

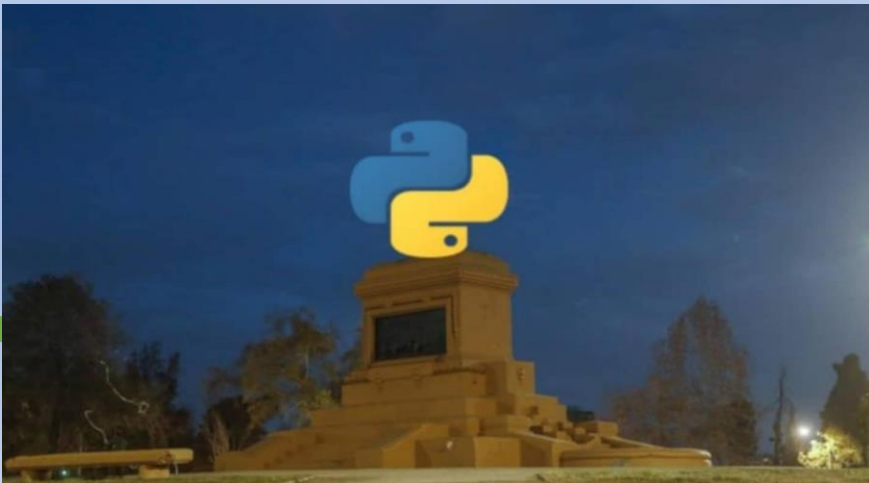
Clase introductoria a Fundamentos de Programación

```
print("Consejos y mas para sobrevivir al ramo")
```



Fundamentos de Programación (10145)

Fundamentos de Computación y Programación (10110)



- La parte de Teoría es exactamente la misma: Python 3.x
- El laboratorio de Civil esta relacionado con Introducción al Diseño en Ingeniería.
- Consiste en 3 informes, cada uno con su presentación + un código principal
- El laboratorio de Ejecución tiene un tema específico dado por la coordinación
- Consiste en 3 informes, cada uno con su presentación + un “portafolio” + un código principal

UNIDAD	TÍTULO	Nº DE HORAS PRESENCIALES
1	FUNDAMENTOS DE PROGRAMACIÓN	24
2	FUNCIONES Y ABSTRACCIÓN	16
3	PROGRAMACIÓN PARA LA INGENIERÍA	20
TOTAL	15 SEMANAS	60

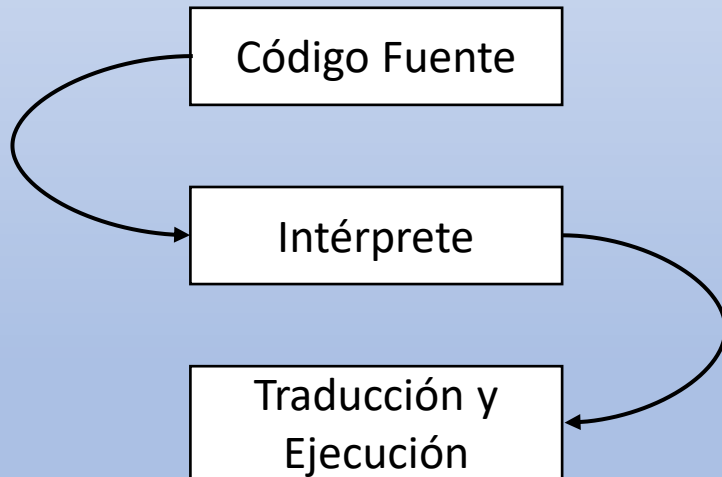


EVALUACIÓN TEORÍA

- La evaluación de cátedra, se calcula a través del promedio simple de cuatro calificaciones con igual ponderación:
 - PEP 1: Fundamentos de programación (Unidad 1)
 - Semana 16-11
 - PEP 2: Funciones y abstracción (Unidades 1 y 2)
 - Semana 21-12
 - PEP 3: Programación para ingeniería (Unidades 1, 2 y 3)
 - Semana 01-03
 - Promedio de notas parciales (6 *quizzes* en Uvirtual)



