Seminario de Lenguajes - Python

Cursada 2024

Clase 1: conceptos básicos e introducción al lenguaje

¿Con qué términos asociás al lenguaje Python?

- Completar: https://www.menti.com/bbh714ivt4

Resultados

Empezamos con las encuestas ...

Veamos en catedras.linti: Encuesta clase 1: sobre el SL

```
ENCUESTA 1: ¿saben qué es el software libre?
Si - NO


ENCUESTA 2: ¿usaron software libre?
A: Si - B: NO - C: No se
```

El software libre

El **software libre** se refiere a la libertad de los usuarios para:

- ejecutar,
- copiar,
- distribuir,
- estudiar,
- cambiar y mejorar el software.

La definición del software libre habla de **libertades**.

> El acceso al código fuente es un requisito previo para esto.

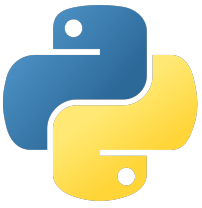
¿Por qué hablamos de software libre?

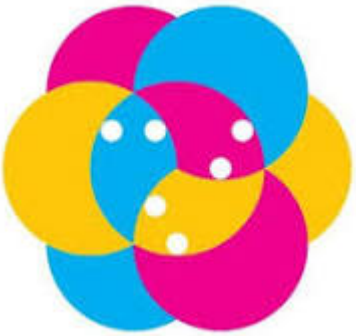
- Nosotros vamos a usar software libre.
- Vamos a proponer que nuestros desarrollos sean software libre.

Hablemos de Python ...

- Desarrollado por Guido Van Rossum en el centro de investigación en Matemáticas CWI en Países Bajos.
- El nombre proviene del grupo de cómicos ingleses Monty Python
- Es un lenguaje que en los últimos años ha crecido de manera constante.
  - Stack Overflow Trends
  - Encuesta Stack Overflow
  - https://githut.info/

Documentación y referencias

Sitio	Sitio web
Python oficial	
Documentación en español	

Sitio	Sitio web
Python Argentina	

Otras referencias

Sitio	Sitio web
Python Guide	The Hitchhiker's Guide to Python
Real Python	Real Python

> **IMPORTANTE**: en los tutoriales y cursos en línea **chequear la versión de Python**.

¿Quiénes usan Python?

Muchas organizaciones han utilizado y utilizan Python para:

- Producción de efectos especiales de películas.
- En sistemas informáticos de la NASA.
- Desarrollo web.
- En ámbito científico.
- Enseñanza de la programación, etc
- + Info:

Características del lenguaje

Es un lenguaje de alto nivel, fácil de aprender. Muy expresivo y legible.

```
random_number = random.randrange(5)
```

```
i_won = False
print("Tenés 2 intentos para adivinar un entre 0 y 4")
tries = 1

while tries < 3 and not i_won:
    entered_number = int(input("Ingresá tu número: "))
    if entered_number == random_number:
        print("Ganaste!")
        i_won = True
    else:
        print("Mmmm ... No, ese número no es... Seguí intentando.")
        tries += 1
if not i_won:
    print("Perdiste :(")
    print(f"El número era: {random_number}")
```

- Sintaxis muy clara

Características del lenguaje (cont.)

- Es **interpretado**, **multiplataforma** y **multiparadigma**: ¿qué significa?
- Posee tipado dinámico y fuerte.
- Tiene un eficiente manejo de estructuras de datos de alto nivel.

Primeros pasos

- Hay intérpretes en línea.
- Descargamos desde el sitio oficial.
- Para **ejecutar** código Python:
  - Usamos la consola de Python: donde se utiliza un modo interactivo y obtener una respuesta por cada línea.
  - Usamos un IDE: como en cualquier otro lenguaje, se escribe el código en un archivo de texto y luego se invoca al intérprete para que lo ejecute.
- +Info

Algunas consideraciones

- Se pueden utilizar entornos virtuales.
  - +Info
- Existe un gestor de paquetes que facilita la instalación de las distintas librerías: pip.
  - +Info

Estamos usando Jupyter notebook

In [ ]:
```
## Adivina adivinador....
import random
random_number = random.randrange(5)
```

```
i_won = False
print("Tenés 2 intentos para adivinar un entre 0 y 4")
tries = 1

while tries < 3 and not i_won:
    entered_number = int(input("Ingresá tu número: "))
    if entered_number == random_number:
        print("Ganaste!")
        i_won = True
    else:
        print("Mmmm ... No, ese número no es... Seguí intentando.")
        tries += 1
if not i_won:
    print("Perdiste :(")
    print(f"El número era: {random_number}")
```

Empecemos por algo más simple

In [ ]:
```
data = 21
print(data)
data ='Hola!'
print(f'{data} ¿Cómo están?')
```

- ¿Algo que llame la atención respecto a otros lenguajes vistos?

