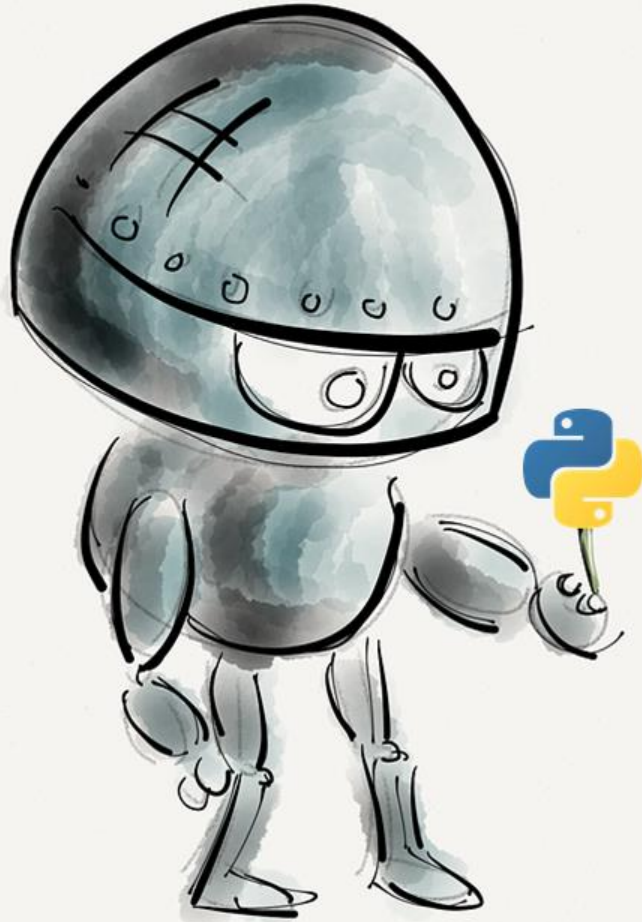




Pensamiento Computacional

Shell, casting,
documentación

```
print("Todos somos programadores en potencia")
```





Shell

>>>

- *Python comand line*: Realmente no me permite codificar un programa. Es una línea de comandos.

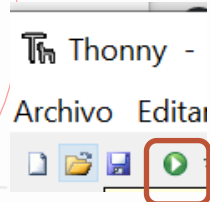
IDE

- Integrated Development Environment para Python, es decir, el entorno de desarrollo integrado para Python
- Ahí escribes tus programas y los “corres”. La “corrida” se muestra en el Shell

Para correr un programa

- El archivo deberá tener nombre.
- Acostúmbrate a nombrar tus programas antes de empezar a escribir el código.
- No olvides la extensión **.py**

¿Y en thonny?



- Muysimple:

Observa la "documentación" Esto implica hacer comentarios sobre tu código para que no sólo tu entiendas.

Puedes poner el texto entre tres comillas y escribir en varios renglones, o poner un signo # y escribir en un solo renglón.

Ningún comentario aparece en la corrida, pero tu programa es más comprensible.

```
1 """
2 Programa que calcula la edad en el 2037
3 AnaRaquel Sanromán
4 2020
5 """
6 print("""Este programa \ncalcula \ntu edad 2037")
7 Dame tu nombre """)
8 nombre = input("Dime en qué año naciste ")
9 nacimiento = int(nombre)
10 print(f"Tu edad en el 2037 será de {edad2037} años")
11
```