PYTHON

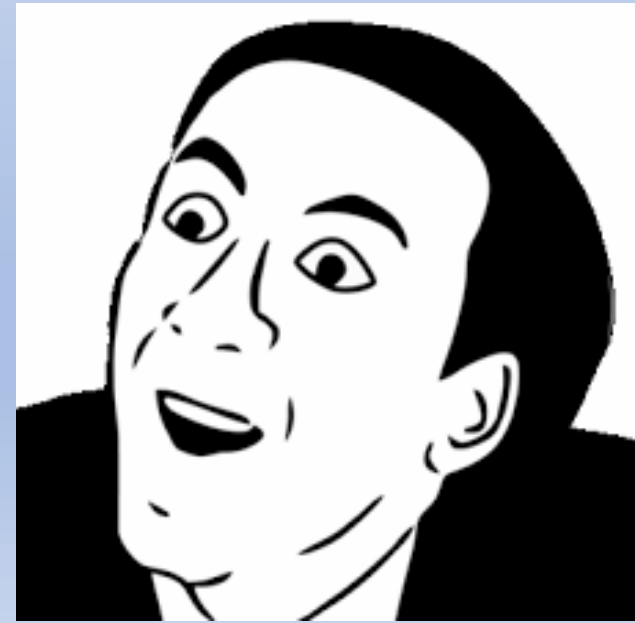
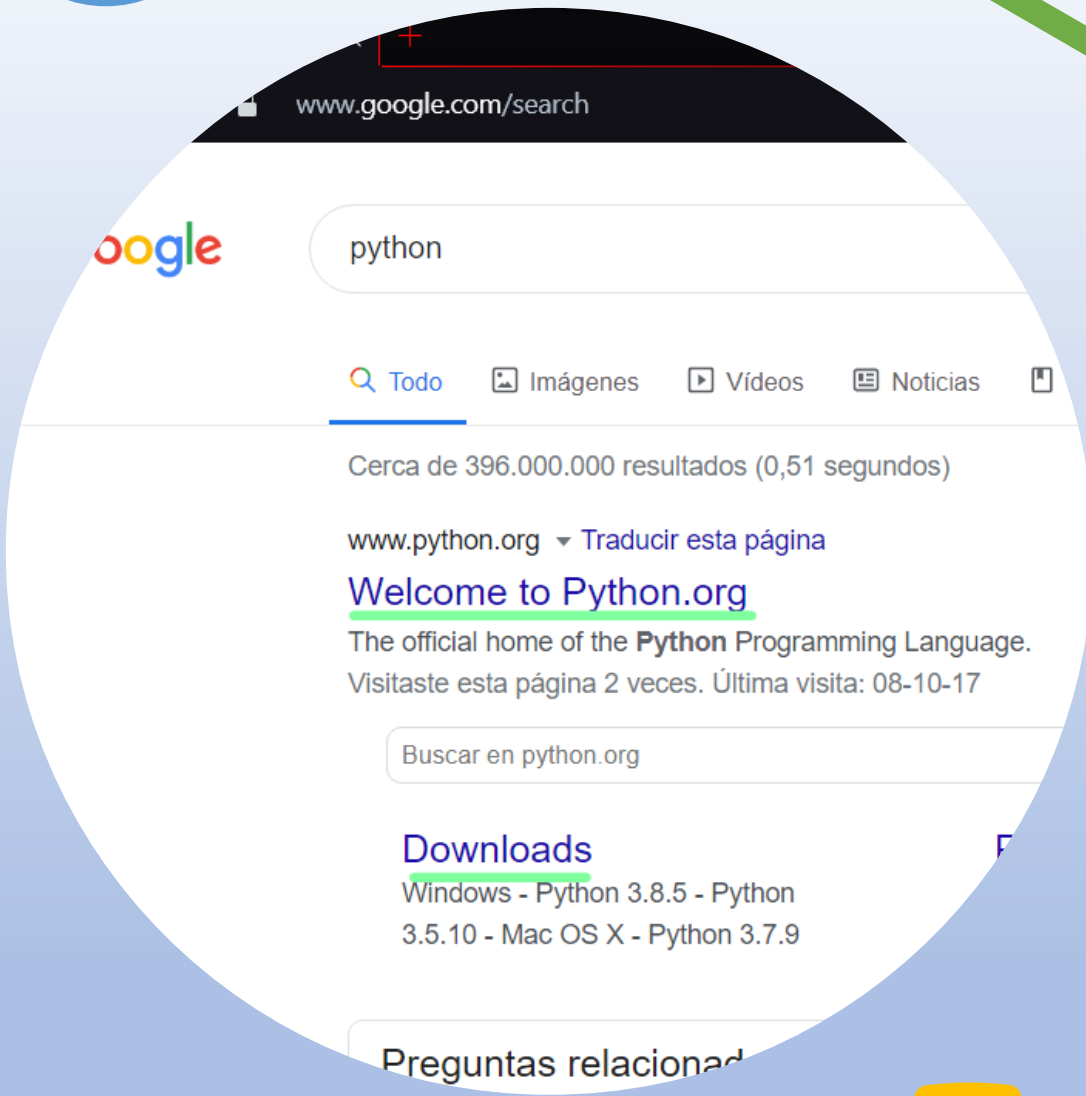


