Python importing modules and arparse

Create a file named **addpackage.py** and type the code below and save it.

```
def addnumbers(a,b):
    return a+b

if __name__ == '__main__':
    print(addnumbers(5,6))
```

Execute the above file by typing **python addpackage.py** in your terminal after navigating to the directory where the file is stored. You should see 11 as the output.

In the same directory where you stored addpackage.py, create a new python file named **importpackage.py** and type the code below and save it.

```
import addpackage
print(addpackage.addnumbers(2,3))
```

Execute this file and notice the output. The answer should be 5.

Notice the use of the import command to import another python package. Because we have defined a function called addnumbers() in the addpackage.py python file, we can always import the python file as a package and use the addnumbers() function. Importing can also be done as below

```
from addpackage import addnumbers
addnumbers(2,3)
```

In the above example, we are directly importing the addnumbers function from addpackage.py .

The ***if __name__ == '__main__':*** statement should be new to you. (Notice the double underscore). When you execute a python file, the python interpreter creates the __name__ variable and assigns the value __main__ to it. Because the __name__ has a double underscore, it is a python special variable.

When the file containing ***if __name__ == '__main__':*** is executed (***python addpackage.py***). Control goes into the if statement because the __name__ variable contains the value __main__.

But when addpackage is imported into another python file, control does not go into the if block of addpackage because __main__ is now set to the file that imports addpackage. In this example when we execute ***python importpackage.py*** the statement ***if __name__ == '__main__':*** in addpackage.py fails and so we only see 5 as the output.

Python argparse module

The argparse module allows users to provide command line inputs during execution.

Create a new python file called **parse.py** and type the below code into it and save the file.

```
#pip install argparse before execution to install the argparse package

import argparse

def add(args):
    return(args.x + args.y)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="Code to add two numbers")
    parser.add_argument('x', type=int, help="First number to add")
    parser.add_argument('y', type=int, help="Second number to add")
    args = parser.parse_args()
    sum = add(args)
    print(sum)

# To execute: python parse.py 10 20
```

The above example accepts two user inputs. These are mandatory positional arguments. Execute the above code by executing ***python parse.py 10 20*** If the command line arguments are not provided we get an error. Notice the different error messages when we provide less than 2 arguments, provide no arguments and provide more than 2 arguments.

Argparse optional arguments.

To convert the mandatory positional arguments to optional arguments, make the following changes

parser.add_argument('--x', type=int, help="First number to add")

parser.add_argument('--y', type=int, help="Second number to add")

We changed x to --x and --y to make them optional arguments. We can now execute the modified code like: ***python parse.py --x=10 --y=20***

Because these are optional arguments we can omit them. But in doing so we get an error because the add function expects arguments to add the numbers. We can get around this error by adding defaults to the add_argument function as below.

parser.add_argument('--x', type=int, default=8, help="First number to add")

parser.add_argument('--y', type=int, default=5, help="Second number to add")

Now when we do not provide arguments, 5 and 8 are used as defaults and the output is 13.

List arguments using argparse

So far we have seen python arguments being provided as positional arguments or as optional arguments. We can also accept a list of numbers as arguments as store it in a python list. To do so

Replace the code with the one below

```
import argparse



if __name__ == '__main__':

    parser = argparse.ArgumentParser(description="Code to add two numbers")
    parser.add_argument('--lst', type=int, nargs='+', help="Second number to add")
    args = parser.parse_args()
    print(args.lst)
```

Execute the code with: ***python parse.py --lst 10 12 13***

Notice the args.lst is now a python list.