Clase_1_Introducción

March 14, 2023

1 Seminario de Lenguajes - Python

1.1 Cursada 2023

1.1.1 Clase 1: conceptos básicos e introducción al lenguaje

2 ¿Con qué términos asociás al lenguaje Python?

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

Resultados

3 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
```

4 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.

5 ¿Por qué hablamos de software libre?

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

6 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
  - https://githut.info/

7 Documentación y referencias

- Sitio Oficial: http://python.org/
- Documentación en español: https://wiki.python.org/moin/SpanishLanguage
- Python Argentina: http://python.org.ar/
- Otras referencias:
  - https://docs.python-guide.org/
  - https://realpython.com/

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

8 ¿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:

9 Características del lenguaje

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

```
numero_aleatorio = random.randrange(5)
gane = False
print("Tenés 2 intentos para adivinar un entre 0 y 4")
intento = 1

while intento < 3 and not gane:
    numero_ingresado = int(input("Ingresa tu número: "))
    if numero_ingresado == numero_aleatorio:
        print("Ganaste!")
        gane = True
    else:
        print("Mmmm ... No.. ese número no es... Seguí intentando.")
        intento += 1
if not gane:
    print("Perdiste :(")
    print("El número era:", numero_aleatorio)
```

- Sintaxis muy clara

10 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.

11 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

12 Algunas consideraciones

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

13 Estamos usando Jupyter notebook

[8]:
```
## Adivina adivinador....
import random
numero_aleatorio = random.randrange(5)
gane = False
print("Tenés 2 intentos para adivinar un entre 0 y 4")
intento = 1

while intento < 3 and not gane:
    numero_ingresado = int(input('Ingresa tu número: '))
    if numero_ingresado == numero_aleatorio:
        print("Ganaste!")
        gane = True
    else:
        print('Mmmm ... No.. ese número no es... Seguí intentando.')
        intento += 1
if not gane:
```

```
    print("Perdiste :(")
    print("El número era:", numero_aleatorio)
```

```
Tenés 2 intentos para adivinar un entre 0 y 4
Ingresa tu número: 2
Ganaste!
```

14 Empecemos por algo más simple

[9]:
```
x = 21
print(x)
x ='hola!'
print(x + '¿Cómo están?')
```

```
21
hola!¿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**

15 Un poco más de variables en Python

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

[10]:
```
texto_1 = 'Estamos haciendo'
print(Texto_1 + ' diferentes pruebas')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 2
      1 texto_1 = 'Estamos haciendo'
----> 2 print(Texto_1 + ' diferentes pruebas')

NameError: name 'Texto_1' is not defined
```

16 Reglas de nombres

- Python hace diferencia entre mayúsculas y minúsculas.
  - Las variables **texto_1** y **Texto_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.

16.0.1 No pueden usarse ninguna de las palabras claves del lenguaje.

Probar **help("keywords")**

[11]:
```
help("keywords")
```

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

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

17 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.

17.1 La identidad de un objeto

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

[12]:
```
a = "hola"
b = a
c = "hola "
#print (a, b, c)
print(id(b), id(a))
```

```
140003835015408 140003835015408
```

18 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.

- Ejemplo:

```
monto_adeudado = 100
valor_de_reembolso = 10
partida_pausada = True
```

- 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.

19 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*

20 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.

21 Indentación en Python

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

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

22 Observemos nuevamente el primer ejemplo:

[13]:
```
## Adivina adivinador....
import random
numero_aleatorio = random.randrange(5)
gane = False
print("Tenés 2 intentos para adivinar un entre 0 y 4")
intento = 1
```

```
while intento < 3 and not gane:
    numero_ingresado = int(input('Ingresa tu número: '))
    if numero_ingresado == numero_aleatorio:
        print("Ganaste!")
        gane = True
    else:
        print('Mmmm ... No.. ese número no es... Seguí intentando.')
        intento += 1
if not gane:
    print("Perdiste :(")
    print("El número era:", numero_aleatorio)
```

```
Tenés 2 intentos para adivinar un entre 0 y 4
Ingresa tu número: 2
Mmmm … No.. ese número no es… Seguí intentando.
Ingresa tu número: 3
Ganaste!
```

23 Los comentarios en Python

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

[ ]:
```
# 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:

```
x = x + 1          # Incrementa x
```

24 Tipos de datos

- ¿Qué tipos de datos vimos en los ejemplos?
- Vimos números enteros, booleanos y cadenas de caracteres

```
gane = False
texto_1 = 'Estamos haciendo'
intento = 1
```

24.1 ¿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?

25 Tipos de datos predefinidos

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

26 Números en Python

[14]:
```
numero_1 = 15
numero_2 = 0o17
numero_3 = 0xF
#type(numero_2)
type(numero_3)
```

[14]:
```
int
```

- 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**

27 Más números en Python

- ¿Cómo puedo saber su tipo?

[15]:
```
numero_4 = 0.0001
numero_5 = 0.1e-3
type(numero_5)
```

[15]:
```
float
```

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

28 Expresiones numéricas

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

[16]:
```
num1 = 8
num2 = 2
divido = num1 / num2

divido
```

[16]: 4.0

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

29 Más operadores

[17]:
```
x = 9
print(x // 2)
print(x % 2)
print(x ** 2)
```

```
4
1
81
```

- 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?

30 Conversiones explícitas

[18]:
```
x = 7.8
y = 2
z = x / y
z
int(z)
```

[18]: 3

- 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.

31 Primer desafío

- 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?

32 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.

[19]:
```
num = input("Ingresa un número: ")
type(num)
```

```
Ingresa un número: 5
```

[19]:
```
str
```

33 ¿Qué otra cosa nos falta para resolver el desafío?

33.1 La sentencia condicional

[20]:
```
num = int(input("Ingresá un número: "))
if num == 3:
    print("Ingresaste un 3!!!")
```