- Creado por Guido Van Rossum en 1989
- El nombre viene inspirado a partir del grupo Monty Python
- Lenguaje de multipropósito, multiparadigma y de estructura simple (buenardo)
- Lenguaje interpretado

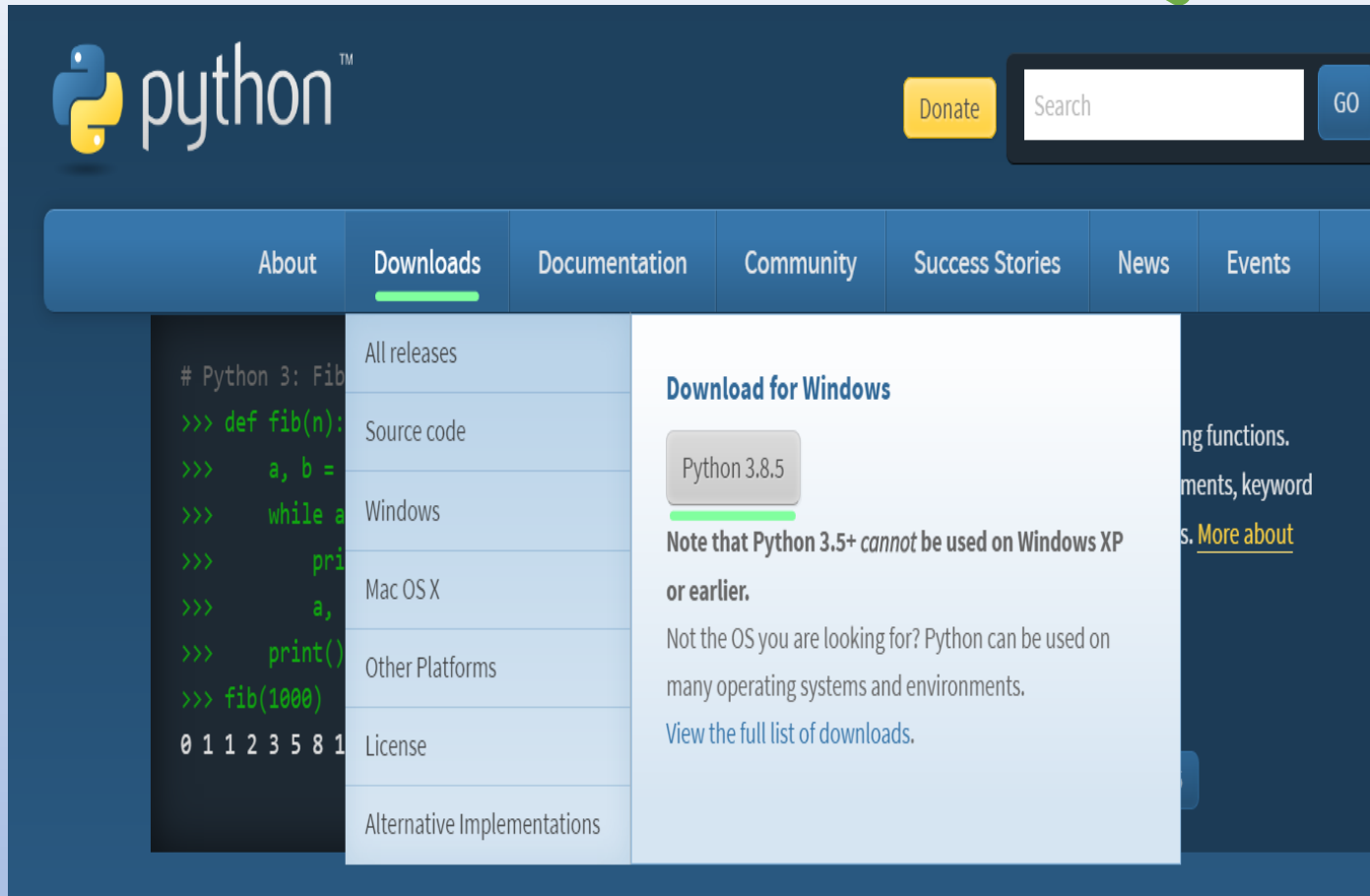
Programar es básicamente explicarle que queremos hacer al computador, en un lenguaje en el que el computador entienda

Instalemos Python :D

1) Googleamos Python jaja



Instalemos Python :D



2) Downloads -> Windows
(Puedes descargar directamente la versión sugerida)

Instalemos Python :D

Python >>> Downloads >>> Windows

Python Releases for Windows

- [Latest Python 3 Release - Python 3.8.6](#)
- [Latest Python 2 Release - Python 2.7.18](#)

Stable Releases

- [Python 3.8.6 - Sept. 24, 2020](#)

Note that Python 3.8.6 *cannot* be used on Windows XP or earlier.

