

Python

Lecture 9

© Jeff Parker

Fall, 2018

PAGE 3	DEPARTMENT	COURSE	DESCRIPTION	PREREQS
	COMPUTER SCIENCE	CPSC 432	INTERMEDIATE COMPILER DESIGN, WITH A FOCUS ON DEPENDENCY RESOLUTION.	CPSC 432

Tonight's outline

Review Homework

Garbage Collection

Object Oriented Programming

Viewing Method invocation with Debugger

Magic Methods

Inheritance

How to approach problem

I'm going to spend class time reviewing the problem
How can you solve this with least amount of effort?

<http://www.catb.org/jargon/html/H/hacker.html>

hacker: n.

[originally, someone who makes furniture with an axe]

Hacker vs Cracker

<http://www.catb.org/~esr/faqs/hacker-howto.html>

ESRaymond - How to become a hacker

The Problem

```
def __init__(self, hour, minute, sec):  
    self.hour    = hour  
    self.minute = minute  
    self.second = sec
```

```
def main():  
    start = Time(39, 75, 00)  
    start.print_time()
```

39:75:00

Changing display

```
def __str__(self):  
    return '%.2d:%.2d:%.2d' %  
        (self.hour, self.min, self.sec)
```

Problems

Print seconds

Prints leading 0 in 01:23:45

Need to print AM/PM

First hour in each half is called 12 rather than zero

Changing display

```
def __str__(self):  
    return '%.2d:%.2d' %  
        (self.hour, self.minutes)
```

Problems

~~Print seconds~~

Prints leading 0 in 01:23:45

Need to print AM/PM

First hour in each half is called 12 rather than zero

Changing display

```
def __str__(self):  
    return '%d:%.2d' %  
        (self.hour, self.minutes)
```

Problems

~~Print seconds~~

~~Prints leading 0 in 01:23:45~~

Need to print AM/PM

First hour in each half is called 12 rather than zero

Changing display

```
if (self.hour >= 12):  
    period = 'PM'  
  
else:  
    period = 'AM'
```

See different solution in Notebook

Problems

~~Print seconds~~

~~Prints leading 0 in 01:23:45~~

~~Need to print AM/PM~~

First hour in each half is called 12 rather than zero

Changing display

```
adjusted_hour = self.hour % 12  
if (adjusted_hour == 0):  
    adjusted_hour = 12
```

We cannot mess with hour

Needed to distinguish 1 AM from 1 PM

Problems

~~Print seconds~~

~~Prints leading 0 in 01:23:45~~

~~Need to print AM/PM~~

~~First hour in each half is called 12 rather than zero~~

Changing display

```
def __str__(self):
    if (self.hour >= 12):
        period = 'PM'
    else:
        period = 'AM'
    adjusted_hour = self.hour % 12
    if (adjusted_hour == 0):
        adjusted_hour = 12
    return '%d:%.2d %s' %
           (adjusted_hour, self.minute, period)
```

Comments

```
def __str__(self):
    # AM or PM? - jdp
    if (self.hour >= 12):
        period = 'PM'
    else:
        period = 'AM'
    # Civilian time - jdp
    adjusted_hour = self.hour % 12
    if (adjusted_hour == 0):
        adjusted_hour = 12
    return '%d:%.2d %s' %
           (adjusted_hour, self.minute, period)
```

Always sign edits in shared code

Clipping Hours

```
def __init__(self, hour, minute, sec):  
    self.hour    = hour  
    self.minute = minute  
    self.second = sec
```

```
def main():  
    start = Time(39, 75, 00)  
    start.print_time()
```

39:75:00

Is this a real problem?

Can't I throw an exception if asked to create an illegal time?

Is this a real problem?

Can't I throw an exception if asked to create an illegal time?

I start watching a movie at 10:45

When it is over, I wonder what time it is.

Something every clock deals with once a day.

Don't throw a hissy fit: deal with it

Clipping hours to < 24

What needs to change? What methods are there?

```
def __init__(self, hour=0, minute=0, second=0):  
def __str__(self):  
def print_time(self):  
def time_to_int(self):  
def is_after(self, other):  
def is_valid(self):  
def __add__(self, other):  
def __radd__(self, other):  
def add_time(self, other):  
def increment(self, seconds):  
def int_to_time(seconds):
```

Some only report status

Some only report status - Display string, convert to int, compare,
Test for validity

```
def __str__(self):  
def print_time(self):  
def time_to_int(self):  
def is_after(self, other):  
def is_valid(self):
```

Some only report status

Some only report status - Display string, convert to int, compare,
Test for validity

```
def __str__(self):  
def print_time(self):  
def time_to_int(self):  
def is_after(self, other):  
def is_valid(self):  Useful to change this?
```

is_valid()

```
def is_valid(self):
    if self.hour < 0 or self.minute < 0 or
self.second < 0:
        return False
    if self.minute >= 60 or self.sec >= 60:
        return False
    if self.hour >= 24: # - jdp
        return False
    return True
```

Use is_valid: before

```
def __add__(self, other):  
    if isinstance(other, Time):  
        return self.add_time(other)  
    else:  
        return self.increment(other)
```

Use is_valid: after

```
def __add__(self, other):  
    assert self.is_valid()
```

When the value passed to assert is False

Throws an error

How to use: after

```
def __add__(self, other):
    assert self.is_valid()
    if isinstance(other, Time):
        assert other.is_valid()
        result = self.add_time(other)
        assert result.is_valid()
    return result

else:
    ...