- No hay una estructura de programa (tipo program.. begin.. end).
- Las variables no se declaran.
  - Las variables se crean **dinámicamente** cuando se les asigna un valor.
- Las variables pueden cambiar de tipo a lo largo del programa.
  - Python cuenta con **tipado dinámico**.

Un poco más de variables en Python

Vamos el siguiente código. ¿Qué creen que imprime este código?

In [ ]:
```
text_1 = 'Estamos haciendo'
print(f'{Text_1}  diferentes pruebas')
```

Reglas de nombres

- Python hace diferencia entre mayúsculas y minúsculas.
  - Las variables **text_1** y **Text_1** son DOS variables DISTINTAS.
- Los nombre de las variables sólo pueden contener letras, dígitos y _.
- Y **siempre** deben comenzar con letra.
  - Vamos a ver que en algunos casos específicos pueden comenzar con _, pero tienen un significado especial que iremos viendo a lo largo de la cursada.

No pueden usarse ninguna de las palabras claves del lenguaje.

Probar: **help("keywords")**

In [ ]:
```
help("keywords")
```

Asignación de variables

- Las variables permiten referenciar a los **objetos** almacenados en la memoria.
- Asignar un valor a una variable indica que se va a "apuntar" o "referenciar" ese objeto a través de ese nombre de variable.
- Cada objeto tiene asociado **un tipo, un valor y una identidad**.
  - La identidad actúa como una referencia a la posición de memoria del objeto. -Una vez que se crea un objeto, su identidad y tipo no se pueden cambiar.

La identidad de un objeto

- Podemos obtener la identidad de un objeto con la función **id()**.

In [ ]:
```
message1 = "hola"
message2 = message1
message3 = "hola "
print(message1, message2, message3)
print(id(message1), id(message2))
```

Respecto a los nombres de variables

- Existen algunas convenciones que VAMOS a adoptar.
- Si se trata de un nombre compuesto vamos a usar el símbolo "_" para separar. ¿Cómo se conoce esta forma?
- Ejemplo:

```
value = 100
score = 10
game_paused = True
i_won = False
```

- Algunas otras convenciones:
  - Los nombres de variables comienzan con letras minúsculas.
  - No usar nombres tales como "l" u "O" que se pueden confundir con unos y ceros.

Python Enhancement Proposals (PEP)

- Las PEP son documentos que proporcionan información a la comunidad de Python sobre distintas características del lenguaje, novedades en las distintas versiones, guías de codificación, etc.
- La PEP 0 contiene el indice de todas las PEP
- La PEP 20: el Zen de Python...
  - Probar: *import this* desde una consola.

Guías de estilo de codificación

> **"El código es leído muchas más veces de lo que es escrito"** (Guido Van Rossum)

- Están especificadas en la PEP 8
- Hay guías sobre la indentación, convenciones sobre los nombres, etc.
- Algunos IDE chequean que se respeten estas guías.
- Su adopción es MUY importante cuando se comparte el código.

Indentación en Python

- **Indentar el código es una buena práctica de programación**.
- Algo que caracteríza a Python es que la **indentación es obligatoria**.
- ¿Cómo podemos indentar código?

Buscar: ¿qué nos dice la PEP 8 sobre esto?

Observemos nuevamente el primer ejemplo:

In [ ]:
```
## Adivina adivinador....
import random
random_number = random.randrange(5)
i_won = False
print("Tenés 2 intentos para adivinar un entre 0 y 4")
tries = 1

while tries < 3 and not i_won:
    entered_number = int(input("Ingresá tu número: "))
    if entered_number == random_number:
        print("Ganaste!")
        i_won = True
    else:
        print("Mmmm ... No, ese número no es... Seguí intentando.")
        tries += 1
if not i_won:
    print("Perdiste :(")
    print(f"El número era: {random_number}")
```

Los comentarios en Python

- Como ya sabemos, los comentarios NO son procesados por el intérprete.
- Comienzan con el símbolo "#".

In [ ]:
```
# Este es un comentario de una línea.

# Si el comentario
# tiene varias líneas
# repito el símbolo "numeral" en cada línea.
```

- Hay una sección en la PEP 8
- Entre las sugerencias:

- Tratar de no utilizar comentarios en la misma línea, trae confusión. Pero si se hace, separarlo bien y que no sea para comentar cosas obvias, como el siguiente ejemplo:

```
score = score + 1        # Incrementa score
```

Tipos de datos

- ¿Qué tipos de datos vimos en los ejemplos?

