Python Modules:

Sure, here's a brief description of the functions of the next five modules and an example for each:

6. `re`: This module provides support for regular expressions. Some examples include:
   - `re.match(pattern, string)`: Searches for the pattern at the beginning of the string and returns a match object.
   - `re.search(pattern, string)`: Searches for the pattern in the string and returns a match object if found.
   - `re.findall(pattern, string)`: Returns all non-overlapping matches of the pattern in the string.

   Example:
   ```python
   import re

   pattern = r"spam"
   string = "spam and eggs"

   match_obj = re.match(pattern, string)
   print(match_obj.group())
   # Output: "spam"

   match_obj = re.search(pattern, string)
   print(match_obj.group())
   # Output: "spam"

   matches = re.findall(pattern, string)
   print(matches)
   # Output: ["spam"]
   ```

7. `csv`: This module provides functions for working with Comma Separated Value (CSV) files. Some examples include:
   - `csv.reader(file)`: Returns a reader object that can be used to iterate over the rows of a CSV file.
   - `csv.writer(file)`: Returns a writer object that can be used to write rows to a CSV file.
   - `csv.DictReader(file)`: Returns a reader object that can be used to iterate over the rows of a CSV file as dictionaries.

   Example:
   ```python
   import csv
   ```

```python
# Read a CSV file
with open("data.csv", "r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)

# Write to a CSV file
with open("data.csv", "w") as file:
    writer = csv.writer(file)
    writer.writerow(["Name", "Age", "Gender"])
    writer.writerow(["Alice", 25, "F"])
    writer.writerow(["Bob", 30, "M"])

# Read a CSV file as dictionaries
with open("data.csv", "r") as file:
    reader = csv.DictReader(file)
    for row in reader:
        print(row["Name"], row["Age"], row["Gender"])
```

8. `json`: This module provides functions for working with JSON data. Some examples include:
   - `json.dumps(obj)`: Returns a JSON string representation of the specified object.
   - `json.loads(string)`: Parses a JSON string and returns a Python object.
   - `json.dump(obj, file)`: Writes a JSON string representation of the specified object to a file.

   Example:
   ```python
   import json

   # Convert a Python object to a JSON string
   data = {"name": "Alice", "age": 25}
   json_str = json.dumps(data)
   print(json_str)

   # Parse a JSON string and convert it to a Python object
   json_str = '{"name": "Bob", "age": 30}'
   data = json.loads(json_str)
   print(data["name"], data["age"])

   # Write a Python object to a JSON file
   data = {"name": "Charlie", "age": 35}
   with open("data.json", "w") as file:
       json.dump(data, file)
```

```
# Read a Python object from a JSON file
with open("data.json", "r") as file:
    data = json.load(file)
    print(data["name"], data["age"])
```

9. `requests`: This module provides functions for sending HTTP requests. Some examples include:
   - `requests.get(url