```

Scaffolding

```
def __add__(self, other):
    assert self._parent is None
    if isinstance(other, Scaffolding):
        assert other._parent is None
        result = Scaffolding()
        assert result._parent is None
        result._parent = self._parent
        return result
    else:
```



Scaffolding

Useful when working: removed when done

Clipping hours to < 24

The rest change times. *Or do they?*

```
def __init__(self, hour=0, min=0, sec=0):  
def __add__(self, other):  
def __radd__(self, other):  
def add_time(self, other):  
def increment(self, seconds):  
def int_to_time(seconds):
```

Clipping hours to < 24

`__init__()` is used by the Time constructor, and must be changed

```
def __init__(self, hour=0, min=0, sec=0):  
    self.hour      = hour  
    self.minute   = minute  
    self.second   = second
```

Clipping hours to < 24

`__init__()` is used by the Time constructor, and must be changed

```
def __init__(self, hour=0, min=0, sec=0):  
    self.hour      = (hour % 24)      # jdp  
    self.minute   = minute  
    self.second   = second
```

Clipping hours to < 24

`__init__()` is used by the Time constructor, and must be changed

```
def __init__(self, hour=0, min=0, sec=0):  
    self.hour      = (hour % 24)      # jdp  
    self.minute   = minute  
    self.second   = second
```

Does this fix the problem of 75 minutes?

Clipping hours to < 24

`__init__()` is used by the Time constructor, and must be changed

```
def __init__(self, hour=0, min=0, sec=0):  
    self.hour = (hour % 24) # jdp  
    self.minute = (minute % 60)  
    self.second = (second % 60)
```

We could also decide to carry extras over

See `divmod()` in `int_to_time()`

divmod()

```
def int_to_time(seconds):
    """Makes a new Time object.
    seconds: int seconds since midnight.
    """
    minutes, second = divmod(seconds, 60)
    hour, minute = divmod(minutes, 60)
    return Time(hour, minute, second)
```

Clipping hours to < 24

```
def __init__(self, hour=0, min=0, sec=0):
```

The remaining methods change times. *Or do they?*

Take each in turn

```
def __add__(self, other):  
def __radd__(self, other):  
def add_time(self, other):  
def increment(self, seconds):  
def int_to_time(seconds):
```

Clipping hours to < 24

```
def __add__(self, other):  
    if isinstance(other, Time):  
        return self.add_time(other)  
    else:  
        return self.increment(other)
```

```
def __radd__(self, other):  
def add_time(self, other):  
def increment(self, seconds):  
def int_to_time(seconds):
```

Clipping hours to < 24

```
def __radd__(self, other):
    """Adds two Time objects or a Time
object and a number."""
    return self.__add__(other)
```

```
def add_time(self, other):
def increment(self, seconds):
def int_to_time(seconds):
```

Clipping hours to < 24

```
def add_time(self, other):
    """Adds two time objects."""
    seconds = self.time_to_int() +
              other.time_to_int()
    return int_to_time(seconds)
```

```
def increment(self, seconds):
def int_to_time(seconds):
```

Clipping hours to < 24

```
def increment(self, seconds):  
    """Returns a new Time..."""  
    seconds += self.time_to_int()  
    return int_to_time(seconds)
```

```
def int_to_time(seconds):
```

Clipping hours to < 24

```
def int_to_time(seconds):
    """Makes a new Time object.
    seconds: int seconds since midnight.
    """
    minutes, second = divmod(seconds, 60)
    hour, minute = divmod(minutes, 60)
    return Time(hour, minute, second)
```

Changes

If Increment had updated an existing time, we would have had to fix something

But none of these routines change the time:
they create a new Time Object

We have fixed the initialization of new Time

Initialization

```
def __init__(self, hour=0, min=0, sec=0):  
    self.hour    = (hour % 24)  
    self.minute = (min % 60)  
    self.second = (sec % 60)
```

Testing

```
# Testing
print(Time(0, 0, 0))
print(Time(0, 1, 2))
print(Time(11, 30, 59))
print(Time(12, 0, 3))
print(Time(23, 2, 2))
print()
print(Time(25, 45, 00))
```

Test increment

```
start = Time(9, 45, 00)
start.print_time()
```

```
# Test Increment
# 24 x 60 x 60 = 86,400
end = start.increment(86500)
assert end.is_valid()
print(end)
```

Tonight's outline

Review Homework

Garbage Collection

Object Oriented Programming

Viewing Method invocation with Debugger

Magic Methods

Inheritance

Dynamic Memory

Dynamic Memory - Challenge in programming in C or C++

Programmer must allocate and free memory

Too few frees lead to Memory Leaks

Too many frees lead to miss access

```
void f(int n) {  
    int* array = calloc(n, sizeof(int));  
    do_some_work(array);  
    free(array);  
}
```