Shell

Python 3.7.2 (bundled)

```
>>> %Run edad.py

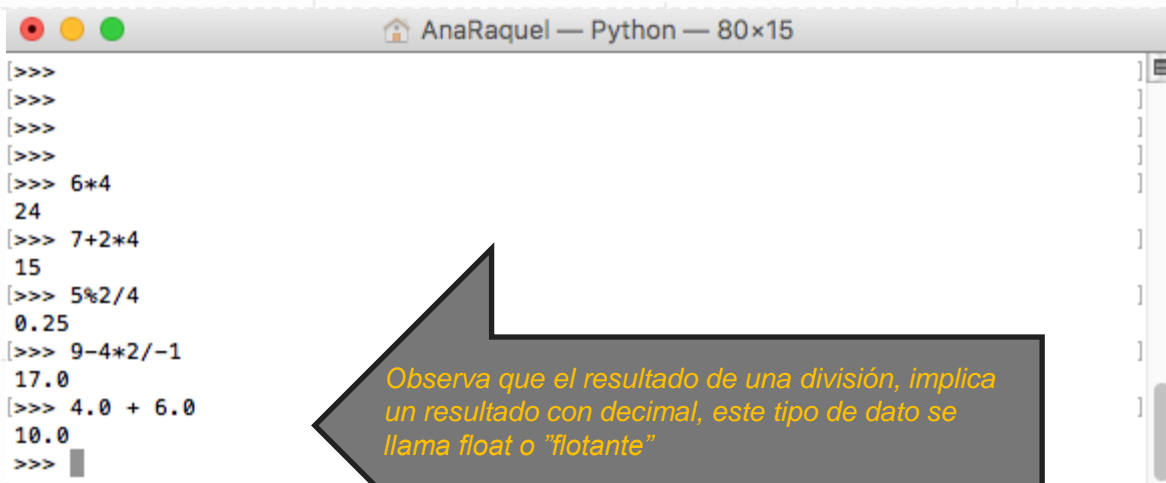
Este programa
calcula
tu edad 2037
Dame tu nombre Ana
Dime en qué año naciste 1969
Ana tu edad en el 2037 será de 68 años

>>>
```

Operaciones sencillas

- Python puede resolver operaciones ingresando directamente en el shell.

*Observa los resultados...
¿Conoces sobre precedencia
de operadores?*



```
>>>  
>>>  
>>>  
>>>  
>>> 6*4  
24  
>>> 7+2*4  
15  
>>> 5%2/4  
0.25  
>>> 9-4*2/-1  
17.0  
>>> 4.0 + 6.0  
10.0  
>>>
```

*Observa que el resultado de una división, implica
un resultado con decimal, este tipo de dato se
llama float o "flotante"*

Operadores

Realiza las siguientes operaciones
en el shell de Python

```
3 + 5
8 - 3
5 * 6
5 ** 2
5 / 2
5 // 2
5 % 2
(5 + 2) ** 2
```

Operadores de división

// DIV
% MOD

$14 \% 3 = 2$

$14 // 3 = 4$

DIV= división
entera

3 14

4

2

MOD = residuo

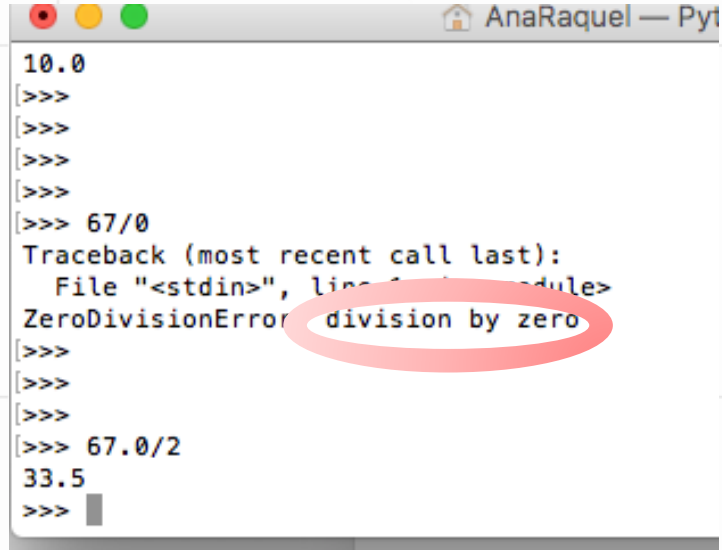
Precedencia de operadores aritméticos

Operación	Operador	Prioridad
Paréntesis	()	1
Potencia	**	2
Multiplicación	*	3
División	/	
División entera	//	
Módulo o residuo	%	
Suma	+	4
Resta	-	

Si están juntos, tienen la **misma prioridad** y se resuelven de izquierda a derecha

Division entre 0

- Dividir entre 0 causa un error porque no hay respuesta.
- Cada que Python te indique un error, revisa el último renglón, normalmente indica el tipo de error y si pones atención, será muy fácil corregirlo.



```
AnaRaquel — Pyt
10.0
[>>>
[>>>
[>>>
[>>>
[>>> 67/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
[>>>
[>>>
[>>> 67.0/2
33.5
[>>> █
```

The screenshot shows a Python interpreter window titled "AnaRaquel — Pyt". The output shows a series of empty prompts followed by the command `67/0`, which results in a `ZeroDivisionError: division by zero`. The error message is highlighted with a red oval. Subsequent prompts show `67.0/2` resulting in `33.5`.

