

Advanced Debugging with pdb and Friends



Douglas Starnes

Author / Entrepreneur / Speaker

linktr.ee/douglasstarnes



Overview



Commands to navigate the source

- Step in
- Step out
- Moving through the stack trace

Read the stack trace

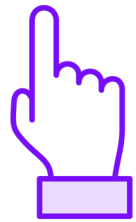
Additional tools to help debug Python code



Commands for navigating the source



step (alias **s**) – similar to “step into” in other debuggers



up (alias **u**) – move up in the stack trace (similar to “step out”)



down (alias **d**) – move down in the stack trace

1

Both **up** and **down** default to 1 level



The step (or s) command

```
def get_investment_info(investment):
    name = investment["name"]
    quantity = investment["quantity"]
    import pdb; pdb.set_trace()
    current_price = get_current_price(name)
    print(f"The current price of {name} is {current_price}.")

def get_current_price(coin):
    print("Getting current price ...")
    if coin == "bitcoin":
        return 10000
    # ...
```



The step (or s) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    → current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```



The step (or s) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")
```

```
➡ def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) step



The step (or s) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")
```

```
def get_current_price(coin):  
    ➡ print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) next



The up (or u) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    → if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) next



The up (or u) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
→ current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) up



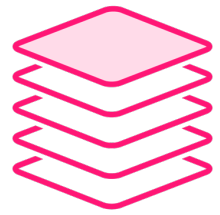
The up (or u) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    → print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

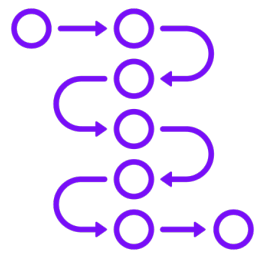
(Pdb) next



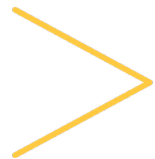
Commands for navigating the stack trace



bt / where (alias w) – print the current stack trace



The frames are displayed vertically from oldest to newest



Angle bracket references the current frame



Thin arrow references the current line in each frame



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    → current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")
```

```
➡ def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) step



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")
```

```
def get_current_price(coin):  
    → print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) next



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    → if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) next



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    → if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) where



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")
```

```
def get_current_price(coin):  
    print("Getting current price ...")
```

```
➡ if coin == "bitcoin":  
    return 10000
```

```
# ...
```

(Pdb) where

```
/root/src/ps/pdbdemo/app.py(34)<module>()  
-> get_investment_info(bitcoin)  
   /root/src/ps/pdbdemo/app.py(28)get_investment_info()  
-> current_price = get_current_price(name)  
> /root/src/ps/pdbdemo/app.py(15)get_current_price()  
-> if coin == "bitcoin":
```



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    → current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")  
  
def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) up



The where (or w) command

```
def get_investment_info(investment):  
    name = investment["name"]  
    quantity = investment["quantity"]  
    import pdb; pdb.set_trace()  
    → current_price = get_current_price(name)  
    print(f"The current price of {name} is {current_price}.")
```

```
def get_current_price(coin):  
    print("Getting current price ...")  
    if coin == "bitcoin":  
        return 10000  
    # ...
```

(Pdb) where

```
/root/src/ps/pdbdemo/app.py(34)<module>()  
-> get_investment_info(bitcoin)  
> /root/src/ps/pdbdemo/app.py(28)get_investment_info()  
-> current_price = get_current_price(name)  
   /root/src/ps/pdbdemo/app.py(15)get_current_price()  
-> if coin == "bitcoin":
```



Commands for displaying the source

list (1)

**Display lines around the current
line or a range of lines**

longlist (11)

**Display the code for the current
function or frame**



Tools to Help You Debug Your Code

PyLint

ipdb

**black, isort, mypy
and more**



Summary



Everything stack trace

- Step into functions
- Step out of functions
- Move up and down in the stack trace
- Display the stack trace
- Read the stack trace
- Display the debugged code in extra context

Tools to help debug

- PyLint
- ipdb