Dynamic Memory

It is easy to loose track of memory

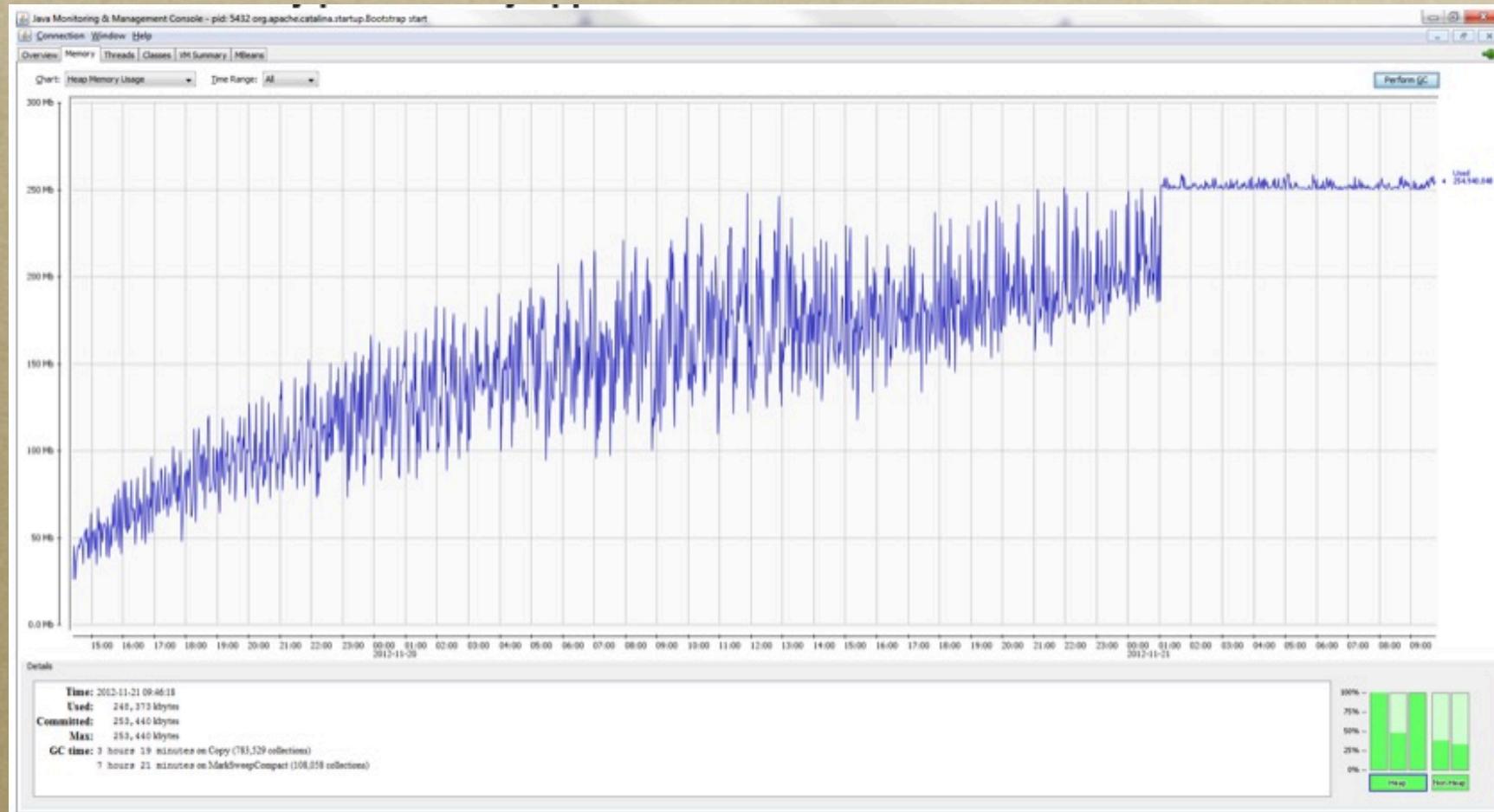
What if there was an exception in `do_some_work`?

Those lost blocks are Garbage

```
void f(int n) {  
    int* array = calloc(n, sizeof(int));  
    do_some_work(array);  
    free(array);  
}
```

Dynamic Memory

<https://stackoverflow.com/questions/13483796/memory-leak-in-a-java-web-application>



Python 3.6

```
1 a = ['c', 'a', 't']
→ 2 b = a
3
4 a = ['d', 'o', 'g']
```

Python Tutor

Frames

Objects

Global frame

a

list

