Key Terms

- **import**: Brings a module into the program's running memory so its contents can be accessed
- **variable**: A name that points to a piece of data
- **assignment**: Assigns a value to a variable
- **expression**: A piece of code that evaluates to a value
- **exception**: An error that stops execution of a program

```
1   import pandas as pd
2
3   # Create a variable pointing to a pandas Series
4   s = pd.Series([1, 2, 3])
5
6   # Assign a new variable pointing to the sum
7   total = s.sum()
8
9   # Update the variable with an expression
10  total += 10
11
12  # Raise a pandas AttributeError exception
13  try:
14      s.no_such_attribute()
15  except AttributeError:
16      print("Oops! That attribute doesn't exist")
```

```
Oops! That attribute doesn't exist
```

Expression Statements in Python

Python Statements

Simple Statements

- Expression Statements
- Assert Statements
- Assignment Statements
- Pass Statements
- Del Statements
- Return Statements
- Yield Statements
- Raise Statements
- Break Statements
- Continue Statements
- Import Statements
- Future Statements
- Global Statements
- Nonlocal Statements

Compound Statements

- If Statements
- While Statements
- For Statements
- Try Statements
- With Statements
- Match Statements
- Function Definitions
- Class Definitions
- Coroutines

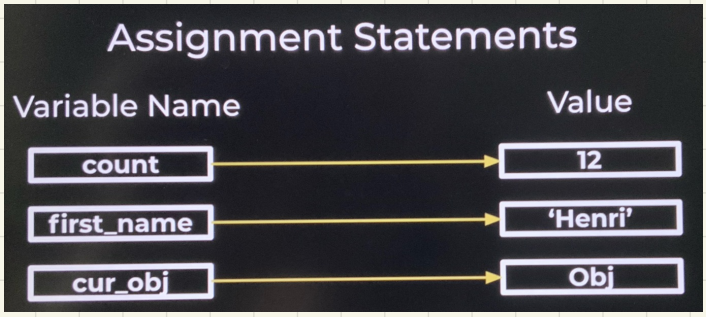
Expression Statements

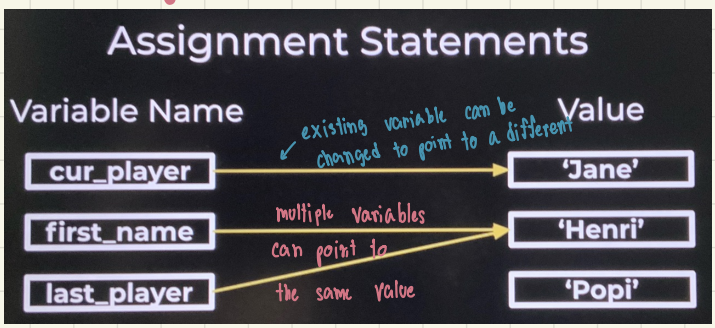
- **Expression** - Code that evaluates to a value
- **Expression Statement** - Statement only containing expressions

```
(env)  $ python  → to access Python shell
Python 3.9.7 (default, Sep 12 2021, 10:26:48)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()  to exit Python shell
(env)  $ pip install ipython  → another Python shell that is popular
```

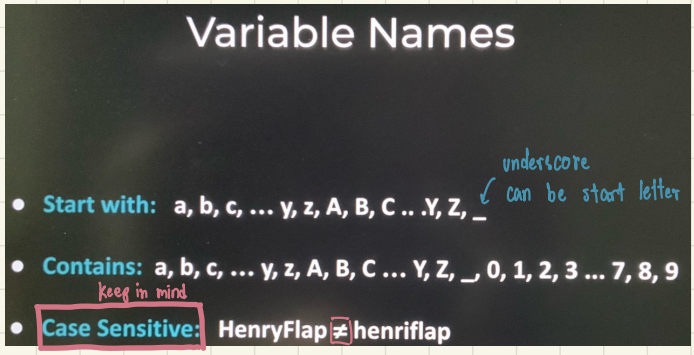
```
(env)  $ ipython  → to access ipython shell
/Users/kennedyrobertbehrman/.pyenv/versions/3.9.7/lib/python3.9/site-packages/IPython/core/interactiveshell.py:934
 UserWarning: Attempting to work in a virtualenv. If you encounter problems, please install IPython inside the vir
ualenv.
  warn("Attempting to work in a virtualenv. If you encounter problems, please "
Python 3.9.7 (default, Sep 12 2021, 10:26:48)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.28.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: 1 + 1
Out[1]: 2

In [2]:  → you can type any math operation here

                → it can also be echo a string (words) : 'Hello'
```