```
Ingresá un número: 3
Ingresaste un 3!!!
```

[21]:
```
num = int(input("Ingresá un número: "))
if num == 3:
    print("Ingresaste un 3!!!")
else:
    print("NO ingresaste un 3!!!")
```

```
Ingresá un número: 5
NO ingresaste un 3!!!
```

34 Ahora... a programar el desafío

[22]:
```
# leer un número desde el teclado e imprimir si el número es o no par.
num = int( input("Ingresá un número: "))

if num % 2  == 0:
    print("Es par")
```

```
else:
    print("No es par")
```

```
Ingresá un número: 5
No es par
```

35 Segundo desafío

- 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**.

[23]:
```
mes = 3
if mes == 1:
    print("Enero")
elif mes == 2:
    print("Febrero")
else:
    print("Ups... Se acabaron las vacaciones!!! :()")
```

```
Ups… Se acabaron las vacaciones!!! :()
```

- ¿**case** en Python?

PEP 636 –> A partir de Python 3.10

36 La solución en Python >= 3.10

[24]:
```
mes = 3
match mes:
    case 1:
        print("Enero")
    case 2:
         print("Febrero")
    case 3:
        print("Ups... Se acabaron las vacaciones!!! :()")
```

```
Ups… Se acabaron las vacaciones!!! :()
```

[25]:
```
cadena = "uno"
match cadena:
    case "uno":
        print("UNO")
    case "dos"  | "tres":
        print("DOS O TRES")
    case _:
        print("Ups.. ninguno de los anteriores")
```

```
UNO
```

- Si queremos saber qué versión de Python estamos usando:

[26]:
```
import sys
sys.version
```

[26]:
```
'3.11.2 (main, Feb  8 2023, 14:49:24) [GCC 9.4.0]'
```

37 Booleanos

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

[27]:
```
# ¿Qué imprime?
x = True
y = False
print(x and y, x or y, not x)
```

```
False True False
```

38 Algunas cosas "extrañas"

[28]:
```
print(20 or 3)
#print(5 and 0)
#print(4 or 0 and 3)
```

```
20
```

38.0.1 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: ==, !=, >, <, >=,<=

[29]:
```
x = 1
y = 2
print(x > y, x != y, x == y)
```

```
False True False
```

39 Cadenas de caracteres

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

[30]:
```
cadena = "letras"
cadena_1 = "123"
```

```
cadena_3 = "!123abc%$  /%$#"
cadena
```

[30]: 'letras'

[32]:
```
menu = """ Menú de opciones:
            1.- Jugar
            2.- Configurar el juego
            3.- Salir
        """
print(menu)
```

```
 Menú de opciones:
            1.- Jugar
            2.- Configurar el juego
            3.- Salir
```

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

40 Operaciones con cadenas de caracteres

[33]:
```
nombre = 'Guido '
apellido = "van Rossum"
print(nombre + apellido)
```

```
Guido van Rossum
```

41 Repetición de cadenas

[34]:
```
linea = "*" * 35
menu = """ Menú de opciones:
            1.- Jugar
            2.- Configurar el juego
            3.- Salir
        """
print(linea)
print(menu)
print(linea)
```

```
***********************************
 Menú de opciones:
            1.- Jugar
            2.- Configurar el juego
            3.- Salir

***********************************
```

42 Comparando cadenas

[35]:
```
print('Python' == 'Python')
print("Python">"Java")
```

```
True
True
```

[36]:
```
print("Python">"python")
```

```
False
```

43 Tercer desafío

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

[37]:
```
letra = input("Ingresar una letra")

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

```
Ingresar una letrad
Es minúscula
```

44 Cuarto desafío

- Dado un caracter ingresado por el teclado, queremos saber si es una comilla o no.
- ¿Hay algún problema?

44.1 Secuencias de escape

[38]:
```
print("Hola\n\t Empezamos a cursar\n\t dos")
print('Año\"22')
print("Imprimo comilla \" ")
```

```
Hola
         Empezamos a cursar
         dos
Año"22
Imprimo comilla "
```

45 La función len()

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

[39]:
```
len("Hola")
```

[39]:
```
4
```

45.1 Quinto desafío

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

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

46 Cada elemento de la cadena

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

[40]:
```
cadena = "Python"
cadena[0]
```

[40]:
```
'P'
```

46.1 Sexto desafío

- Determinar si una palabra ingresada desde el teclado es un sustantivo propio.
  - Solo vamos a controlar si la palabra empieza con una letra mayúscula.

[41]:
```
# Posible solución
palabra = input("Ingresar una palaabra: ")

if palabra[0] >="A" and palabra[0] <="Z":
    print("Es un sustantivo propio.")
```

```
Ingresar una palaabra: Argentina
Es un sustantivo propio.
```

47 Recorriendo la cadena

- La sentencia **for**:

[42]:
```
cadena = "casa"
for car in cadena:
    print(car)
```

```
c
a
```

```
s
a
```

48 Último desafío

- Escribir un programa que ingrese desde teclado una cadena de caracteres e imprima cuántas letras “a” contiene.

[43]:
```
# Posible solución al desafio
cadena = input("Ingresá una cadena: ")
cant_a = 0
for car in cadena:
    if car == "a" or car == "A":
        cant_a += 1
print(cant_a)
```

```
Ingresá una cadena: La casa de Ana
5
```

[44]:
```
# Otra posible solución
cadena = input("Ingresá una cadena: ")
print(cadena.count("a") + cadena.count("A"))
```

```
Ingresá una cadena: La casa de Ana
5
```

49 Seguimos la próxima ...