0	"c"	1	"a"	2	"t"
---	-----	---	-----	---	-----

Python 3.6

```
1 a = ['c', 'a', 't']
→ 2 b = a
3
→ 4 a = ['d', 'o', 'g']
```

Frames

Objects

Global frame

a

b

list

0	"c"	1	"a"	2	"t"
---	-----	---	-----	---	-----

Python 3.6

```
1 a = ['c', 'a', 't']
2 b = a
3
→ 4 a = ['d', 'o', 'g']
```

Edit this code

Frames

Objects

Global frame

a

b

list

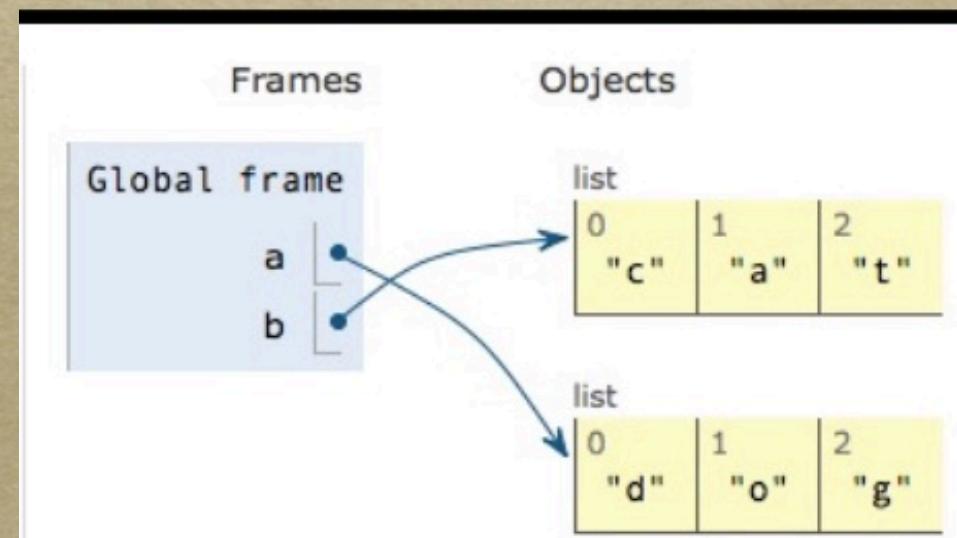
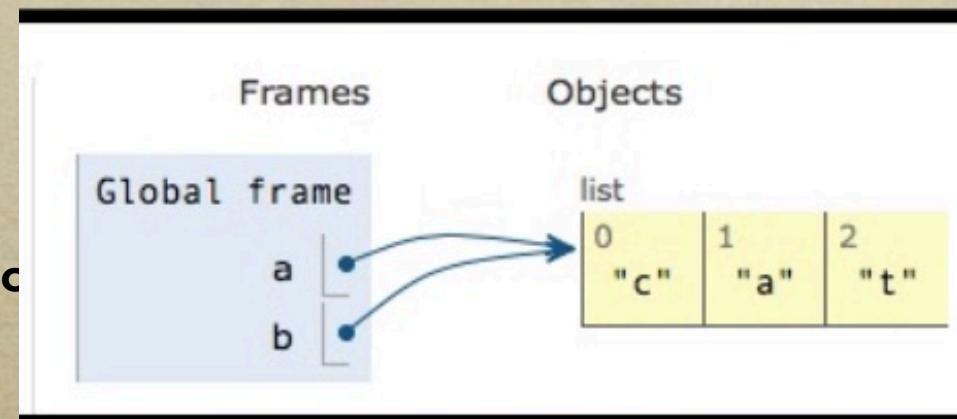
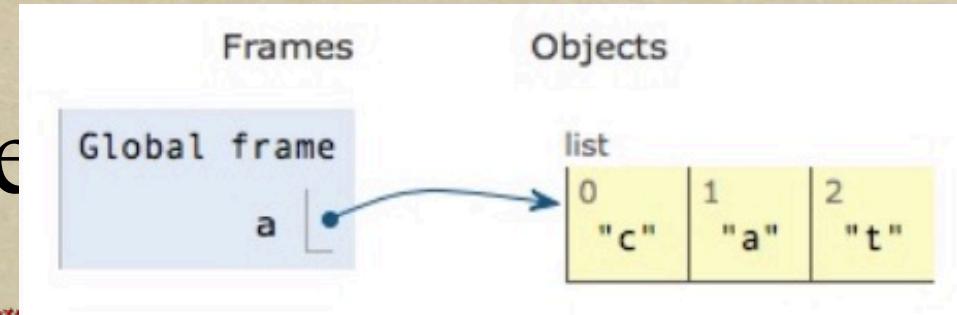
0	"c"	1	"a"	2	"t"
---	-----	---	-----	---	-----