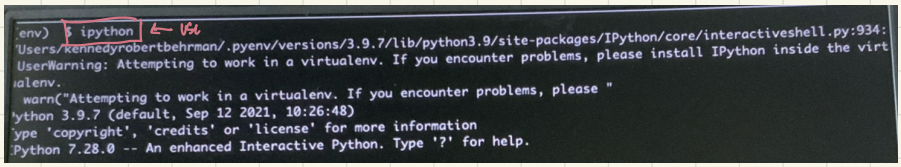
Assignment Statements in Python

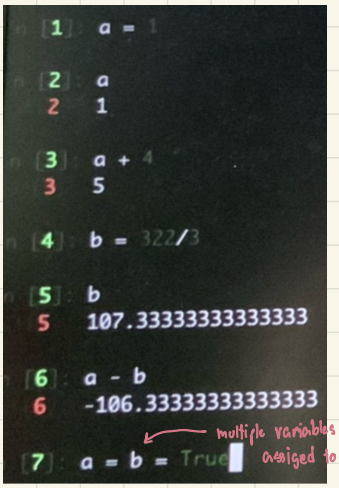


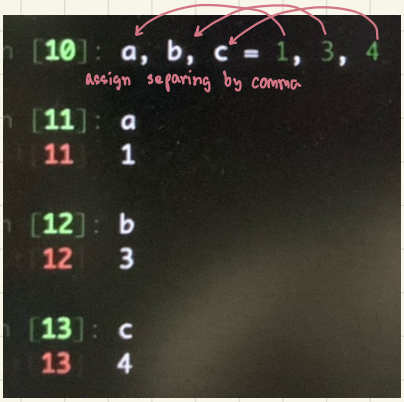


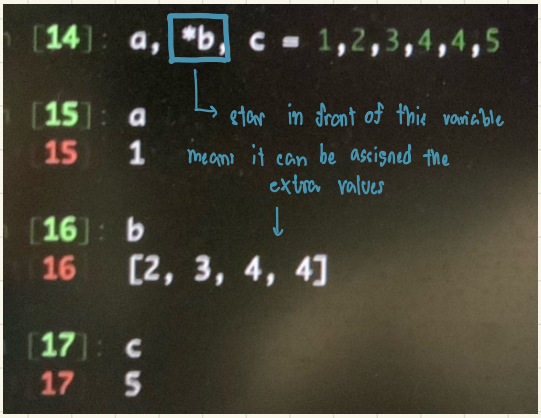
- **Important:** Variable points to value (data)
  it points to data, it's not data itself