- ¿Qué variables encuentran?
- ¿De qué tipo es cada una?

```
## Adivina adivinador....
import random
random_number = random.randrange(5)
i_won = False
print("Tenés 2 intentos para adivinar un entre 0 y 4")
tries = 1
```

- Además de números enteros, booleanos, vimos cadenas de caracteres

```
i_won = False
tries = 1
text_1 = 'Estamos haciendo'
```

¿Qué nos indica un tipo de datos?

- El tipo de datos me indica **qué valores** y **qué operaciones** puedo hacer con una determinada variable.

- Dijimos que Python tiene tipado dinámico y fuerte. ¿Qué significa?

Tipos de datos predefinidos

- Built-In Data Types
  - Números (enteros, flotantes y complejos)
  - Booleanos
  - Cadenas de texto
  - Listas, tuplas, diccionarios y conjuntos.

Números en Python

In [ ]:
```
number1 = 15
number2 = 0o17
number3 = 0xF
#type(number2)
type(number3)
```

- Todas las variables son de tipo **int**.
- Difieren en la forma de expresar el valor:
  - Si lo expresamos como un **octal**, debemos anteponer un **0o** (cero y letra o)
  - Si lo expresamos como un **hexadecimal**, debemos anteponer un **0x**

Más números en Python

- ¿Cómo puedo saber su tipo?

In [ ]:
```
number4 = 0.0001
number5 = 0.1e-3
type(number5)
```

- Todas son variables de tipo **float** que representan los valores reales.

Expresiones numéricas

- Los números pueden utilizarse en expresiones aritméticas utilizando los operadores clásicos: +, -, / y *.
- ¿Cuál creen que es el valor de la variable **division**? y de qué **tipo** es?

In [ ]:
```
number1 = 8
number2 = 2
division = number1 / number2
division
type(division)
```

- La división entre enteros devuelve un **float**.
- Una expresión con números int y float, se convierte a **float**.

Más operadores

In [ ]:
```
number = 9
print(number // 2)
print(number % 2)
print(number ** 2)
```

- Corresponden a la división entera, el resto de la división entera y a la potencia.
- **Buscar en la PEP8:** ¿hay algunas sugerencias respecto a la forma en que se escriben las expresiones y la asignación de variables?

Conversiones explícitas

In [ ]:
```
number_float = 7.8
number_int = 2
result = number_float / number_int
print(result)
int(result)
```

- Las funciones **int()** y **float()** convierten en forma explícita su argumento a tipo int y float.
- Hay otras funciones similares que permiten convertir un argumento a otros tipos de Python que veremos luego.

DESAFÍO 1

> Queremos ingresar un número desde el teclado e imprimir si el número es o no par.

- ¿Cómo sería el pseudocódigo de esto?

```
Ingresar un número desde el teclado
SI es par:
    Mostrar mensaje: "es par"
SINO:
    Mostrar mensaje: "NO es par"
```

¿Cómo ingresamos datos desde el teclado?

La función input

- Permite ingresar datos desde el teclado (más adelante veremos esto con más detalle).
- El tipo de datos ingresado es siempre un **str** (cadena de caracteres), por lo que se tiene que hacer una conversión explícita si no queremos operar con strings.

In [ ]:
```
number = input("Ingresa un número: ")
type(number)
```

¿Qué otra cosa nos falta para resolver el desafío?

La sentencia condicional

In [ ]:
```
number = int(input("Ingresá un número: "))
if number == 3:
    print("Ingresaste un 3!!!")
```

In [ ]:
```
number = int(input("Ingresá un número: "))
if number == 3:
    print("Ingresaste un 3!!!")
else:
    print("NO ingresaste un 3!!!")
```

Ahora... a programar el desafío

In [ ]:
```
# Posible solución

number = int( input("Ingresá un número: "))
```

```
if number % 2  == 0:
    print("Es par")
else:
    print("No es par")
```

DESAFÍO 2

> Queremos ingresar un número desde el teclado e imprimir si es múltiplo de 2, 3 o 5.

Pista: Python tiene otra forma de la sentencia condicional: **if-elif-else**.

In [ ]:
```
month = 3
if month == 1:
    print("Enero")
elif month == 2:
    print("Febrero")
else:
    print("Ups... Se acabaron las vacaciones!!! :()")
```

- ¿**case** en Python? -> **pattern matching**

PEP 636 --> A partir de Python 3.10

El ejemplo anterior en Python >= 3.10

In [ ]:
```
month = 3
match month:
    case 1:
        print("Enero")
    case 2:
         print("Febrero")
    case 3:
        print("Ups... Se acabaron las vacaciones!!! :()")
```

In [ ]:
```
word = "uno"
match word:
    case "uno":
        print("UNO")
    case "dos"  | "tres":
        print("DOS O TRES")
    case _:
        print("Ups.. ninguno de los anteriores")
```

- Si queremos saber qué versión de Python estamos usando:

In [ ]:
```
import sys
sys.version
```

In [ ]:
```
# Solución al DESAFÍO 2
```

Booleanos

- Sólo permite dos únicos valores: **True** y **False**
- Operadores lógicos: **and**, **or** y **not**.