list

0	"d"	1	"o"	2	"g"
---	-----	---	-----	---	-----

Reference

```
>>> import sys  
  
>>> a = list('cat')  
  
>>> print(sys.getrefcount(a))  
2 # There is an extra count  
  
>>> b = a  
  
>>> print(sys.getrefcount(a))  
3  
  
>>> a = list('dog')  
  
>>> print(sys.getrefcount(b))  
2
```



Garbage Collection

SORRY, YOU CAN'T EMPTY THE GARBAGE YET. A PAPER TOWEL IN HERE IS CURRENTLY IN USE BY SOME OBJECT IN YOUR HOUSE.



Garbage Collection

'Which one?' 'I dunno, it's your house. Just check each object.' 'Check it for *what*?' 'Whether it looks like it might have touched a paper towel at some point and then forgotten to let go.' '...' 'You can also Google to learn how to check which things are using which resources.' 'You know, I'll just leave the towel there and try again tomorrow.'



Real Time Systems

Real Time Systems need to respond in a fixed time

Your power steering must respond in real time

Garbage Collection was time consuming

Real Time Systems avoided languages with GC

Modern GC is faster, but more memory also helps

Tonight's outline

Review Homework

Garbage Collection

Object Oriented Programming

Viewing Method invocation with Debugger

Magic Methods

Inheritance

OO Goals

Encapsulation: Split Interface from Implementation

You should be able to change how we implement a Dictionary

If we preserve Interface, old programs still work

Protect the Implementation from poorly-written programs

Inherit functionality from superclass

Tonight's outline

Review Homework

Garbage Collection

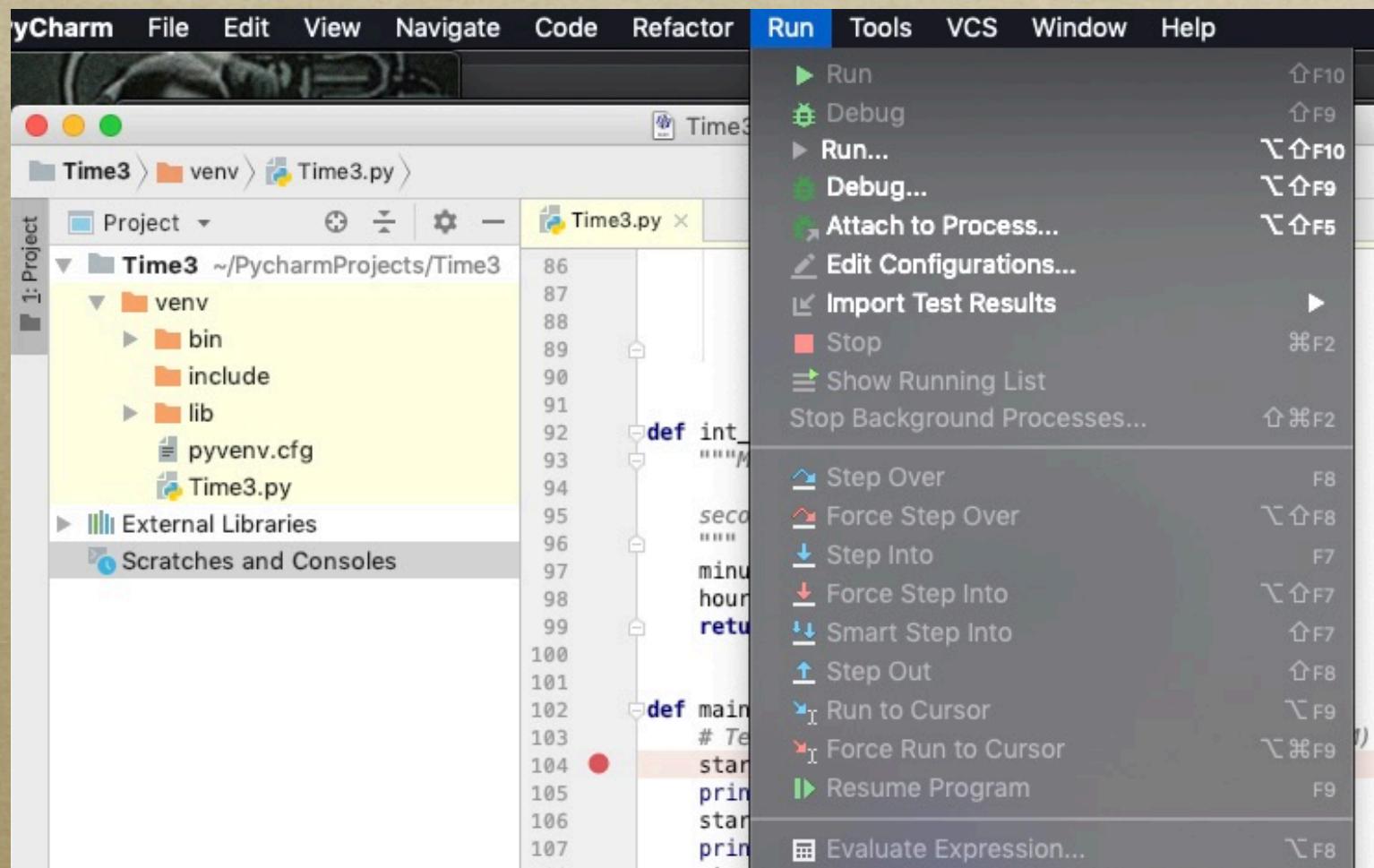
Object Oriented Programming

Viewing Method invocation with Debugger

Magic Methods

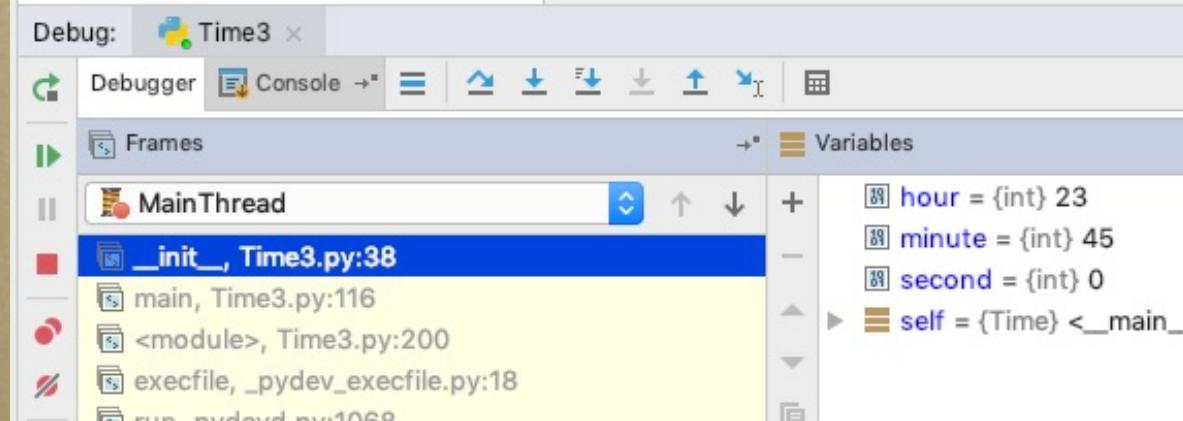
Inheritance

Debugger



__init__

```
36
37
38 def __init__(self, hour=0, minute=0, second=0): self: <__main__.Time object at 0x
39     self.hour = (hour % 24)
40     self.minute = minute
41     self.second = second
```



start = Time(23, 45)

Execution

Stack

The screenshot shows a Python debugger interface with the following details:

- Code Editor:** Displays a script named `Time3.py` with the following content:

```
112
113
114
115      # -Test Construction - jdp
116      start = Time(1000, 45, 00)
117      print(start)
118
119      start = Time(23, 45, 00)
120      print(start)
121      start = Time(24, 00, 00)
122      print(start)
123      start = Time(25, 45, 00)
124      print(start)

# Test addition - jdp
```
- Debug Bar:** Shows the current state as `Step Into (F7)`.
- Frames:** A list of frames showing the call stack:
 - MainThread
 - main, Time3.py:115 (selected)
 - <module>, Time3.py:200
 - execfile, _pydev_execfile.py:18
 - run, pydevd.py:1068
 - main, pydevd.py:1658
- Variables:** Shows the variable `start` with the value `{Time} 4:45 P`.

Who called __str__?

```
17
18     class Time(object):
19         """Represents the time of day.
20
21         attributes: hour, minute, second
22         """
23
24     @↑     def __str__(self): self: 4:45 PM
25         # Morning or night - jdp
26         # AM or PM?
27         if (self.hour >= 12):
28             period = 'PM'
29         else:
30             period = 'AM'
31         # Civilian time - jdp
32         adjusted_hour = self.hour % 12
33         if (adjusted_hour == 0):
34             adjusted_hour = 12
35         return '%2d:%.2d %s' % (adjusted_hour, self.minute, period)
36
```

Who called __str__?

The diagram illustrates a call stack between two code snippets. A red arrow points from the `main()` function in the bottom snippet up to the `__str__()` method in the top snippet, indicating the flow of the call.

