* **Python introducción**
* **Selección y configuración de Interprete**
  + - Crear proyecto
    - Project interpreter
    - Virtual Enviroment

How To Install Python 3 and Set Up a Local Programming Environment on macOS. (2019).

<https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-local-programming-environment-on-macos>

* **Código prueba1 “hola mundo”**
* **Tipos de variables Python**
  + - Numbers
    - String
    - Boolean
    - List ([] Variable)
    - Tuple (Read only)
    - **Dictionary**
* **Strings**
  + - Concadenar +, lower, upper, islower, isupper, len()
    - index [], .index(“G”), replace(“Giraffe”, “Academy”)
* **Números**
  + - abs(), pow(a,b), max(4,6), min (4, 6), round(3.6)
    - from math import \*
* **Introducir datos input(“introduce un nombre: ”)**
* **Modificar variables int(), float(), string()**
* **Listas** friends = [“john”, “pam”, “kim”], print()
  + - friends[0], friends[-1]
    - rango friends[1:3]
    - friends.extend([nueva\_lista]) unir listas juntas
    - friends.append() -> añadir elemento al final de una lista
    - friends.insert(posición, “elemento”)
    - friends.remove(“elemento”)
    - friends.pop -> remover el último elemento de la lista
    - friends.count(elemento) -> contar n. De ‘elemento’ en lista
    - friends.sort() | friends.reverse()
    - friends2 = friends.copy()
* **Tupla** coordendas(4,5)
* **Funciones** def say\_hi():
* Define función la cual se ejecuta

say\_hi()

* + Parámetros
* **Return**

def potencia(num):

return num\*num

print(potencia)

* **If**
  + Lógicas ‘or’, ‘and’, not()
  + Boolean True, False, <= , >=
* **Diccionario**

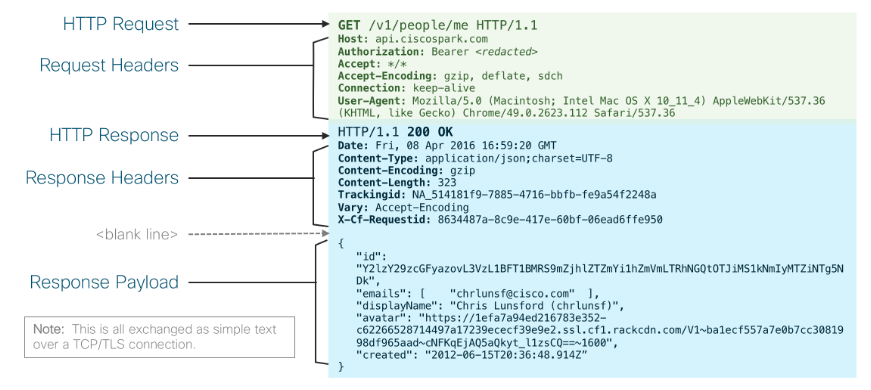
d = {'a': 10, 'b': 20, 'c': 30}  
  
print(d.get('b'))  
print(d.get('z',"no valido"))

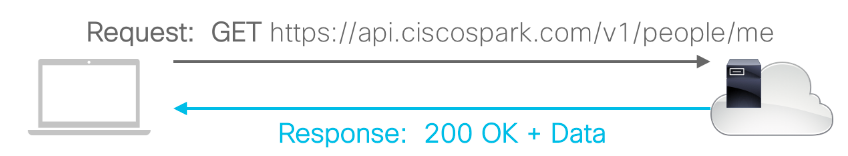
Aprende Python - Curso completo para principiantes (2018). <https://www.youtube.com/watch?v=rfscVS0vtbw&t=343s>

* **Peticiones para consumir API Python**

What is REST? What is an API? (2019). <https://developer.cisco.com/learning/lab/what-are-rest-apis/step/1>

* + - Instalación Requests y urllib2.
    - JSON (equivalente a diccionarios en Python)
    - OBSERACIÓN enfoque desde cliente para creación de servidores uso de FLASK y Django
    - Aplicaciones para consumir API: cURL, Postman, Python Requests, OpenAPI/Swagger
    - Métodos de ‘requests’: GET, PUSH, PUT, DELETE





* + - httpbin.org servicio de peticiones y respuestas para pruebas
    - Método GET envío de parámetros QUERY (?)
      1. Ingresar datos en forma diccionario a parámetro ‘arg’
      2. Get para obtener parámetros tipo json
    - Método POST
      1. Insertar datos dentro de parámetro data en objeto json
      2. Insertar datos mediante solo json
      3. Encabezados
    - Método PUT para actualizar recurso de servidor
    - Método DELETE para borrar recurso del servidor

Python Requests Tutorial: Request Web Pages, Download Images, POST Data, Read JSON, and More (2019)**.** <https://www.youtube.com/watch?v=tb8gHvYlCFs>

# [Taller Consumir APIs en Python](https://www.youtube.com/watch?v=SD_gvyH8tLU&list=PLpOqH6AE0tNguX5SG8HpcD3lfmzWrIn9n) (2017).<https://www.youtube.com/playlist?list=PLpOqH6AE0tNguX5SG8HpcD3lfmzWrIn9n>