Introducción_a_Python

September 4, 2021

1 Proyecto de Software

1.1 Cursada 2021

2 ¿Qué abordaremos en este video?

- Una introducción a Python.
  - Tipos básicos
  - Funciones

3 Programación en el servidor

Roadmap 2021

4 Nosotros usaremos Python

4.1 ¿Por qué?

- Es un lenguaje que en los últimos años ha crecido de manera constante.
  - Stack Overflow Trends
  - `https://githut.info/`

5 Hablemos de Python ...

- Desarrollado por Guido Van Rossum en el centro de investigación en Matemáticas CWI en Países Bajos.
- En febrero se cumplieron 30 años de su aparición.
- El nombre proviene del grupo de cómicos ingleses Monty Python

6 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.

7 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

8 Importante: la legibilidad

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- … The Zen of Python

9 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…

10 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 IDEs chequean que se respeten estas guías.
- Su adopción es MUY importante cuando se comparte el código.

11 Características del lenguaje (cont.)

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

12 Primeros pasos

- Usar intérpretes en línea.
- Descargar intérprete 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 en las guías del Seminario de Python

13 Algunas consideraciones

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

14 Estamos usando Jupyter Lab

[1]:
```
# Adivina adivinador....
import random
numero_aleatorio = random.randrange(5)
gane = False

print("Tenés 3 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! 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))
```

```
Tenés 3 intentos para adivinar un entre 0 y 4
Ingresa tu número: 2
Mmmm … No.. ese número no es… Seguí intentando.
Ingresa tu número: 1
Mmmm … No.. ese número no es… Seguí intentando.

 Perdiste :(
 El número era: 0
```

15 Empezamos de a poco ...

- Las variables no se declaran.
  - Se crean **dinámicamente** cuando se les asigna un valor.
- Pueden cambiar de tipo a lo largo del programa.
  - Python cuenta con **tipado dinámico**
- Las variables permiten referenciar a los objetos almacenados en la memoria.
- 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()**.

[3]:
```
a = "hola"
b = a
c = "hola "
print(a, b, c)

print(id(a), id(b))
```

```
hola hola hola
139774749620592 139774749620592
```

16 Sentencia import

[71]:
```
import string
import random
letras = string.ascii_lowercase
num = random.randrange(4)
num
#print(random.choice(letras))
```

[71]: 2

[72]:
```
from math import sqrt
raiz = sqrt(16)
print(raiz)
```

```
4.0
```

random.choice() vs. sqrt()

¿Por qué no math.sqrt() o choice()?

17 Un poco más ...

[13]:
```
# Se usan triple comillas para cadenas de más de una línea
print("""
    La computadora ha pensado un número en un rango de 0 a 4.
    Tenés 5 intentos para adivinarlo.
    ¿Lo intentás?
""")
```

```
    La computadora ha pensado un número en un rango de 0 a 4.
    Tenés 5 intentos para adivinarlo.
    ¿Lo intentás?
```

[14]:
```
valor = input('Ingresa algo ')
print(type(valor))
```

```
Ingresa algo 7
<class 'str'>
```

18 Cadenas con format

[15]:
```
intento = 3
print('Hola {} !!! Ganaste! y necesitaste {} intentos!!!'.format("claudia",␣
 ↪intento))
```

```
Hola claudia !!! Ganaste! y necesitaste 3 intentos!!!
```

[73]:
```
x = 4
print("{0:2d} {1:3d} {2:4d}".format(x, x*x, x*x*x))
```

```
 4  16   64
```

19 Los f-String

- Fueron introducidos a partir de la versión 3.6.
- Ver la PEP 498
- +Info en la documentación oficial
- Una forma más sencilla de usar el format:

[18]:
```
nombre = "Claudia"
print(f'Hola {nombre} !!! Ganaste! y necesitaste {intento} intentos!!!')
```

```
Hola Claudia !!! Ganaste! y necesitaste 3 intentos!!!
```

[19]:
```
cad1 = "Cadena alineada a izquierda"
cad2 = "Cadena alineada a derecha"
cad3 = "Cadena centrada"
print(f"\n{cad1:<30}\n{cad2:>30}")
print(f"\n{cad3:^30})")
print(f"\n{cad3:*^30}")
```

```
Cadena alineada a izquierda
     Cadena alineada a derecha

       Cadena centrada        )

*******Cadena centrada********
```