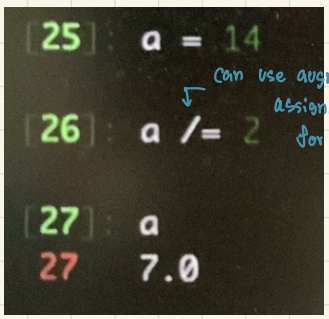




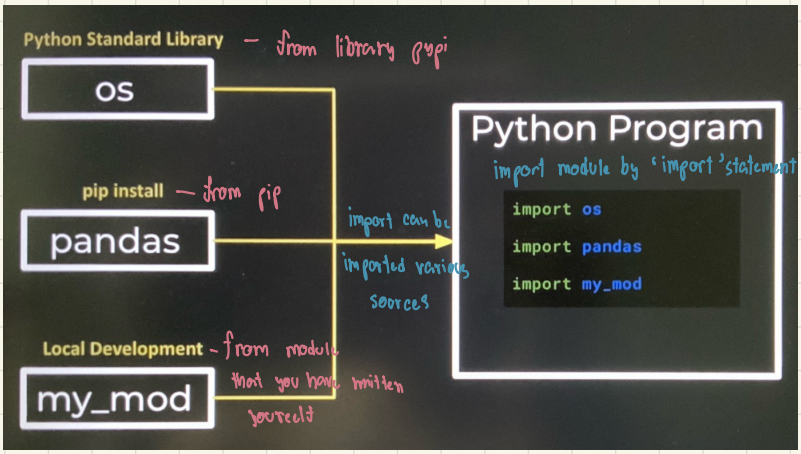








Import Statements in Python




```
1  import os → imported 'os' package

2  os → enoch address ,where this module at
2  <module 'os' from '/Users/kennedyrobertbehrman/.pyenv/versions/3.9.7/lib/python3.9/os.py'>
       from os function: get current working directory
3  os.getcwd() → similar to pwd from shell
3  '/Users/kennedyrobertbehrman/Google Drive/projects/python.and.pandas.for.data.engineering.Duke.Coursera'

4  os.path
4  <module 'posixpath' from '/Users/kennedyrobertbehrman/.pyenv/versions/3.9.7/lib/python3.9/posixpath.py'>
          ↑ define in sub-module
5  os.path.isdir(' ')
5  True
                    we can only import path
6  from os import path   with out parent module
7  path.isdir(' ')
7  True    ↘ so we can access sub module path without the parent module name
8  from os.path import isdir → we can also import a specific function
9  isdir(' ')
9  True

10 from os import * ← import all module
```

* star syntax is a wild card, stand in for all possible sub-modules and functions.

not recommended
- just import package
- specific module, function

Alias

```
In [11]: import pandas as pd
                          alias

In [12]: pd
    12   <module 'pandas' from '/Users
das/__init__.py'>

In [13]: pd.DataFrame
    13   pandas.core.frame.DataFrame
```

Other Simple Statements in Python

Python Statements

Simple Statements

- Expression Statements
- Assert Statements → deal with "Exceptions"
- Assignment Statements
- Pass Statements
- Del Statements
- Return Statements
- Yield Statements
- Raise Statements
- Break Statements
- Continue Statements
- Import Statements
- Future Statements — Type of import statement
- Global Statements
- Nonlocal Statements

↳ import a feature from a later version of python into the current version

↓

rare situation for preparing to migrate code between python versions

Compound Statements

- If Statements
- While Statements
- For Statements
- Try Statements
- With Statements
- Match Statements
- Function Definitions
- Class Definitions
- Coroutines

Exceptions:

```
1 # Simple program demonstrating ZeroDivisionError
2
3 a = 23/0
4 print(a)
```

```
$ vim raise_example.py
$ python raise_example.py
Traceback (most recent call last):
  File "/Users/kennedyrobertbehrman/Google Drive/projects/python.and.pandas.for.data.engineering.Duke.Coursera/my_first_project/raise_example.py", line 3, in <module>
    a = 23/0
ZeroDivisionError: division by zero
```

in this code

code stopped here, and the exception was raised

The exception message contains what is known as a traceback

in ipython

```
In [1]: 23/0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-2701679944cb> in <module>
----> 1 23/0

ZeroDivisionError: division by zero
```

exception message

used for bring up an exception

```
In [2]: raise NotImplementedError("My message")
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
<ipython-input-2-d7f140a829f8> in <module>
----> 1 raise NotImplementedError("My message")

NotImplementedError: My message
```

we have raised the built in exception

useful when we have defined a functions name but that method or function has not been implemented yet

Exception: • Python has many built in exceptions
Can create your own exceptions

Assert:

• used to confirm that some condition in your programe is True

```
In [3]: assert 1 + 1
In [4]: a = True
In [5]: assert a
In [6]: assert 1 == 2
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-6-ce781a32d075> in <module>
----> 1 assert 1 == 2

AssertionError:

In [7]: assert 1 == 2, "you didn't think that was true?!"
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-7-d160a90516d8> in <module>
----> 1 assert 1 == 2, "you didn't think that was true?!"

AssertionError: you didn't think that was true?!
```

if it is True, Assert will not show up

if it is Fault

you can insert the reason

A search statements are generally used during development or debugging of programs