NEON CODES

ATLANTIS.py LIBRARY

TERMINAL BASED APPLICATION LIBRARY

Objective

Atlantis.py is a python library designed to assist with the development of small scale terminal based applications (Portals).
It has inbuilt methods that provide extreme ease of use and possesses scope for versatility.

Workings

The Atlantis library allows the creation of commands that can be used in a portal session. These commands can be invoked withing the session by entering the respective *invoker number*

Setup

1. Start by importing the Atlantis library

```
import atlantis
```

2. Create an instance of the Atlantis Engine

```
portalEngine = atlantis.Engine()
```

3. Setup the Engine by referencing a custom setup function to the `portalEngine.setup` Property

```
def setupEngine():
    # your code here
    #Idea: start by adding functions for your application
    pass
portalEngine.setup = setupEngine
```

4. Initiate the Engine

```
portalEngine.activatePortal()
```

Methods

- **addCommand(*command*)**

Adds a command to the list of executable commands from within the portal

The format for adding a command is as follows

```
{
    "name": "command  Name",
    "desc": "Description of the command",
    "func": lambda : print("hello world")
}
```

So to add a command, just pass in the dictionary into the method as follows

```
portalEngine.addCommand({"name": "command  Name", "desc": "Description of the command","func": lambda : print("hello world") })
```

- **listCommands()**

Lists the existing portal commands in a presentable manner

- **runCommand(*command_number*)**

Enter the *invoker* number of the command in order to run it

```
portalEngine.runCommand(2)
```

- **setup()**

Empty mutable method that can be modified to run code before initiating the prompt session. One good way to use this method is to add all the portal commands as part of the setup function:

```
def setupEngine():
    portalEngine.addCommand({"name": "command  Name", "desc": "Description of the command","func": lambda : print("hello world") })

portalEngine.setup = setupEngine
```

- **loop(*f=function*)**

This method runs over and over again throughout the prompt session after every input within the session.
A function argument can be passed that also runs whenever a command is inputted

- **activatePortal()**

Initiates a portal session

- **deactivatePortal(*q=bool*)**

Closes the ongoing portal session.
The user can be prompted for confirmation, if the argument in the method is set to true.

Utilities

The Atlantis library has a builtin list of utility functions that are ready to use in your programs. All of these can be accessed with

`portalEngine.utils`

- **prompt(text, newline=bool)**

Prompt the user for a given text.
Optional newline.

- **boolPrompt(text, trueText, FalseText)**

Prompt the user for a boolean response. If the user responds with the value of the true text, the function returns true; similarly for the false text.

Preferences

Adjust the Atlantis engine as per your varied need by tweaking the preferences set up in the engine. This preferences can be accessed by

```
portalEngine.preferences
```

- **RUN_HELP_ON_STARTUP**

Lists down the executable commands in the portal on the initiation of the portal session.

Index

Portal Session: When the terminal based application begins, a Portal session is said to be initiated

Invoker number: When a new command is added to the list of portal commands. It is given a unique number.