- Download [Windows help file](#)
- Download [Windows x86-64 embeddable zip file](#)
- Download [Windows x86-64 executable installer](#)
- Download [Windows x86-64 web-based installer](#)
- Download [Windows x86 embeddable zip file](#)
- Download [Windows x86 executable installer](#)
- Download [Windows x86 web-based installer](#)
- [Python 3.8.6rc1 - Sept. 8, 2020](#)

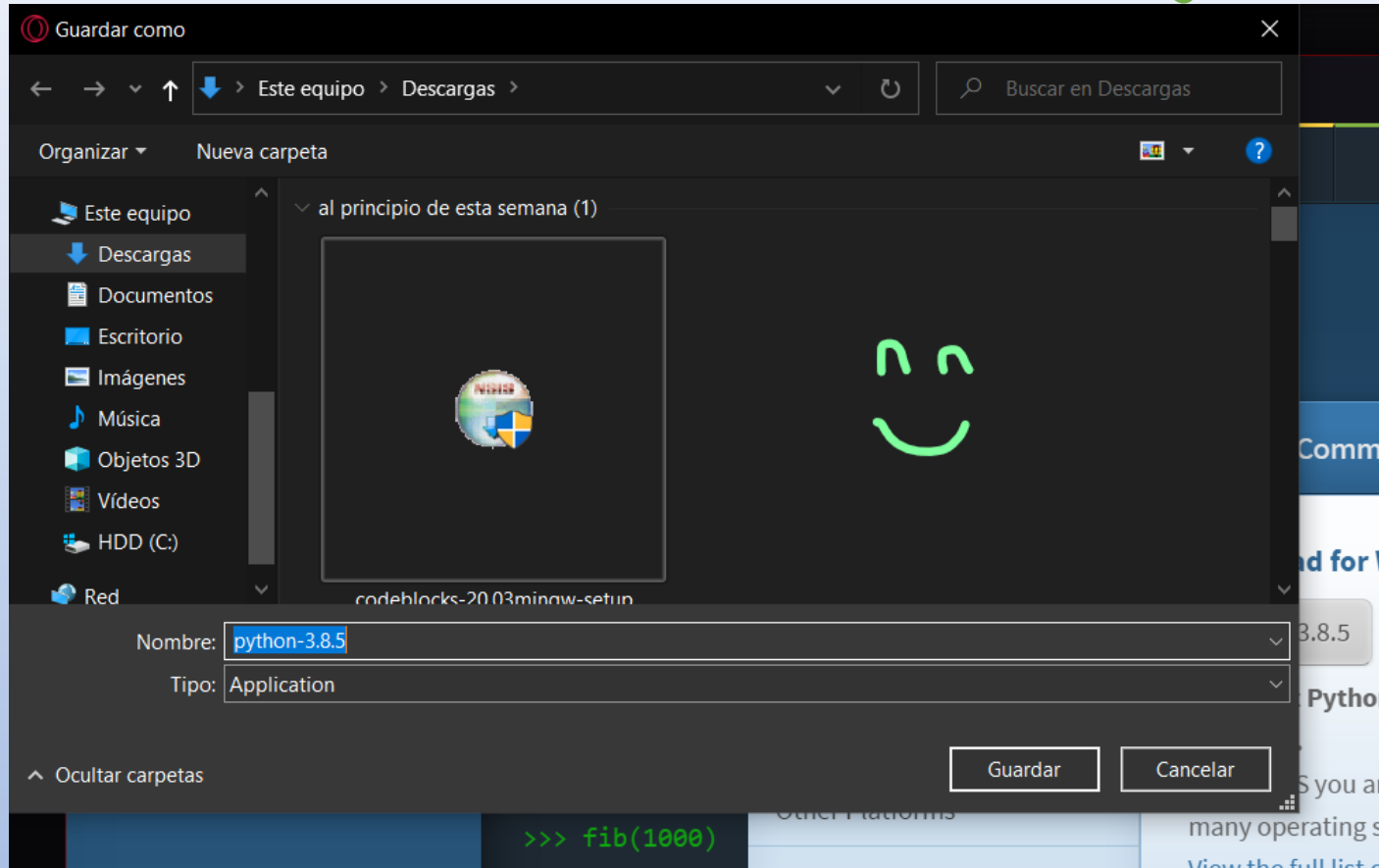
Note that Python 3.8.6rc1 *cannot* be used on Windows XP or earlier.