Caller (Bottom Snippet):

```
102     def main():
103         # Test printing. Key difficulty is midnight
104         start = Time(0, 45, 00) start: 12:45 AM
105         print(start)
106         start = Time(11, 45, 00)
107         print(start)
108         start = Time(12, 00, 00)
```

Callee (Top Snippet):

```
17
18     class Time(object):
19         """Represents the time of day.
20
21             attributes: hour, minute, second
22             """
23
24     @t
25     def __str__(self): self: 4:45 PM
26         # Morning or night - jdp
27         # AM or PM?
28         if (self.hour >= 12):
29             period = 'PM'
30         else:
31             period = 'AM'
32         # Civilian time - jdp
33         adjusted_hour = self.hour % 12
34         if (adjusted_hour == 0):
35             adjusted_hour = 12
36
37         return '%2d:%.2d %s' % (adjusted_hour, self.minute, period)
38
```

Int2Time

The screenshot shows the PyCharm debugger interface during the execution of the `int_to_time` function. The code editor on the right displays the function definition:

```
91 def int_to_time(seconds): seconds: 36437
92     """Makes a new Time object.
93
94     seconds: int seconds since midnight.
95
96     """
97     minutes, second = divmod(seconds, 60)
98     hour, minute = divmod(minutes, 60)
99     return Time(hour, minute, second)
100
101
```

The line `minutes, second = divmod(seconds, 60)` is highlighted in yellow, indicating it is the current line of execution. A red dot marks the cursor position on this line. The code editor has a light beige background with syntax highlighting.

The debugger window on the left shows the call stack. The current frame is `int_to_time, Time3.py:97`, which is highlighted in blue. Other frames listed include `increment, Time3.py:78`, `main, Time3.py:127`, `<module>, Time3.py:200`, `execfile, _pydev_execfile.py:18`, `run, pydevd.py:1068`, `main, pydevd.py:1658`, and `<module>, pydevd.py:1664`.

Testing

```
start = Time(9, 45)
duration = Time(23, 35)
print(start + duration)
```

Testing

```
def __add__(self, other):
    """Adds two Time objects or a
Time object and a number."""
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)
```

Testing

```
def __add__(self, other):
    """Adds two Time objects or a Time object and a number.
    Time object and a number."""
    if isinstance(other, Time):
```

```
54
55     def __add__(self, other): self: 9:45 AM other: 1:35 AM
56     """Adds two Time objects or a Time object and a number.
57     other: Time object or number of seconds
58     """
59
60     if isinstance(other, Time):
61         return self.add_time(other)
62     else:
63         return self.increment(other)
64
```

```
        return self.add_time(other)
    else:
        return self.increment(other)
```

Testing

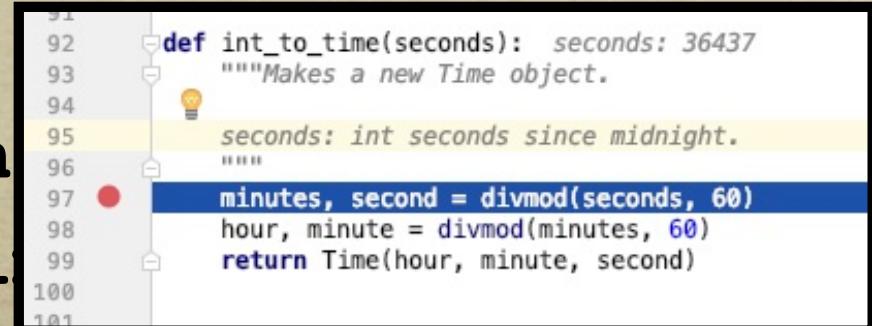
```
def add_time(self, other):
    """Adds two time objects."""
    assert self.is_valid() and
           other.is_valid()
    seconds = self.time_to_int() +
              other.time_to_int()
    return int_to_time(seconds)
```

int_to_time

```
def int_to_time(seconds):
    """Makes a new Time object.
    seconds: int seconds since midnight.
    """
    minutes, second = divmod(seconds, 60)
    hour, minute = divmod(minutes, 60)
    return Time(hour, minute, second)
```

int_to_time

```
def int_to_time(seconds):
    """Makes a new Time object.
    seconds: int seconds since midnight.
    """
    minutes, second = divmod(seconds, 60)
    hour, minute = divmod(minutes, 60)
    return Time(hour, minute, second)
```



Tonight's outline

Review Homework

Garbage Collection

Object Oriented Programming

Viewing Method invocation with Debugger

Magic Methods

Inheritance

Magic Methods

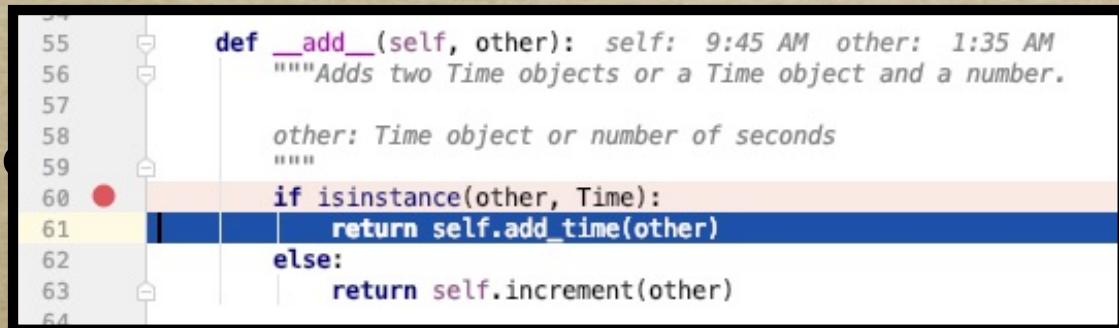
```
def __add__(self, other):
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)
```

```
start = Time(9, 45)
duration = Time(1, 35)
print(start + duration)
print(start + 1337)
```

Magic Methods

```
def __add__(self, other):  
    if isinstance(other, Time):  
        return self.add_time(other)
```

```
else:  
    return s
```



```
start = Time(9, 45)
```

```
duration = Time(1, 35)
```

```
print(start + duration)
```

```
print(start + 1337)
```

