PROJECT REPORT: 01.05.2017

PROJECT STATUS : around 80%

Accomplished tasks:

1. Python server:
    - Managing clients
    - Managing players
    - Registering games
    - Handling game master
    - Passing messages
2. Game master:
    - Maintaining games
    - Starting games
    - Placing pieces
    - Adding players
    - Handling messages
    - Finishing the games
3. Client
    - Connecting to the existing game
    - Closing connection
    - Managing messages
4. Player
    - Parsing messages
    - Moving on the board
    - Discovering the pieces
    - Playing with strategy
5. Messages
    - All the messages required to play the game correctly
6. Strategy
    - Strategy in which players are taking all actions
7. Information
    - Base class with the information about the game
8. Unit Tests
    - Every function is tested to check its correctness

sys


```
This module provides access to some objects used or maintained by the
interpreter and to functions that interact strongly with the interpreter.

Dynamic objects:

argv -- command line arguments; argv[0] is the script pathname if known
path -- module search path; path[0] is the script directory, else ''
modules -- dictionary of loaded modules

displayhook -- called to show results in an interactive session
excepthook -- called to handle any uncaught exception other than SystemExit
  To customize printing in an interactive session or to install a custom
  top-level exception handler, assign other functions to replace these.

exitfunc -- if sys.exitfunc exists, this routine is called when Python exits
  Assigning to sys.exitfunc is deprecated; use the atexit module instead.

stdin -- standard input file object; used by raw_input() and input()
stdout -- standard output file object; used by the print statement
stderr -- standard error object; used for error messages
  By assigning other file objects (or objects that behave like files)
  to these, it is possible to redirect all of the interpreter's I/O.

last_type -- type of last uncaught exception
last_value -- value of last uncaught exception
last_traceback -- traceback of last uncaught exception
  These three are only available in an interactive session after a
  traceback has been printed.

exc_type -- type of exception currently being handled
exc_value -- value of exception currently being handled
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
exc_traceback -- traceback of exception currently being handled
  The function exc_info() should be used instead of these three,
  because it is thread-safe.

Static objects:

float_info -- a dict with information about the float inplementation.
long_info -- a struct sequence with information about the long implementation.
maxint -- the largest supported integer (the smallest is -maxint-1)
maxsize -- the largest supported length of containers.
maxunicode -- the largest supported character
builtin_module_names -- tuple of module names built into this interpreter
version -- the version of this interpreter as a string
version_info -- version information as a named tuple
hexversion -- version information encoded as a single integer
copyright -- copyright notice pertaining to this interpreter
platform -- platform identifier
executable -- absolute path of the executable binary of the Python interpreter
prefix -- prefix used to find the Python library
exec_prefix -- prefix used to find the machine-specific Python library
float_repr_style -- string indicating the style of repr() output for floats
__stdin__ -- the original stdin; don't touch!
__stdout__ -- the original stdout; don't touch!
__stderr__ -- the original stderr; don't touch!
__displayhook__ -- the original displayhook; don't touch!
__excepthook__ -- the original excepthook; don't touch!

Functions:

displayhook() -- print an object to the screen, and save it in __builtin__._
excepthook() -- print an exception and its traceback to sys.stderr
exc_info() -- return thread-safe information about the current exception
exc_clear() -- clear the exception state for the current thread
exit() -- exit the interpreter by raising SystemExit
getdlopenflags() -- returns flags to be used for dlopen() calls
getprofile() -- get the global profiling function
getrefcount() -- return the reference count for an object (plus one :-)
getrecursionlimit() -- return the max recursion depth for the interpreter
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

getsizeof() -- return the size of an object in bytes
gettrace() -- get the global debug tracing function
setcheckinterval() -- control how often the interpreter checks for events
setdlopenflags() -- set the flags to be used for dlopen() calls
setprofile() -- set the global profiling function
setrecursionlimit() -- set the max recursion depth for the interpreter
settrace() -- set the global debug tracing function

Functions

__displayhook__ = displayhook(...)

```
displayhook(object) -> None

Print an object to sys.stdout and also save it in __builtin__._
```

__excepthook__ = excepthook(...)

```
excepthook(exctype, value, traceback) -> None

Handle an exception by displaying it with a traceback on sys.stderr.
```

call_tracing(...)