3) También puedes buscar la versión que mas te acomode.

En este caso, seleccionaremos la versión de 86-64 Bits, en la opción de instalador como ejecutable

Instalemos Python :D

4) Guardamos y ejecutamos (doble click xd) el instalador; el ejecutable .exe



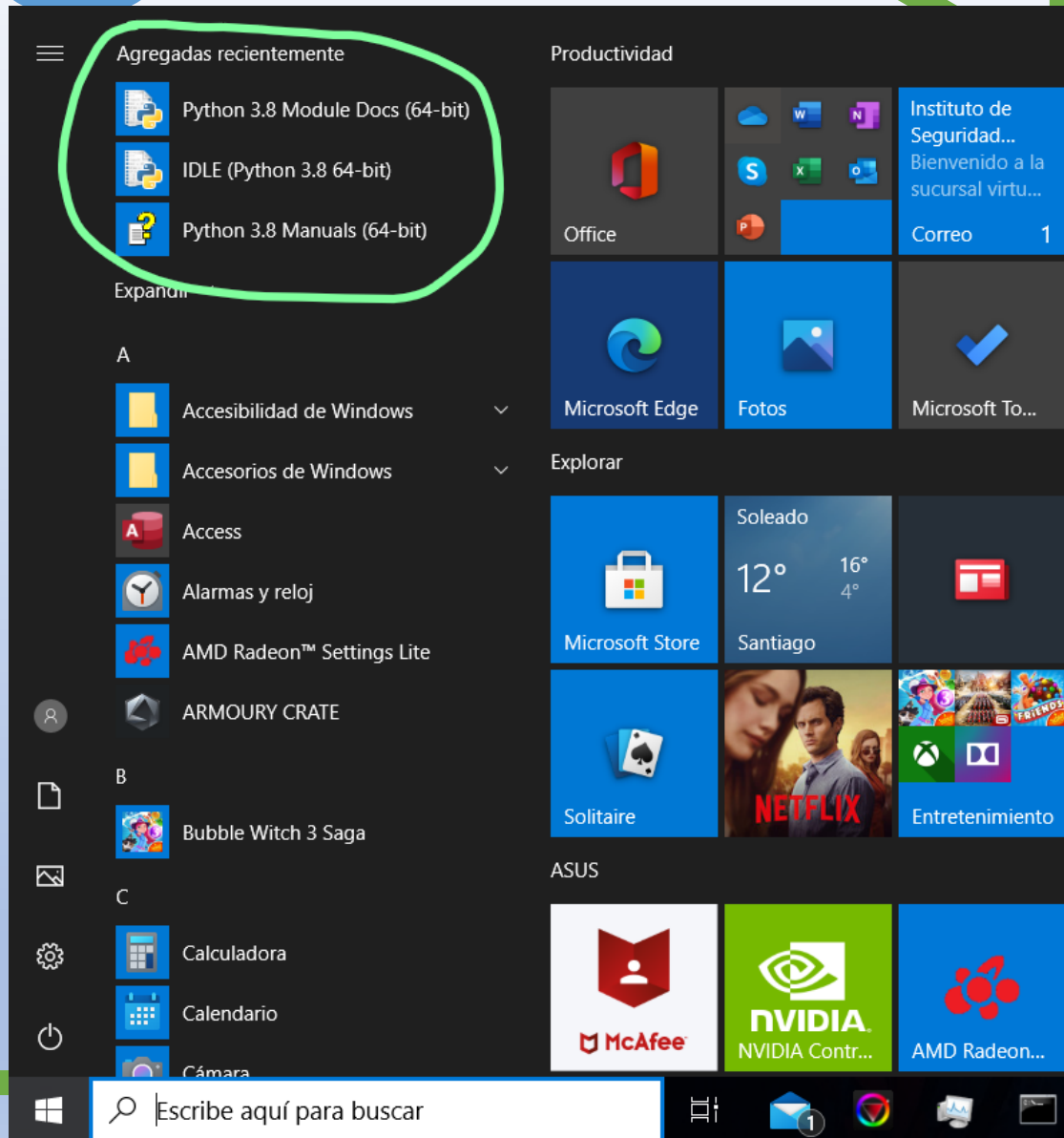
Instalemos Python :D



6) Nos aparecerá esta ventana, deben marcar la casilla “Add Python 3.x to PATH”, esto hará que tu sistema operativo añada el interprete a las variables de entorno, de esta forma tu computador sabrá acerca de la existencia del intérprete y sus procedimientos. **(IMPORTANTE PARA LA UNIDAD 3!)**

