The `time` module in Python provides functionality for working with time, including measuring time intervals, formatting dates, and pausing execution. Here are some key aspects of the `time` module along with examples:

### Getting Current Time:

```python
import time

# Get current time in seconds since the epoch
current_time = time.time()

print("Current Time (seconds since epoch):", current_time)
```

### Formatting Time:

```python
import time

# Get current time struct
current_time_struct = time.localtime()

# Format time as a string
formatted_time = time.strftime("%Y-%m-%d %H:%M:%S", current_time_struct)

print("Formatted Time:", formatted_time)
```

### Sleeping:

The `sleep` function can be used to pause the execution of a program for a specified number of seconds:

```python
import time

print("Start")
time.sleep(2)  # Pause for 2 seconds
print("End")
```

### Measuring Time Intervals:

```python
import time

# Record start time
start_time = time.time()
```

```
# Some operation or code to measure
for _ in range(1000000):
    pass

# Record end time
end_time = time.time()

# Calculate elapsed time
elapsed_time = end_time - start_time

print("Elapsed Time:", elapsed_time, "seconds")
```

### Working with Struct Time:

The `time` module represents time as a struct time object, and you can manipulate it:

```python
import time

# Get current time struct
current_time_struct = time.localtime()

# Access individual components of the time
year = current_time_struct.tm_year
month = current_time_struct.tm_mon
day = current_time_struct.tm_mday

print("Year:", year)
print("Month:", month)
print("Day:", day)
```

### Parsing Time Strings:

```python
import time

# Parse a time string to a struct time
time_string = "2023-12-07 15:30:00"
parsed_time_struct = time.strptime(time_string, "%Y-%m-%d %H:%M:%S")

print("Parsed Time Struct:", parsed_time_struct)
```

### Clock Time:

The `time` module also provides a `clock` function for measuring CPU time:

```python
import time

# Record start time
start_cpu_time = time.clock()

# Some CPU-intensive operation
for _ in range(1000000):
    pass

# Record end time
end_cpu_time = time.clock()

# Calculate CPU time
cpu_time = end_cpu_time - start_cpu_time

print("CPU Time:", cpu_time, "seconds")
```

Keep in mind that the `time` module deals with wall-clock time, and for more precise measurements or performance profiling, you might want to consider the `timeit` module or other profiling tools.

These examples cover some of the basic functionalities provided by the `time` module. For more details and additional functions, refer to the official Python documentation for the `time` module: [time — Time access and conversions](https://docs.python.org/3/library/time.html).