```
call_tracing(func, args) -> object

Call func(*args), while tracing is enabled.  The tracing state is
saved, and restored afterwards.  This is intended to be called from
a debugger from a checkpoint, to recursively debug some other code.
```

callstats(...)

```
callstats() -> tuple of integers

Return a tuple of function call statistics, if CALL_PROFILE was defined
when Python was built.  Otherwise, return None.

When enabled, this function returns detailed, implementation-specific
details about the number of function calls executed. The return value is
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    a 11-tuple where the entries in the tuple are counts of:
    0. all function calls
    1. calls to PyFunction_Type objects
    2. PyFunction calls that do not create an argument tuple
    3. PyFunction calls that do not create an argument tuple
       and bypass PyEval_EvalCodeEx()
    4. PyMethod calls
    5. PyMethod calls on bound methods
    6. PyType calls
    7. PyCFunction calls
    8. generator calls
    9. All other calls
    10. Number of stack pops performed by call_function()
```

displayhook(...)

```
    displayhook(object) -> None

    Print an object to sys.stdout and also save it in __builtin__._
```

exc_clear(...)

```
    exc_clear() -> None

    Clear global information on the current exception.  Subsequent calls to
    exc_info() will return (None,None,None) until another exception is raised
    in the current thread or the execution stack returns to a frame where
    another exception is being handled.
```

exc_info(...)

```
    exc_info() -> (type, value, traceback)

    Return information about the most recent exception caught by an except
    clause in the current stack frame or in an older stack frame.
```

excepthook(...)

```
    excepthook(exctype, value, traceback) -> None
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Handle an exception by displaying it with a traceback on sys.stderr.

exit(...)

exit([status])

Exit the interpreter by raising SystemExit(status).
If the status is omitted or None, it defaults to zero (i.e., success).
If the status is an integer, it will be used as the system exit status.
If it is another kind of object, it will be printed and the system
exit status will be one (i.e., failure).

getcheckinterval(...)

getcheckinterval() -> current check interval; see setcheckinterval().

getdefaultencoding(...)

getdefaultencoding() -> string

Return the current default string encoding used by the Unicode
implementation.

getdlopenflags(...)

getdlopenflags() -> int

Return the current value of the flags that are used for dlopen calls.
The flag constants are defined in the ctypes and DLFCN modules.

getfilesystemencoding(...)

getfilesystemencoding() -> string

Return the encoding used to convert Unicode filenames in
operating system filenames.

getprofile(...)

getprofile()

Return the profiling function set with sys.setprofile.

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

See the profiler chapter in the library manual.

getrecursionlimit(...)

getrecursionlimit()

Return the current value of the recursion limit, the maximum depth
of the Python interpreter stack. This limit prevents infinite
recursion from causing an overflow of the C stack and crashing Python.

getrefcount(...)

getrefcount(object) -> integer

Return the reference count of object. The count returned is generally
one higher than you might expect, because it includes the (temporary)
reference as an argument to getrefcount().

getsizeof(...)

getsizeof(object, default) -> int

Return the size of object in bytes.

gettrace(...)

gettrace()

Return the global debug tracing function set with sys.settrace.
See the debugger chapter in the library manual.

setcheckinterval(...)

setcheckinterval(n)

Tell the Python interpreter to check for asynchronous events every
n instructions. This also affects how often thread switches occur.

setdlopenflags(...)

setdlopenflags(n) -> None

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Set the flags used by the interpreter for dlopen calls, such as when the
interpreter loads extension modules. Among other things, this will enable
a lazy resolving of symbols when importing a module, if called as
sys.setdlopenflags(0). To share symbols across extension modules, call as
sys.setdlopenflags(ctypes.RTLD_GLOBAL). Symbolic names for the flag modules
can be either found in the ctypes module, or in the DLFCN module. If DLFCN
is not available, it can be generated from /usr/include/dlfcn.h using the
h2py script.

setprofile(...)

setprofile(function)

Set the profiling function. It will be called on each function call
and return. See the profiler chapter in the library manual.

setrecursionlimit(...)

setrecursionlimit(n)

Set the maximum depth of the Python interpreter stack to n. This
limit prevents infinite recursion from causing an overflow of the C
stack and crashing Python. The highest possible limit is platform-
dependent.

settrace(...)

settrace(function)

Set the global debug tracing function. It will be called on each
function call. See the debugger chapter in the library manual.

