Clase_1_Introducción

March 14, 2022

1 Seminario de Lenguajes - Python

1.1 Cursada 2022

1.1.1 Clase 1

2 ¿Qué sabemos de Python?

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

Resultados

3 Empezamos con las encuestas ...

Veamos en catedras.linti

```
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.

5 Python es software libre

Esto significa que disponemos de las cuatro libertades que figuran en la definición del software libre:

- La libertad de **usar** el programa, con cualquier propósito (**libertad 0**).
- La libertad de **estudiar** cómo funciona el programa, y adaptarlo a necesidades propias (**libertad 1**).
- La libertad de **distribuir** copias (**libertad 2**).
- La libertad de **mejorar** el programa y **hacer públicas** las mejoras a los demás, de modo que toda la comunidad se beneficie. (**libertad 3**).

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

6 ¿Por qué hablamos de software libre?

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

7 Hablemos de Python ...

- Desarrollado por Guido Van Rossum en el centro de investigación en Matemáticas CWI en Holanda.
- 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/`

8 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 (iremos agregando estas referencias en el entorno del curso):
  - https://docs.python-guide.org/
  - https://realpython.com/

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

9 ¿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:

10 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 5 intentos para adivinar un entre 0 y 99")
intento = 1

while intento < 6 and not gane:
    numero_ingresado = int(input('Ingresa tu número: '))
    if numero_ingresado == numero_aleatorio:
        print('Ganaste! y necesitaste {} intentos!!!'.format(intento))
```

```
        gane = True
    else:
        print('Mmmm ... No.. ese número no es... Seguí intentando.')
        intento += 1
if not gane:
    print('\n Perdiste :(\n El número era: {}'.format(numero_aleatorio))
```

- Sintaxis muy clara

11 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.

12 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

13 Algunas consideraciones

- Se pueden utilizar entornos virtuales.
  - +Info
- Existe un gestor de paquetes que facilita la instalación de las distintas librerías: pip.
  - +Info
- En nuestro caso, van a tener una VM con las instalaciones básicas.

14 Estamos usando Jupyter notebook

[ ]:
```
## Adivina adivinador....
import random
numero_aleatorio = random.randrange(5)
gane = False

print("Tenés 3 intentos para adivinar un entre 0 y 99")
intento = 1
```

```
while intento < 4 and not gane:
    numero_ingresado = int(input('Ingresa tu número: '))
    if numero_ingresado == numero_aleatorio:
        print('Ganaste! y necesitaste {} intentos!!!'.format(intento))
        gane = True
    else:
        print('Mmmm ... No.. ese número no es... Seguí intentando.')
        intento += 1
if not gane:
    print('\n Perdiste :(\n El número era: {}'.format(numero_aleatorio))
```

15 Empecemos por algo más simple

[ ]:
```
x = 21
print(x)
#x ='hola!'
#print(x + '¿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**

16 Un poco más de variables en Python

Vamos el siguiente código

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

- ¿Qué creen que imprime este código?

17 Reglas de nombres

- Python hace diferencia entre mayúsulas 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.
- No pueden usarse ninguna de las palabras claves del lenguaje. Probar **help("keywords")**

18 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 un puntero a la posición de memoria del objeto. -Una vez que se crea un objeto, su identidad y tipo no se pueden cambiar.
- Podemos obtener la identidad de un objeto con la función **id()**.

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

print(id(c), id(a))
```

19 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.

20 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…

21 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.

22 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?

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

23 Observemos el primer ejemplo:

[ ]:
```
## Adivina adivinador....
import random
numero_aleatorio = random.randrange(5)
gane = False

print("Tenés 3 intentos para adivinar un entre 0 y 99")
intento = 1

while intento < 3 and not gane:
    numero_ingresado = int(input('Ingresa tu número: '))
    if numero_ingresado == numero_aleatorio:
        print('Ganaste! y necesitaste {} intentos!!!'.format(intento))
        gane = True
    else:
        print('Mmmm ... No.. ese número no es... Seguí intentando.')
    intento += 1
if not gane:
    print('\n Perdiste :(\n El número era: {}'.format(numero_aleatorio))
```

24 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
```

25 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
```

- ¿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?

26 Tipos de datos

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

27 Números en Python

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

- 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**

28 Más números en Python

- ¿Cómo puedo saber su tipo?

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

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

29 Expresiones numéricas

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

[ ]:
```
x = 8
y = 2
z = x / y
w = z / 2
#print(z)
```

- La división entre enteros devuelve un **float**.

- Una expresión con números int y float, se convierte a **float**.

30 Más operadores

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

31 Conversiones explícitas

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

- 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.

32 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?

33 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.

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

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

34.1 La sentencia condicional

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

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

35 Ahora... a programar el desafío

[ ]:
```
# 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")
```

36 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**.

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

- ¿**case** en Python?

PEP 636 –> Para Python 3.10

37 La solución en Python 3.10

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

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

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

38 Booleanos

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

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

39 Algunas cosas “extrañas”

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

39.0.1 En Python los booleanos son valores numéricos

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

39.0.2 Precedencias

- El operador **and** tiene mayor precedencia que el **or**.
- Operadores relacionales: ==, !=, >, <, >=,<=

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

40 Cadenas de caracteres

- Secuencia de caracteres encerrados entre comillas simples ’ ’ o comillas dobles ” ”.

```
[ ]: cadena = "letras"
     cadena_1 = "123"
     cadena_2 = '123abc'
     cadena_3 = "!123abc%$  /%$#"
     print(cadena_2)
```

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

- ”””(tres ”) permiten escribir cadenas de más de una línea.
- **Buscar:** ¿hay alguna regla en la PEP 8 sobre esto?

41 Operaciones con cadenas de caracteres

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

42 Repetición de cadenas

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

43 Comparando cadenas

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

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

44 Tercer desafío

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

[ ]:
```
# Solución al desafío
letra =
```

[ ]:
```
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")
```

45 Cuarto desafío

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

45.1 Secuencias de escape

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

46 La función len()

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

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

46.1 Quinto desafío

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

[ ]:
```
#Solución al desafío
```

47 Cada elemento de la cadena

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

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

48 Recorriendo la cadena

- La sentencia **for**

[ ]:
```
for car in cadena:
    print(car)
```

49 Sexto desafío

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

[ ]:
```
# Solución al desafio
```

50 Seguimos la próxima ...