In [ ]:
```
# ¿Qué imprime?
value1 = True
value2 = False
print(value1 and value2, value1 or value2, not value1)
```

Algunas cosas "extrañas"

In [ ]:
```
print(20 or 3)
print(5 and 0)
print(4 or 0 and 3)
```

En Python los booleanos son valores numéricos

- Todo valor **diferente a cero (0)** es **True**.
- Todo valor **igual a cero (0)** es **False**.

- Debemos verificar las precedencias de los operadores:
  - Como podemos ver que el operador **and** tiene mayor precedencia que el **or**.
- Operadores relacionales: ==, !=, >, <, >=,<=

In [ ]:
```
x = 1
y = 2
print(x > y, x != y, x == y)
```

Cadenas de caracteres

- Secuencia de caracteres encerrados entre comillas simples ' ' o comillas dobles " ".

In [ ]:
```
word1 = "letras"
word2 = '123'
word3= "!123abc%$  /%$#"
word2
```

In [ ]:
```
menu =""" Menú de opciones:
            1.- Jugar
            2.- Configurar el juego
            3.- Salir
        """
print(menu)
```

- """ (tres " o ') permiten escribir cadenas de más de una línea.

Operaciones con cadenas de caracteres

In [ ]:
```
name = 'Guido '
surname = "van Rossum"
print(f'{name} {surname} es el "creador" de Python')
```

Repetición de cadenas

In [ ]:
```
line = "*" * 35
menu = """ Menú de opciones:
            1.- Jugar
            2.- Configurar el juego
            3.- Salir
        """
print(line)
print(menu)
print(line)
```

Comparando cadenas

In [ ]:
```
print('Python' == 'Python')
print("Python">"Java")
```

In [ ]:
```
print("Python">"java")
```

DESAFÍO 3

> Dado una letra ingresada por el teclado, queremos saber si es mayúscula o minúscula.

In [ ]:
```
# Una posible solución
letter = input("Ingresar una letra")

if letter >="a" and letter <="z":
    print("Es minúscula")
elif letter >= "A" and letter <= "Z":
    print("Es mayúscula")
else:
    print("NO es una letra")
```

DESAFÍO 4

> Dado un caracter ingresado por el teclado, queremos saber si es una comilla o no.

Con lo visto hasta el momento, ¿hay algún problema?

Secuencias de escape

In [ ]:
```
print("Hola\n\t Empezamos a cursar\n\t dos")
print('Año\"22')
print("Imprimo comilla \" ")
```

In [ ]:
```
# Solución al desafío 4
```

La función len()

- **len():** retorna la cantidad de caracteres de la cadena.

In [ ]:
```
len("Hola")
```

DESAFÍO 5

> Dadas dos cadenas ingresadas desde el teclado, imprimir aquella que tenga más caracteres.

In [ ]:
```
#Posible solución al desafío
```

Cada elemento de la cadena

- Se accede mediante un índice entre [].
- **Comenzando desde 0 (cero).**

In [ ]:
```
word = "Python"
word[0]
```

DESAFÍO 6

> Determinar si una palabra ingresada desde el teclado es un sustantivo propio o no.

IMPORTANTE: solo vamos a controlar si la palabra empieza con una letra mayúscula.

In [ ]:
```
# Posible solución
word = input("Ingresar una palabra: ")

if word[0] >="A" and word[0] <="Z":
    print("Es un sustantivo propio.")
else:
    print("NO es un sustantivo propio.")
```

Recorriendo la cadena

- La sentencia **for**:

In [ ]:
```
word = "casa"
for letter in word:
    print(letter)
```

In [ ]:
```
word = "casa"
for letter in word:
    print(letter, end=" ")
```

DESAFÍO 7

> Escribir un programa que ingrese desde teclado una cadena de caracteres e imprima cuántas letras "a" contiene.

In [ ]:
```
# Posible solución al desafio
word = input("Ingresá una cadena: ")
count_a = 0
for letter in word:
    if letter == "a" or letter == "A":
```

```
        count_a += 1
print(count_a)
```

Empezamos a adaptarnos al estilo "Pythonic"

In [ ]:
```
# Otra posible solución
word = input("Ingresá una cadena: ")
print(word.count("a") + word.count("A"))
```

In [ ]:
```
# Sin tener que repetir count para mayuscula y minuscula
word = input("Ingresá una cadena: ")
print(word.lower().count('a'))
```

Seguimos la próxima ...