Magic Methods

```
def __add__(self, other):
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)
```

```
def __add__(self, other):
    """Adds two Time objects or a Time object and a number.
    other: Time object or number of seconds
    """
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)
```

```
start = Time(9, 45)
duration = Time(1, 35)
print(start + duration)
print(start + 1337)
```

Magic Methods

```
def __add__(self, other):
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)

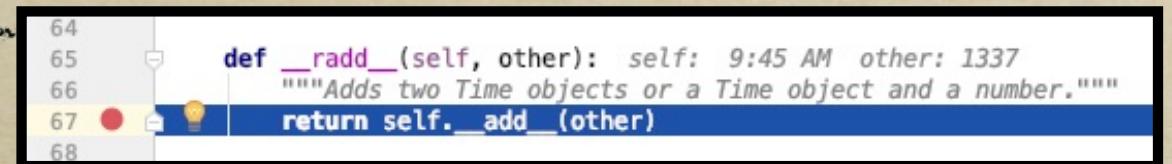
start = Time(9, 45)
...
print(start + 1337)
print(1337 + start) # How do we resolve?
```

Magic Methods

```
def __add__(self, other):
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)

start = Time(9, 45)
...
print(start + 1337)
print(1337 + start) # int.__add__(Time)
```

Magic Methods



```
def __radd__(self, other):
    return self.__add__(other)
```

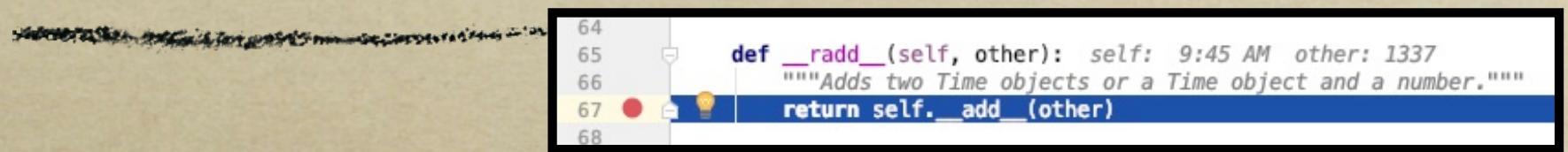
```
start = Time(9, 45)
```

```
...
```

```
print(start + 1337)
```

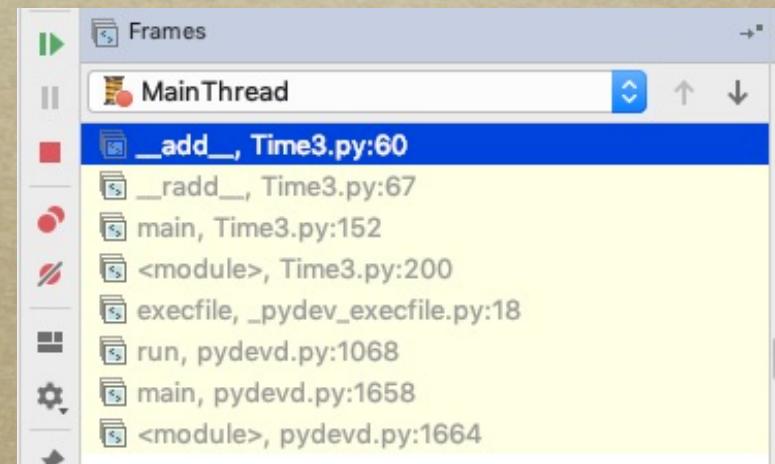
```
print(1337 + start)
```

Magic Methods



```
def __radd__(self, other):
    return self.__add__(other)
```

```
start = Time(9, 45)
...
print(start + 1337)
print(1337 + start)
```



Sum

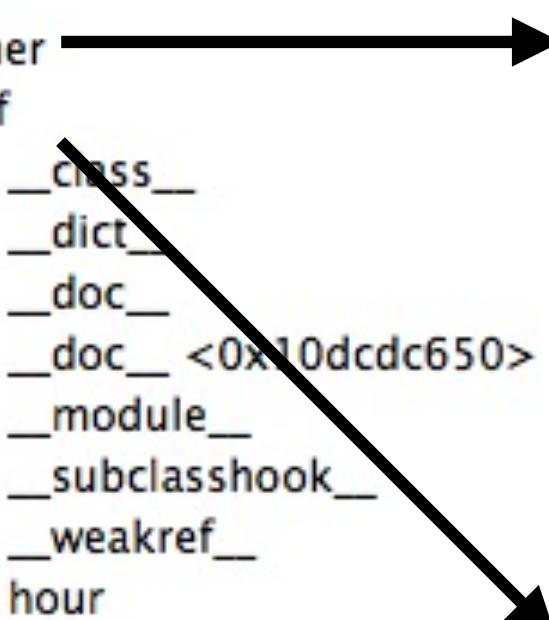
```
print('Example of polymorphism')  
t1 = Time(7, 43)  
t2 = Time(7, 41)  
t3 = Time(7, 37)  
total = sum([t1, t2, t3])  
print(total)
```

```
- def __radd__(self, other):
    """Adds two Time objects or a Time object and a number."""
    return self.__add__(other)
```

Search in Files Search Stack Data Exceptions Breakpoints Testing

Time2.py (pid 2947) (paused)

__radd__(): Time2.py, line 49

Variable	Value
locals	<dict 0x10e5ac088; len=2>
other	0
self	
__class__	<__main__.Time 0x10e5ee828; len=1>
__dict__	<getset_descriptor 0x10dcdb2d0; len=0>
__doc__	<getset_descriptor 0x10e5ed4c8; len=0>
__doc__	"Represents the time of day.\n \n attributes: hour, minute"
__module__	"The most base type"
__subklasshook__	"__main__"
__weakref__	<classmethod_descriptor 0x10dcdb168; len=0>
hour	<getset_descriptor 0x10e5ed510; len=0>
minute	7
second	43
	0

```
- def __add__(self, other):
    """Adds two Time objects or a Time object and a number.

    other: Time object or number of seconds
    """
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)
```

