```
print(x[0:3])
```

```
print(x[3::2])
```

```
# pasar a mayúsculas  
a = "Hello, World!"  
print(a.upper())
```

```
# pasar a minúsculas  
  
a = "Hello, World!"  
print(a.lower())
```

```
# remover espacios  
a = " Hello, World! "  
print(a.strip())
```

```
# formato  
edad = 36  
txt = "Tengo {} años"  
print(txt.format(edad))
```

```
# formato  
cantidad = 3  
item = 567  
precio = 49.95  
orden = "quiero {} de {} por {} guaranies"  
print(orden.format(cantidad, item, precio))
```

Cadenas o Texto(str)

https://www.w3schools.com/python/python_strings_methods.asp

Method	Description
<u>capitalize()</u>	Converts the first character to upper case
<u>casefold()</u>	Converts string into lower case
<u>center()</u>	Returns a centered string
<u>count()</u>	Returns the number of times a specified value occurs in a string
<u>encode()</u>	Returns an encoded version of the string
<u>endswith()</u>	Returns true if the string ends with the specified value
<u>expandtabs()</u>	Sets the tab size of the string
<u>find()</u>	Searches the string for a specified value and returns the position of where it was found
<u>format()</u>	Formats specified values in a string
format_map()	Formats specified values in a string
<u>index()</u>	Searches the string for a specified value and returns the position of where it was found
<u>isalnum()</u>	Returns True if all characters in the string are alphanumeric
<u>isalpha()</u>	Returns True if all characters in the string are in the alphabet
<u>isascii()</u>	Returns True if all characters in the string are ascii characters
<u>isdecimal()</u>	Returns True if all characters in the string are decimals
<u>isdigit()</u>	Returns True if all characters in the string are digits
<u>isidentifier()</u>	Returns True if the string is an identifier

Ejercicios

- Escribir una variable `nombre_compañero` y asignar el nombre del compañero de al lado. Imprimir un saludo
- Escribe un programa que pida al usuario su nombre y lo salude por su nombre.
- Asignar el nombre y el apellido de tu compañera/o a las variables `nombre` y `apellido`. Luego, imprimir una frase, utilizando las variables. EJEMPLO: "Hola, Juan Pérez! Bienvenido a la segunda clase de Fundamentos de Programación"

Ejercicios

- Escribir un programa que evalúe la siguiente función

$$3 \cdot a - 4b/a^2$$

Dado: $a = 2$ y $b = 5$

- Escriba un programa que calcule la circunferencia de un círculo, dado su radio. Asuma $\pi = 3,14$.
- Dado un valor de Temperatura en grados Centígrados, imprima el valor en grados Fahrenheit:

$$^{\circ}\text{F} = \frac{9}{5} \cdot ^{\circ}\text{C} + 32$$

Ejercicios

- Solicitar al usuario un número de 5 cifras e imprimir el dígito de la centena.
- Calcular el promedio de tres números ingresados por el usuario.
- Calcular la longitud de la hipotenusa de un triángulo rectángulo dado sus catetos.
- Dada una lista de palabras, utilizar `join()` para convertirlas en una oración completa, mostrando el resultado al usuario.

Ejercicios

- Pide al usuario que ingrese una palabra y determina si es un palíndromo (se lee igual de adelante hacia atrás que de atrás hacia adelante). El programa debe devolver True o False, sin usar estructuras de control condicional (if).
- Escribe un programa en Python que lea tres números (enteros o flotantes) desde la entrada del usuario y muestre la suma, el promedio, el número más alto y el más bajo.

Ejercicios

- Pide al usuario que introduzca un número de horas y conviértelo en segundos. Muestra el resultado.
- Escribe un programa que convierta dólares a euros. El usuario debe ingresar la cantidad en dólares y el programa debe mostrar la cantidad equivalente en euros. Considera una tasa de conversión fija.
- Solicita al usuario su peso en kilogramos y su altura en metros. Calcula y muestra su Índice de Masa Corporal (IMC).

$$IMC = \frac{Peso [Kg]}{Altura [m]^2}$$

Ejercicios

- Solicita al usuario el capital inicial, la tasa de interés anual (como un porcentaje) y el tiempo en años. Calcula y muestra el interés simple ganado.

$$I = P \cdot r \cdot t$$

Donde:

- I es el interés simple,
- P es el capital inicial (principal),
- r es la tasa de interés anual en forma decimal (por ejemplo, una tasa del 5% se expresaría como 0.05),
- t es el tiempo en años.