20 Importante: ¡la indentación!

[75]:
```
import string
import random
letras = string.ascii_lowercase
letra = random.choice(letras)
if letra  == "A":
    print("Adivinaste")
 else:
    print('Mmmm ... No es una A... Seguí intentando.')
```

```
  File "<tokenize>", line 7
    else:
```

```
    ^
IndentationError: unindent does not match any outer indentation level
```

20.1 Tipos de datos

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

```
gane = False
texto_1 = 'Adivinaste!'
intento = 1
temperatura = 17.5
```

- ¿Qué nos indica un tipo de datos?

21 Colecciones básicas

[31]:
```
cadena = "Python"
lista = [1, 2, "hola", [1, 2], True]
tupla = (1, 2, "hola", lista, (1,2), False)
diccionario = {0: lista, "tupla": tupla}

print(cadena[0])
print(lista[0])
print(tupla[0])
print(diccionario[0])
```

```
P
1
1
[1, 2, 'hola', [1, 2], True]
```

22 Listas, tuplas, diccionarios

- Mutables e inmutables

```
cadena = "Python"
lista = [1, 2, "hola", [1, 2], True]
tupla = (1, 2, "hola", lista, (1,2), False)
diccionario = {"lista": lista, "tupla": tupla}
```

- ¿Modificamos estas secuencias?

[33]:
```
tupla[0]= "A"
tupla
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_16177/3362913016.py in <module>
----> 1 tupla[0]= "A"
      2 tupla

TypeError: 'tuple' object does not support item assignment
```

23 slicing

- El operador **:** permite obtener subsecuencia.
- El formato es **secuencia[inicio:fin]**
- No incluye al elemento cuyo índice es **fin**.
- **[:]** devuelve toda la secuencia.
- Si los índices son negativos, se recorre de derecha a izquierda.

[37]:
```
cadena = "Python"
lista = [1, 2, "hola", [1, 2], True]
#print(cadena[1:])
print(lista[:-2])
```

```
[1, 2, 'hola']
```

24 La cantidad de elementos...

[38]:
```
print(len(cadena))
print(len(lista))
```

```
6
5
```

25 Son todas referencias...

[39]:
```
rock = ["Riff", "La Renga", "La Torre"]
blues = ["La Mississippi", "Memphis"]

musica = rock

rock.append("Divididos")

print(musica)
```

```
['Riff', 'La Renga', 'La Torre', 'Divididos']
```

[40]:
```
print(id(musica))
print(id(rock))
print(id(blues))
```

```
139774749384064
139774749384064
139774749650624
```

25.0.1 Tarea para el hogar...

¿Cómo hacemos para copiar a otra zona de memoria?

26 De texto a listas ...

[42]:
```
palabras = "En esta clase aparecen grandes bandas".split(" ")
palabras
```

[42]:
```
['En e', 'ta cla', 'e aparecen grande', ' banda', '']
```

27 Estructuras de control: sentencias condicionales

- if
- if .. else
- if .. elif.. elif.. else
- A if C else B

[76]:
```
criptos = ["DAI", "USDT"]
cripto = "BTC"
tipo_cripto = "estable" if cripto in criptos  else "cambiante"
print(f"{cripto} es {tipo_cripto}")
```

```
BTC es cambiante
```

IMPORTANTE: Python utiliza la **evaluación con circuito corto**.

[77]:
```
x = 1
y = 0
if True or x/y:
    print("Mmmm raro...")
else:
    print("nada")
```

```
Mmmm raro...
```

28 Estructuras de control: iteraciones

- while
- for .. in

[78]:
```
i = 5
while i >0:
    print(i)
    i -= 1
```

```
5
4
3
2
1
```

[49]:
```
for num in range(2, 5):
    print(num)
```

```
2
3
4
```

[51]:
```
dias = ["domingo", "lunes", "martes", "miércoles", "jueves", "viernes",␣
 ↪"sábado"]
for d in dias:
    print(d)
```

```
domingo
lunes
martes
miércoles
jueves
viernes
sábado
```

29 Definición por comprensión

[52]:
```
cuadrados = [x**2 for x in range(10)]
print(cuadrados)

pares = [x for x in cuadrados if x % 2 == 0]
print(pares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 4, 16, 36, 64]
```

[53]:
```
dicci = dict ([(x, x**2) for x in (2, 4, 6)])
print(dicci)
```

```
{2: 4, 4: 16, 6: 36}
```

30 Conjuntos en Python

- Un conjunto es una colección de datos heterogéna, **desordenada**, **NO indexada** y **sin elementos duplicados**.

