```
.
├── README.md
├── agents.yaml
├── crypto_market.py
├── ddgs.py
├── forex.py
├── llama_crew
│   ├── GenericAgents.ipynb
│   ├── agents
│   │   ├── __init__.py
│   │   ├── agent.py
│   │   └── loader.py
│   ├── chat
│   │   ├── __init__.py
│   │   └── utils.py
│   ├── helper.py
│   └── tools
│       ├── __init__.py
│       ├── sample_tools.py
│       └── search_tools.py
├── sample_call_agent.py
├── sample_task.py
└── tools.yaml
5 directories, 18 files
```

agents.yaml

```
agents:
  - name: mathematician
    role: "Answers mathematics questions"
    prompt: "You are an expert in mathematics, you will answer questions
related to maths."
    tools: [add, subtract, multiply, divide]
    verbose: true
  - name: online_research
    role: "Expert in online search"
    prompt: "You are an expert in online search"
    tools: [search, wikipedia_summary, wikipedia_page]
    verbose: true
  - name: oracle
    role: "Answers questions about people's life and death"
    prompt: "You are the Oracle and you make up stories about people's
life and death."
    verbose: true
  - name: python_coder
    role: "executes python code and returns the results as string"
    prompt: "You are a python programmer"
    tools: repl
```

```
    verbose: true
  - name: crypto_trader
    role: "Answers questions regarding crypto"
    prompt: "You are a crypto trader expert"
    tools: get_top_cryptocurrencies_current_data
  - name: trader
    role: "Answers questions regarding trading"
    prompt: "You are a trader expert"
    tools: [get_forex_exchange_rates, get_stock_price]
```

tools.yaml

```
tools:
  - name: add
    module: llama_crew.tools.sample_tools
    function: add
  - name: subtract
    module: llama_crew.tools.sample_tools
    function: subtract
  - name: multiply
    module: llama_crew.tools.sample_tools
    function: multiply
  - name: divide
    module: llama_crew.tools.sample_tools
    function: divide
  - name: search
    module: llama_crew.tools.search_tools
    function: search_ddg
  - name: repl
    module: llama_crew.tools.sample_tools
    function: repl
  - name: wikipedia_summary
    module: llama_crew.tools.sample_tools
    function: get_wikipedia_summary
  - name: wikipedia_page
    module: llama_crew.tools.sample_tools
    function: get_wikipedia_page
  - name: get_top_cryptocurrencies_current_data
    module: llama_crew.tools.sample_tools
    function: get_top_cryptocurrencies
  - name: get_forex_exchange_rates
    module: llama_crew.tools.sample_tools
    function: get_forex_exchange_rates
  - name: get_stock_price
    module: llama_crew.tools.sample_tools
    function: get_stock_price
```

sample_task.py

```
#!/usr/bin/env python
# coding: utf-8

import argparse
from llama_crew.helper import get_openai_api_key
OPENAI_API_KEY = get_openai_api_key()

import nest_asyncio
nest_asyncio.apply()

import yaml
from llama_crew.tools import Tool, load_tools_config
from llama_index.llms.openai import OpenAI
from llama_crew.agents.agent import Orchestrator
from llama_crew.agents.loader import load_agents

# get current path of the file
import os
current_path = os.path.dirname(os.path.abspath(__file__))
defaults = {"tools_config": os.path.join(current_path,'tools.yaml'),
"agents_config": os.path.join(current_path,'agents.yaml')}

parser = argparse.ArgumentParser(description='Run the orchestrator')
parser.add_argument('--tools_config', type=str,
default=defaults["tools_config"], help='Path to the tools configuration
file')
parser.add_argument('--agents_config', type=str,
default=defaults["agents_config"], help='Path to the agents
configuration file')
parser.add_argument("--verbose", action="store_true", help="Print out
the responses from the agents and the evaluation of the responses.")
# The query to send to the orchestrator, take all the rest of the
arguments as the query even when not enclosed in quotes
parser.add_argument('query', nargs=argparse.REMAINDER, help='The query
to send to the orchestrator')
args = parser.parse_args()

tools_config = load_tools_config(args.tools_config)
all_tools = [Tool(config) for config in tools_config["tools"]]
llm = OpenAI(model="gpt-3.5-turbo")

agents_config = yaml.safe_load(open(args.agents_config, 'r'))


agents = load_agents(llm, agents_config, all_tools)


# agents.keys()

# agents["oracle"].query(" When will I get married?")
```

```
# agents["oracle"].state

# agents["mathematician"].query("How much is 10 / 3 * 3")

# agents["mathematician"].state

# agents['ceo_expert'].query("What is the best way to increase
revenue?")


director = Orchestrator(llm, agents, agents_config=agents_config,
verbose=args.verbose)

# director.query("Ask the oracle When will I get married?")

# director.query("How much is 10 / 3 * 3")

# director.query("Today is May 20th, What was the dollar price on
ethereum back in May 1st 2024?")

# director.query("Use python requests to get the current price of
bitcoin and print it out.")


# director.query("list all the files in the current directory")

# director.query("how much disk space do I have left on my home
directory?")

director.query(" ".join(args.query))
```