Capítulo 2

CGI - Python

CGI

- CGI: Common Gateway Interface
  - Un servidor web es comúnmente usado como una entrada a una aplicación
    - Acceso a documentos “legacy”
    - Interacción con base de datos
  - CGI es un acuerdo entre los implementadores de servidores HTTP
    - La idea es integrar scripts y programas de acceso a los datos
  - Uso típico en formularios HTML para construir aplicaciones con bases de datos
- Un CGI se ejecuta siempre en tiempo-real
  - Genera información dinámica
    - A diferencia de los documentos estáticos

CGI

- Un programa CGI es un ejecutable
  - Permite a todo el mundo ejecutar un programa en el servidor
  - Típicamente residen en un directorio “cgi-bin”
  - Puede ser escrito en cualquier lenguaje que se pueda ejecutar en el servidor
    - C/C++, Fortran, Perl, PHP, Python, Java, etcétera
  - CGI en C (por ejemplo) necesitan ser compilados
    - Típicamente existe el directorio “cgi-src”
  - Muchos prefieren escribir CGI scripts
    - Más fácil para debugear, modificar y mantener

Python

- https://www.python.org/
- Creado a principios de la década de 1990
  - Nombre proviene del gusto de Guido van Rossum por Monty Python
- Lenguaje de programación interpretado, dinámico y multiplataforma
  - Enfocado en “legibilidad” del código
  - Multi-paradigma:
    - Programación Orientada al Objeto
    - Programación imperativa
    - Programación funcional
- Licencia de código abierto

Python

- Intérprete
  - Normalmente está instalado en “/usr/bin/python3”

```
jourzua@anakena ~ $ /usr/bin/python3 -V
Python 3.8.5
```

```
√ ~ % /usr/bin/python3 -V
Python 3.7.3
```

Python

- Intérprete

```
√ ~ % /usr/bin/python3
Python 3.7.3 (default, Sep  5 2019, 17:14:41)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license"
>>> print("Hola mundo!")
Hola mundo!
```

Python

- Análisis léxico
  - Un programa de Python es leído por un analizador sintáctico
    - Recibe un flujo de tokens generados por el analizador léxico
- Sintaxis
  - Comentarios comienzan con #
  - También se usa # en la primera o segunda línea para declarar la codificación de caracteres que se usará

Python

- Sintaxis
  - Podemos unir varias líneas físicas en una línea lógica:

```
if 1900 < year < 2100 and 1 <= month <= 12 \
   and 1 <= day <= 31 and 0 <= hour < 24 \
   and 0 <= minute < 60 and 0 <= second < 60:
       return 1
```

  - Una línea que termina en “\” no puede llevar un comentario

Python

- Sintaxis
  - Expresiones entre paréntesis pueden dividirse en más de una línea sin usar “\”:

```
month_names = ['Januari', 'Februari', 'Maart',      # These are the
               'April',   'Mei',      'Juni',       # Dutch names
               'Juli',    'Augustus', 'September',  # for the months
               'Oktober', 'November', 'December']   # of the year
```

  - En este caso sí pueden llevar comentarios

Python

- Sintaxis
  - Indentación: espacio y tabulaciones al principio de una línea determina la agrupación de las instrucciones
    - No se recomienda mezclar espacios y tabulares al principio de las líneas del un mismo programa
- Palabras claves

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

Python

- Sintaxis y comentarios:

```
# this is the first comment
spam = 1  # and this is the second comment
          # ... and now a third!
text = "# This is not a comment because it's inside quotes."
```

Python

- Intérprete:
  - Podemos usar el intérprete como una calculadora con números de tipo int, float
  - Operadores matemáticos +, -, * y / funcionan de la misma forma que en otros lenguajes:

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
```

Python

- Operaciones matemáticas

```
>>> 17 / 3
5.666666666666667
>>> 17 // 3
5
>>> 17 % 3
2
>>> 5 * 3 + 2
17
```

```
>>> 5 ** 2
25
>>> 2 ** 7
128
>>> width = 20
>>> height = 5 * 9
>>> width * height
900
```

Python

- Strings: se puede delimitar por comillas simples o dobles

```
>>> 'spam eggs'  # single quotes
'spam eggs'
>>> 'doesn\'t'  # use \' to escape the single quote...
"doesn't"
>>> "doesn't"  # ...or use double quotes instead
"doesn't"
>>> '"Yes," they said.'
'"Yes," they said.'
>>> "\"Yes,\" they said."
'"Yes," they said.'
>>> '"Isn\'t," they said.'
'"Isn\'t," they said.'
```

Python

- Strings: caracteres especiales

```
>>> '"Isn\'t," they said.'
'"Isn\'t," they said.'
>>> print('"Isn\'t," they said.')
"Isn't," they said.
>>> s = 'First line.\nSecond line.'  # \n means newline
>>> s  # without print(), \n is included in the output
'First line.\nSecond line.'
>>> print(s)  # with print(), \n produces a new line
First line.
Second line.
```

Python

- Strings: raw strings

```
>>> print('C:\some\name')  # here \n means newline!
C:\some
ame
>>> print(r'C:\some\name')  # note the r before the quote
C:\some\name
```

- Strings literales:

```
print("""\
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
""")
```

Python

- Strings: pueden ser concatenados con + y repetidos con *

```
>>> "tangananica" * 2 + "-tanganana"
'tanganani catanganani ca-tanganana'
```

- Podemos tratar los Strings como un arreglo:

```
>>> word = 'Python'
>>> word[0]  # character in position 0
'P'
>>> word[5]  # character in position 5
'n'
```

Python

- Strings como arreglos
  - Podemos usar índices negativos:

```
>>> word[-1]  # last character
'n'
>>> word[-2]  # second-last character
'o'
>>> word[-6]
'P'
```

  - Porciones del arreglo:

```
>>> word[0:2]  # characters from position 0 (included) to 2 (excluded)
'Py'
>>> word[2:5]  # characters from position 2 (included) to 5 (excluded)
'tho'
```

Python

- Strings
  - Índice inicial siempre se incluye y el final se excluye:

```
>>> word[:2] + word[2:]
'Python'
>>> word[:4] + word[4:]
'Python'
```

  - Índice no puede sobrepasar el límite:

```
>>> word[42]  # the word only has 6 characters
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

Python

- Strings
  - Los caracteres son inmutables:

```
>>> word[0] = 'J'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> word[2:] = 'py'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

  - En caso de querer un nuevo String, se debe crear

```
>>> 'J' + word[1:]
'Jython'
>>> word[:2] + 'py'
'Pypy'
```