[54]:
```
bandas = {"AC/DC", "Metallica", "Greta Van Fleet", "Soda Stéreo", "Los Piojos"}
type(bandas)
```

[54]:
```
set
```

[55]:
```
letras = set("alabanza")
letras
```

[55]:
```
{'a', 'b', 'l', 'n', 'z'}
```

31 Operaciones

[56]:
```
bandas = {"AC/DC", "Metallica", "Greta Van Fleet", "Soda Stéreo", "Los Piojos"}
bandas_nacionales = set(("Soda Stéreo", "La Renga", "Los Piojos"))

print("Foo Fighters" in bandas)

todos = bandas | bandas_nacionales
print(todos)

algunos = bandas & bandas_nacionales
print(algunos)

algunos1 = bandas - bandas_nacionales
print(algunos1)
```

```
False
{'Greta Van Fleet', 'Soda Stéreo', 'AC/DC', 'Metallica', 'La Renga', 'Los
Piojos'}
{'Soda Stéreo', 'Los Piojos'}
{'AC/DC', 'Greta Van Fleet', 'Metallica'}
```

31.1 Tarea para el hogar ...

1. ¿Qué diferencias hay entre $\mathbf{x = \{\}}$ y $\mathbf{x = set()}$?
2. ¿Cómo recorremos un dicionario?

32 Funciones

[57]:
```
import string
import random

def generar_clave(largo_clave):
    clave = ''
    for c in range(largo_clave):
        clave  += random.choice(letras)
    return clave

letras = string.ascii_lowercase
letras += string.ascii_uppercase
letras += string.digits

mi_clave = generar_clave(8)
print(mi_clave)
```

```
llCGgM4F
```

33 Un poco más sobre funciones

[60]:
```
def generar_clave(largo_clave, todo_minusculas = True):
    clave = ''
    for c in range(largo_clave):
        clave  += random.choice(letras)
    if todo_minusculas:
        return clave.lower()
    else:
        return clave

mi_clave = generar_clave(8)
print(mi_clave)
```

```
yvpgiaql
```

- Las funciones pueden tener valores por defecto.
- Estos parámetros siempre se ubican **al final** de la lista de parámetros.
- Más información en documentación oficial sobre funciones

34 ¿Cuándo se evalúan los valores por defecto en los parámetros?

[63]:
```
i = 4
def funcion(x=i):
    print(x)

i = 10
```

```
funcion()
```

```
4
```

35 Variables locales

- Analicemos este ejemplo:

[67]:
```
x = 12
def funcion1():
    temp = x + 1
    print(temp)

def funcion2():

    x = x + 1
    print(x)

funcion2()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
/tmp/ipykernel_16177/2464988638.py in <module>
      9     print(x)
     10
---> 11 funcion2()

/tmp/ipykernel_16177/2464988638.py in funcion2()
      6 def funcion2():
      7
----> 8     x = x + 1
      9     print(x)
     10

UnboundLocalError: local variable 'x' referenced before assignment
```

36 Variables locales y globales

- Uso de **global** y **nonlocal**.

[70]:
```
x = 0
y = 1
def uno():
    x = 10
    y = 101
    def uno_uno():
```

```
        nonlocal x
        #global x
        x = 100
        print(f"En uno_uno: {x} -- {y}")

    uno_uno()
    print(f"En uno: {x} -- {y}")


uno()
print(f"En ppal: {x} -- {y}")
```

```
En uno_uno: 100 -- 101
En uno: 100 -- 101
En ppal: 0 -- 1
```

37 Seguimos en la semana próxima