Instalemos Python :D

7) Ahora esta todo listo para comenzar a programar :D





generator.net

Variables

En Python es muy útil contar con **variables** que nos permiten **almacenar valores** (datos) de manera ordenada para posteriormente darles un uso, ya sea en algún cálculo o procedimiento particular.

```
a = 39  
b = a  
c = a + b  
print(c)
```

El signo = nos permitirá asignarles a las variables un valor.

File Edit Format Run Options Window Help

```
variable mal creada = 5
```

```
variable_bien_creada = 5
```

Tipos de datos

Enteros

```
#int (integer)
a1 = 10
a2 = -2510
```

De punto Flotante

```
#float (punto flotante)
b1 = -121.5
b2 = 25.10
```

Listas

```
#list (lista)
L = [3,2,1]
```

Valores lógicos

```
#bool (booleano)
c1 = True
c2 = False
```

Texto

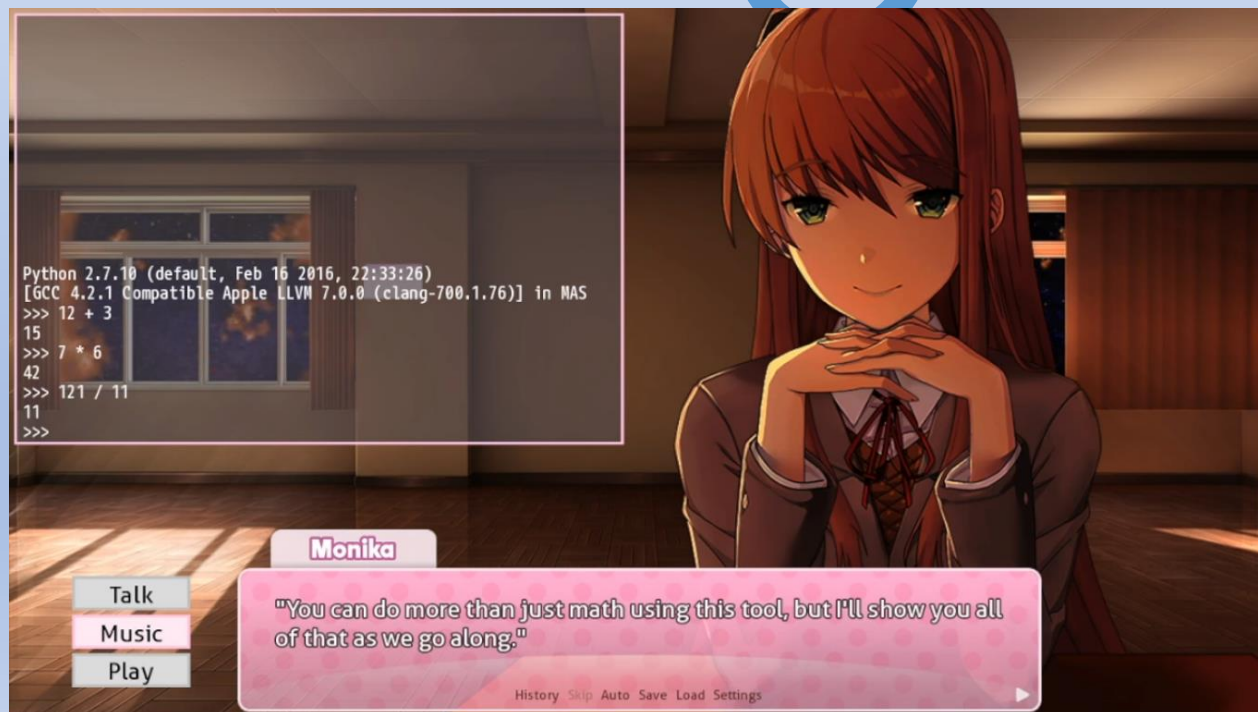
```
#str (string / cadena de texto)
d1 = "Este es un texto"
d2 = 'Asi tambien se puede'
d3 = 'Este es un texto con "comillas"'
```