Data

__stderr__ = <open file '<stderr>', mode 'w'>
__stdin__ = <open file '<stdin>', mode 'r'>

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

__stdout__ = <open file '<stdout>', mode 'w'>
api_version = 1013
argv = ['/usr/bin/pydoc', '-w', 'sys']
builtin_module_names = ('__builtin__', '__main__', '_ast', '_codecs', '_sre', '_symtable', '_warnings', '_weakref', 'errno', 'exceptions', 'gc', 'imp', 'marshal', 'posix', 'pwd', 'signal', 'sys', 'thread', 'xxsubtype', 'zipimport')
byteorder = 'little'
copyright = 'Copyright (c) 2001-2015 Python Software Foundati...ematisch Centrum, Amsterdam.\nAll Rights Reserved.'
dont_write_bytecode = False
exc_value = TypeError("<module 'sys' (built-in)> is a built-in module",)
exec_prefix = '/System/Library/Frameworks/Python.framework/Versions/2.7'
executable = '/usr/bin/python'
flags = sys.flags(debug=0, py3k_warning=0, division_warn...unicode=0, bytes_warning=0, hash_randomization=0)
float_info = sys.float_info(max=1.7976931348623157e+308, max_...epsilon=2.220446049250313e-16, radix=2, rounds=1)
float_repr_style = 'short'
hexversion = 34015984
long_info = sys.long_info(bits_per_digit=30, sizeof_digit=4)
maxint = 9223372036854775807
maxsize = 9223372036854775807
maxunicode = 65535
meta_path = []
modules = {'UserDict': <module 'UserDict' from '/System/Library/Framewo...amework/Versions/2.7/lib/python2.7/UserDict.pyc'>, '__builtin__': <module '__builtin__' (built-in)>, '__main__': <module '__main__' from '/usr/bin/pydoc2.7'>, '_abcoll': <module '_abcoll' from '/System/Library/Framewor...ramework/Versions/2.7/lib/python2.7/_abcoll.pyc'>, '_codecs': <module '_codecs' (built-in)>, '_collections': <module '_collections' from

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

'/System/Library/Fra...s/2.7/lib/python2.7/lib-dynload/_collections.so'>, '_functools': <module '_functools' from '/System/Library/Frame...ons/2.7/lib/python2.7/lib-dynload/_functools.so'>, '_heapq': <module '_heapq' from '/System/Library/Framework...ersions/2.7/lib/python2.7/lib-dynload/_heapq.so'>, '_locale': <module '_locale' from '/System/Library/Framewor...rsions/2.7/lib/python2.7/lib-dynload/_locale.so'>, '_osx_support': <module '_osx_support' from '/System/Library/Fra...ork/Versions/2.7/lib/python2.7/_osx_support.pyc'>, ...}
path = ['.', '/System/Library/Frameworks/Python.framework/Versions/2.7/bin', '/Library/Python/2.7/site-packages/pip-9.0.1-py2.7.egg',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python27.zip',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-darwin',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-mac',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-mac/lib-scriptpackages',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-tk',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-old',
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-dynload',
'/Library/Python/2.7/site-packages',
'/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python',
'/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python/PyObjC']
path_hooks = [<type 'zipimport.zipimporter'>]
path_importer_cache = {'/Library/Python/2.7/site-packages': None, '/Library/Python/2.7/site-packages/pip-9.0.1-py2.7.egg': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/Extras/lib/python/PyObjC': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/bin': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/encodings': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-dynload': None,
'/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-old': <imp.NullImporter object>, ...}

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

platform = 'darwin'
prefix = '/System/Library/Frameworks/Python.framework/Versions/2.7'
py3kwarning = False
stderr = <open file '<stderr>', mode 'w'>
stdin = <open file '<stdin>', mode 'r'>
stdout = <open file '<stdout>', mode 'w'>
subversion = ('CPython', '', '')
version = '2.7.10 (default, Feb 6 2017, 23:53:20) \n[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)]'
version_info = sys.version_info(major=2, minor=7, micro=10, releaselevel='final', serial=0)
warnoptions = []

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com