Asignación

- Puedes asignar el resultado de una operación a una VARIABLE.
- Una variable, como su nombre lo indica, almacena temporalmente un valor.
- Ese **valor** se asigna con el operador asignación **=**
- Al hecho de asignarle un valor inicial a una variable se conoce como **inicialización**.

Podemos utilizar la función **print** para ver el contenido de una variable

```
AnaRaquel — Python — 41x13
>>>
[>>> a=89
[>>> b=1
[>>> c=a+b
[>>> print(c)
[90
[>>>
```

Asignaciones

- Simples

$a = 15$

- Contador

$a = a+1$

- Acumulador

$a = a+b$

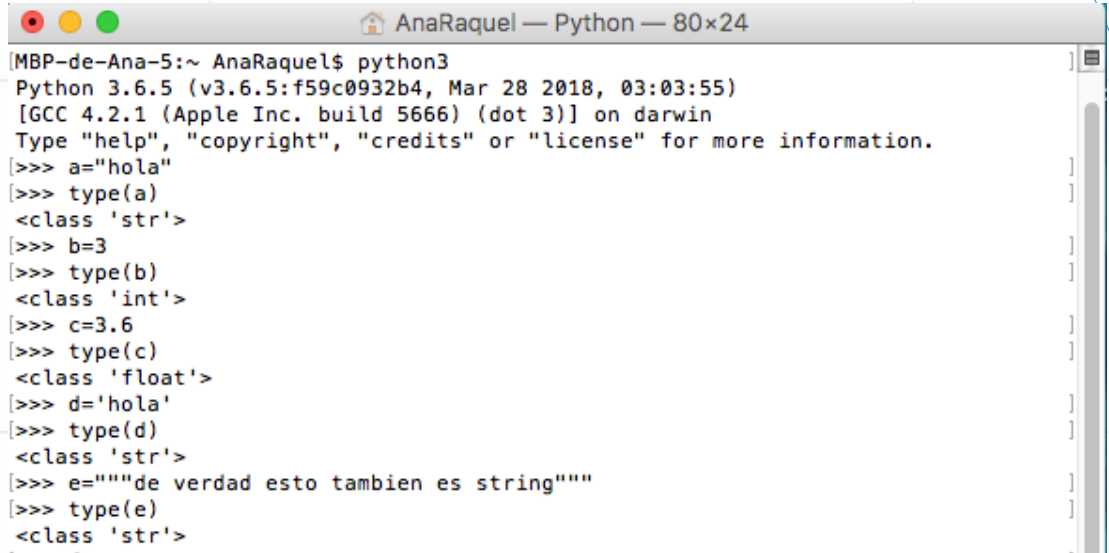
- De trabajo

$a = c+b*2/4$

Expresiones	Equivalencias
$\text{num} += 10;$	$\text{num} = \text{num} + 10;$
$\text{num} -= 10;$	$\text{num} = \text{num} - 10;$
$\text{num} *= 10;$	$\text{num} = \text{num} * 10;$
$\text{num} /= 10;$	$\text{num} = \text{num} / 10;$
$\text{num} \% = 10;$	$\text{num} = \text{num} \% 10;$

Tipos de datos

- Abre el Shell de Python.
- Inicializa variables con diferentes datos.
- Con la instrucción **type** averiguaremos el tipo de dato detectado por Python.



```
AnaRaquel — Python — 80x24
MBP-de-Ana-5:~ AnaRaquel$ python3
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> a="hola"
[>>> type(a)
<class 'str'>
[>>> b=3
[>>> type(b)
<class 'int'>
[>>> c=3.6
[>>> type(c)
<class 'float'>
[>>> d='hola'
[>>> type(d)
<class 'str'>
[>>> e="""de verdad esto tambien es string"""
[>>> type(e)
<class 'str'>
```