Previa aclaración de escritura:

```
>>> int(3.64)
```

3

Cuando escribimos un “>>>” significa que eso es lo que esta escrito en el código, y la siguiente línea inmediata se refiere al resultado de dicha línea/s de código



Conversión de tipos de datos

>>> int(3.654) ----> TRUNCA EL NÚMERO

3

>>> float(5)

5.0

>>> int(5)

5

>>> str(-45)

'-45'

>>> int('hola')

Error

Operadores Aritméticos

Suma



```
var_1 = 90  
var_2 = 100  
suma = var_1 + var_2 + 5
```



La variable suma vale 195

Resta



```
var_1 = 90  
resta = 1000.0 - var_1
```



La variable resta vale 910.0

Producto



```
var_1 = 2  
var_2 = 4  
producto = var_1 * var_2
```



La variable producto vale 8

División



```
var_1 = 90  
var_2 = 100  
div1 = var_1 / var_2  
div2 = var_2 / var_1
```



La variable div1 vale 0.9
La variable div2 vale 1.111111....

División Entera

//

Se redondea al entero más cercano.

```
var_1 = 3.5  
var_2 = 2  
div_ent = var_1 // var_2
```

La variable div_ent vale 1.0

Exponente

**

```
var_1 = 2  
var_2 = 3  
potencia = var_1 ** var_2
```

La variable potencia vale $2^3 = 8$

Módulo

%

Resto de la división.

```
var_1 = 25  
var_2 = 5  
resto1 = var_1 % var_2  
resto2 = var_2 % var_1
```

La variable var_1 vale 0
La variable var_2 vale 5

OPERADORES DE COMPARACIÓN

- Además de los operadores aritméticos, en Python existen los siguientes **operadores de comparación**

■ Mayor	>	4.3 > 3.2
■ Mayor o igual	>=	4.0 >= 4.1
■ Menor	<	-2 < 0
■ Menor o igual	<=	-3.14 <= -3.2
■ Igual	==	2.0 == 2
■ Distinto	!=	-2 != -2.1

PRECEDENCIA DE 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
Módulo (o resto)	%	Binario	Por la izquierda	3
Suma	+	Binario	Por la izquierda	4
Resta	-	Binario	Por la izquierda	4
Igual que	==	Binario	—	5
Distinto de	!=	Binario	—	5
Menor que	<	Binario	—	5
Menor o igual que	<=	Binario	—	5
Mayor que	>	Binario	—	5
Mayor o igual que	>=	Binario	—	5
Negación	not	Unario	—	6
Conjunción	and	Binario	Por la izquierda	7
Disyunción	or	Binario	Por la izquierda	8

Entrada y Salida de datos

Para la entrada o lectura de datos se utiliza la función **input**:

```
variable = int(input('Ingrese el número x: '))
```

* Recordar darle **formato** a lo que reciben, y guardarlo en una **variable**.

Para la salida o impresión de datos se utiliza la función **print**:

```
print('El valor de x es: ', x)
```

* Recordar la **coma** para imprimir varias cosas (se usa como separador).

eval(): Si se coloca antes de un input se puede “controlar” el tipo de dato de la entrada. Muy útil cuando se quiere trabajar con listas ingresadas por entrada.

Buenas Prácticas: *Nombres de las Variables*

Para escribir código legible, debemos elegir buenos nombres para nuestras variables que **describan lo que representan**.

Mala Práctica

```
h = input("Ingrese su nombre: ")
```

No es muy descriptivo.



Buena Práctica

```
nombre = input("Ingrese su nombre: ")
```

Queda claro lo que representa la variable.



Otras buenas practicas <3


- NO usar **espacios** para el nombre de una variable. Las variables se deben escribir con **minúscula** separando espacios con **guiones bajos**
- NO usar nombres de variables **iniciadas en números**.
- Si bien Python acepta tildes para los nombres de variables, una buena práctica es **no** utilizarlas.
- Ej: nombre_variable
- Constantes siguen la misma reglas de las variables, pero se escriben en mayúsculas
- Ej: PI