Search in Files Search Stack Data Exceptions Breakpoints Testing
Time2.py (pid 2947) (paused) __add__: Time2.py, line 45

Variable	Value
locals	<dict 0x10e5ecb08; len=2>
other	0
self	<__main__.Time 0x10e5ee828; len=1> <getset_descriptor 0x10dcdb2d0; len=0> <getset_descriptor 0x10e5ed4c8; len=0> "Represents the time of day.\n \n attributes: hour, minute" "The most base type" " __main__ " <classmethod_descriptor 0x10dcdb168; len=0> <getset_descriptor 0x10e5ed510; len=0>
hour	7
minute	43
second	0

```
def __add__(self, other):
    """Adds two Time objects or a Time object and a number.

    other: Time object or number of seconds
    """
    if isinstance(other, Time):
        return self.add_time(other)
    else:
        return self.increment(other)
```

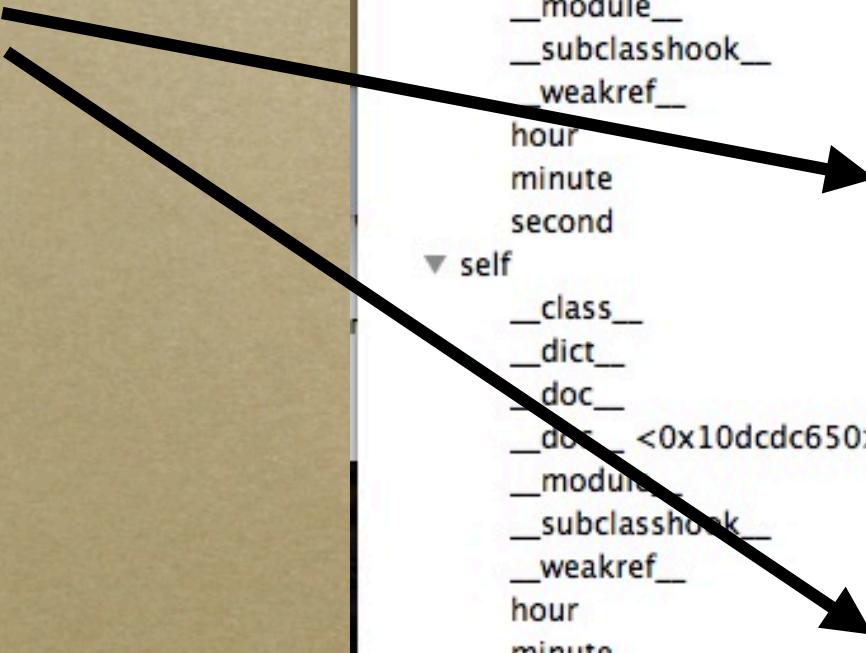
Search in Files Search Stack Data Exceptions Breakpoints Testing

Time2.py (pid 2947) (paused)

__add__: Time2.py, line 43

Variable	Value
locals	<dict 0x10e608648; len=2>
other	<__main__.Time 0x10e5ee908; len=1>
__class__	<getset_descriptor 0x10dcdb2d0; len=0>
__dict__	<getset_descriptor 0x10e5ed4c8; len=0>
__doc__	"Represents the time of day.\n \n attributes: hour, minute, second"
__doc__ <0x10dc dc650>	"The most base type"
__module__	"__main__"
__subclasshook__	<classmethod_descriptor 0x10dc db168; len=0>
__weakref__	<getset_descriptor 0x10e5ed510; len=0>
hour	7
minute	41
second	0
self	<__main__.Time 0x10e5eeb38; len=1>
__class__	<getset_descriptor 0x10dcdb2d0; len=0>
__dict__	<getset_descriptor 0x10e5ed4c8; len=0>
__doc__	"Represents the time of day.\n \n attributes: hour, minute, second"
__doc__ <0x10dc dc650>	"The most base type"
__module__	"__main__"
__subclasshook__	<classmethod_descriptor 0x10dc db168; len=0>
__weakref__	<getset_descriptor 0x10e5ed510; len=0>
hour	7
minute	43
second	0

total = sum



```
- def __add__(self, other):
-     """Adds two Time objects or a Time object and a number.
-
-     other: Time object or number of seconds
-
-     """
-     if isinstance(other, Time):
-         return self.add_time(other)
-     else:
-         return self.increment(other)
```

Search in Files Search Stack Data Exceptions Breakpoints Testing
Time2.py (pid 2947) (paused) _add_(): Time2.py, line 43

Variable	Value
locals	<dict 0x10e605888; len=2>
other	<__main__.Time 0x10e5ee860; len=1>
__class__	<getset_descriptor 0x10dcdb2d0; len=0>
__dict__	<getset_descriptor 0x10e5ed4c8; len=0>
__doc__	"Represents the time of day.\n \n attributes: hour, minute, second"
__doc__ <0x10dc dc650>	"The most base type"
__module__	"__main__"
__subclasshook__	<classmethod_descriptor 0x10dc db168; len=0>
__weakref__	<getset_descriptor 0x10e5ed510; len=0>
hour	7
minute	37
second	0
self	<__main__.Time 0x10e5ee978; len=1>
__class__	<getset_descriptor 0x10dcdb2d0; len=0>
__dict__	<getset_descriptor 0x10e5ed4c8; len=0>
__doc__	"Represents the time of day.\n \n attributes: hour, minute, second"
__doc__ <0x10dc dc650>	"The most base type"
__module__	"__main__"
__subclasshook__	<classmethod_descriptor 0x10dc db168; len=0>
__weakref__	<getset_descriptor 0x10e5ed510; len=0>
hour	15
minute	24
second	0

total = sum