Tipos de datos simples

int
(entero)

- 8
- 345

float
(real)

- 7.0
- 3.1415

string
(cadena de caracteres)

- 'Hola mundo!'
- "Bienvenidos"
- """"Puedo tener tres comillas para un texto""""
- '''o tres apóstrofes'''

Variables

- Como verás, las variables tienen un nombre.
- En los ejemplos anteriores son simples letras, pero es MUY importante que te acostumbres a poner nombres de variables que **identifiquen** para qué están siendo utilizadas.
- Pueden empezar con letra ó guión bajo _
- Sólo pueden contener números, letras ó _
- Hace diferencia entre mayúsculas o minúsculas.
- NO puedes usar las palabras reservadas

Palabras reservadas

```
[>>> help ("keywords")]
```

Here is a list of the Python keywords. Enter any keyword to get more help.

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

```
>>> 
```

No puedes usar palabras Reservadas como nombre de una variable

Entrada y salida a pantalla (input / print)

input()

siempre guardará la
entrada como un String

input(mensaje_opcional)

- Lo utilizamos asignando el resultado del input a una variable.

Ejemplo:

nombre=input("¿Cuál es tu nombre?")

- Lo que conteste el usuario por teclado se asigna a la variable nombre como un dato de tipo String

print()

Sin parámetro,
implica un cambio de
línea.

- **print() - despliegue a pantalla**

print(objeto)

print(objeto1, objeto2, etc)

print("string_con_formato")

Casting

input

siempre
obtendrá
u String

Pero podemos usar una
funcion que convierta
ese string a un número.

Para ello
usamos
int o **float**

```
x="23"  
y=25  
entero=int(x)  
print (entero)  
flotante=float(x)  
print (flotante)  
cadena=str(y)  
print cadena
```

Ejemplo con input:

```
dato=input("Numero 1:")  
num1=int(dato)
```

O en un solo paso (una línea de código)

```
num1=int(input("Numero 1:"))
```

A programar

THINK



CODE

Recuerda...



Si necesitamos utilizar el ingreso del usuario como dato numérico, debemos convertir el *string* recibido al tipo de dato (*int* o *float*)

Se puede hacer en dos pasos:

```
dato=input("Numero 1: ")
```

```
num1=int(dato)
```

O en un sola línea de código

```
num1=int(input("Numero 1: "))
```

Si necesitamos que sea de tipo flotante

```
num1=float(input("Numero 1: "))
```



Recuerda...

01

Analizar

Comprender que nos está pidiendo el problema

03

Codear (codificar)

Es hora de trasladar el algoritmo a código.

02

Diseño

Hacemos un plan de acción, un algoritmo.

04

Pruebas

Comprobamos su efectividad y eficacia.

Ejercicios:

Plantea el algoritmo y desarrolla el código para resolver los siguientes problemas.

1. Saca el promedio de calificaciones de tu semestre pasado.
2. Convierte una cantidad dada por el usuario en pesos a dólares.
3. Calcula la propina que debe dejar una persona de acuerdo con un monto de consumo. El porcentaje de propina es una entrada.

Algoritmo implica que planteas **tu pseudocódigo**:

Identifica qué pide el programa, revisa con qué entradas cuentas y plantea la solución.

¡No olvides documentar tu código!

Qué aprendimos...



Casting

Input recibe la entrada de texto del teclado como string, necesito convertirla a número con la función `int()` o `float()`



Shell

`>>>` No escribo programas en él, pero puedo probar instrucciones y es donde corre el programa



Documentación

`#` para escribir un renglón

`"""`

Varias líneas de código las encierro en tres comillas

`"""`

```
print("¡Ya sé  
programar!")
```



Recursos que utilizarás

- Libro de texto y videos asociados
 - <https://automatetheboringstuff.com/>
 - Videos asociados
 - https://youtu.be/1F_OgqRuSdl
- 