Decisiones: if-else

- La sentencia if ejecuta las instrucciones solo si se cumple una condición. Si la condición es falsa no se hace nada. **Es obligatoria.**
- La sentencia elif (else if) permite ejecutar instrucciones si ninguna de las condiciones anteriores se cumple, y su condición sí se cumple. **Es opcional.**
- La sentencia else se conoce como sentencia por defecto: se ejecuta si ninguna de las condiciones de los if o elif anteriores son verdaderas. **Es opcional y único.**

```
if condicion1:  
    <sentencia 1>  
elif condicion2:  
    <sentencia 2>  
elif condicion3:  
    <sentencia 3>  
else:  
    <sentencia 4>
```

indentación



Iteración: while

- La sentencia while repite una instrucción mientras una **condición** sea verdadera
- **CUIDADO!** Debemos asegurarnos que nuestro programa no se quede atascado en un **ciclo infinito**
- Contador: Variable que usualmente se utiliza para los ciclos while

```
while <condicion>:  
    <instrucción que se repite>
```

Break: nos permite “escapar” de un ciclo while.
Se considera una mala practica

```
while True: ciclo infinito
```

Ejercicio 1: Viaje

Aburrido durante la pandemia, comienza a buscar en internet lugares a los que le gustaría viajar cuando todo termine. Ya que encuentra muchas opciones y no sabe cuál elegir, decide utilizar sus conocimientos de programación para hacer un algoritmo que calcule el costo total de un viaje, a partir de la distancia [km], el rendimiento de su auto [km/l] y el costo del litro de combustible [\$/l] con la siguiente fórmula:

$$\text{costoTotal} = \frac{\text{distancia}}{\text{rendimiento}} \cdot \text{costo}$$

Considere que su programa debe leer la distancia, el rendimiento y el costo, y mostrar por pantalla el costo total.

CHALLENGE ACCEPTED



```
Ingrese la distancia: 150
Ingrese el rendimiento: 10
Ingrese el costo: 700
El costo total es: 10500.0
```

Solución Ejercicio 1: Viaje

```
distancia = float(input("Ingrese la distancia: "))  
rendimiento = float(input("Ingrese el rendimiento: "))  
costo = float(input("Ingrese el costo: "))  
costo_total = distancia / rendimiento * costo  
print("El costo total es: ", costo_total)
```



Ejercicio 2: Triangulo

La fórmula de Herón calcula el área de cualquier triángulo a través de sus lados:

$$A = \sqrt{s(s - a)(s - b)(s - c)}$$

Donde **a**, **b** y **c** son los lados y **s** el semiperímetro:

$$s = \frac{(a+b+c)}{2}$$

Desarrolle un programa en Python que calcule el área **A**, guíese y respete el formato del ejemplo.

```
Ingrese lado: 3
Ingrese lado: 4
Ingrese lado: 5
El área es 6.0
```

Solución Ejercicio 2: Triangulo

```
# BLOQUE PRINCIPAL
# ENTRADAS
lado_a = float(input("Ingrese el lado a: "))
lado_b = float(input("Ingrese el lado b: "))
lado_c = float(input("Ingrese el lado c: "))
# PROCESAMIENTO
s = (lado_a + lado_b + lado_c) / 2
area = ((s * (s - lado_a) * (s - lado_b) * (s - lado_c))) ** 0.5
# SALIDA
print(area)
```

Ejercicio 3: Sumatoria

Construya en Python un programa para calcular la siguiente serie finita, para los primeros n números naturales.

$$\sum_{k=1}^n \frac{k^2+2}{k^3+6k}$$

Solución Ejercicio 3: Sumatoria.

```
# DATOS DE ENTRADA
# Solicita un número
numero = int(input("Ingrese el número de términos de la serie: "))
# Inicializa la variable iteradora
k = 1
# Inicializa la variable acumuladora
suma = 0

# PROCESAMIENTO
while k <= numero:
    # Calcula los resultados parciales
    suma = suma + ((k ** 2 + 2)/(k ** 3 + 6 * k))
    # Incrementa el iterador
    k = k + 1

# SALIDA
print("La suma de los primeros", numero, "términos de la serie es: ", suma)
```

Profes Recomendados:

- Cristian Sepúlveda
- Paulo Quinsacara
- Alejandro Cisterna
- Pamela Landero

MUCHAS GRACIAS POR ASISTIR!

Algunos links útiles:

- Python Tutor:
<http://pythontutor.com/visualize.html#mode=edit>
- PEP 8 Online: <http://pep8online.com/>
- Funciones nativas en Python:
<https://docs.python.org/es/3/library/functions.html>

En mi drive varios resúmenes y cosas que pueden ayudarles!

Link (Correo USACH):

https://drive.google.com/drive/folders/1CifZ_2vfwwx8Qolk1NrXXsavAZ0FpjuvF?usp=sharing