Developing an IoT-based Python script for a smart parking system involves several steps. Here's a high-level overview of the process:

1. **Hardware setup :**
   - You'll need IoT hardware like ESP8266 for sensor integration and communication.
   - Ultrasonic sensors, magnetic sensors can be used to detect vehicle presence.

2. **Connectivity :**
   - Set up your IoT device to connect to the internet, typically using Wi-Fi or cellular connectivity.
   - You might need to configure the hardware to connect to your IoT platform.

3. **Iot Platform:**
   - Choose an IoT platform like AWS IoT, Google Cloud IoT, or Microsoft Azure IoT to manage device data and communication.

4. **Python script** :
 Pre install RPi.GPIO and
Create a python script like 'smart_parking_system.py'

```
import RPi.GPIO as GPIO
import time
import paho.mqtt.client as mqtt

# Initialize Ultrasonic Sensor GPIO pins
TRIG = 23
ECHO = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# MQTT Configurations
MQTT_BROKER = "your_mqtt_broker_url"
MQTT_PORT = 1883
MQTT_TOPIC = "smart_parking/parking_spot_1"  # Adjust the topic as needed

# Initialize the MQTT client
client = mqtt.Client()

def get_distance():
    GPIO.output(TRIG, False)
    time.sleep(0.2)

    GPIO.output(TRIG, True)
    time.sleep(0.00001)
```

```
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150  # Speed of sound = 34300 cm/s

    return round(distance, 2)

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

def publish_data():
    while True:
        try:
            distance = get_distance()
            if distance < 10:  # Adjust the threshold for your parking spot
                status = "Occupied"
            else:
                status = "Available"

            data = {"parking_spot": 1, "status": status, "distance": distance}
            client.publish(MQTT_TOPIC, str(data))
            time.sleep(10)  # Adjust the update frequency
        except KeyboardInterrupt:
            GPIO.cleanup()
            client.disconnect()
            break

client.on_connect = on_connect
client.connect(MQTT_BROKER, MQTT_PORT, 60)

try:
    publish_data()
except KeyboardInterrupt:
    pass

GPIO.cleanup()
```

5. **Data processing:**
   - Process data received from sensors to determine parking availability.

- Implement algorithms to detect and update parking spot occupancy.

6. **User interface** :
   - Develop a web or mobile app using a framework like Flask, Django, or a frontend library like React or Angular.
   - Create a user interface to display real-time parking availability and enable user interactions.

7. **Notification :**
   - Implement alerts and notifications for users when parking spots become available or are occupied.