Python Function Arguments

you can define a function that takes variable number of arguments

define such functions using default, keyword and arbitrary arguments

Arguments

In the user-defined function : defining a function and calling it. Otherwise, the function call will result in an error. Here is an example.

```
def greet(name, msg):
    """This function greets to
    the person with the provided message"""
    print("Hello", name + ', ' + msg)

greet("Monica", "Good morning!")
```

Output

Hello Monica, Good morning!

Variable Function Arguments

Up until now, functions had a fixed number of arguments. In Python, there are other ways to define a function that can take variable number of arguments. Three different forms of this type are described below.

Python Default Arguments

Function arguments can have default values in Python. We can provide a default value to an argument by using the assignment operator (=).

```
def greet(name, msg="Good morning!"):
    """
    This function greets to
    the person with the
    provided message.
    If the message is not provided,
    it defaults to "Good
    morning!"
    """
    print("Hello", name + ', ' + msg)
greet("Kate")
greet("Bruce", "How do you do?")
```

Output

```
Hello Kate, Good morning!
Hello Bruce, How do you do?
```

Python Anonymous/Lambda Function

What are lambda functions in Python?

an anonymous function is a function that is defined without a name.

While normal functions are defined using the def keyword in Python, anonymous functions are defined using the lambda keyword.

How to use lambda Functions in Python?

A lambda function in python has the following syntax.

Syntax of Lambda Function in python

lambda arguments: expression

- Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

```
# Program to show the use of lambda functions
double = lambda x: x * 2

print(double(5))
```

Output

10

Use of Lambda Function in python

- We use lambda functions when we require a nameless function for a short period of time.
- generally use it as an argument to a higher-order function (a function that takes in other functions as arguments). Lambda functions are used along with built-in functions like filter(), map()
- The filter() function in Python takes in a function and a list as arguments.

The function is called with all the items in the list and a new list is returned which contains items for which the function evaluates to True. Here is an example use of filter() function to filter out only even numbers from a list.

```
# Program to filter out only the even items from a list

my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

print(new_list)
```

Output

```
[4, 6, 8, 12]
```

The map() function in Python takes in a function and a list. The function is called with all the items in the list and a new list is returned which contains items returned by that function for each item.

Here is an example use of map() function to double all the items in a list.

```
# Program to double each item in a list using map()

my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

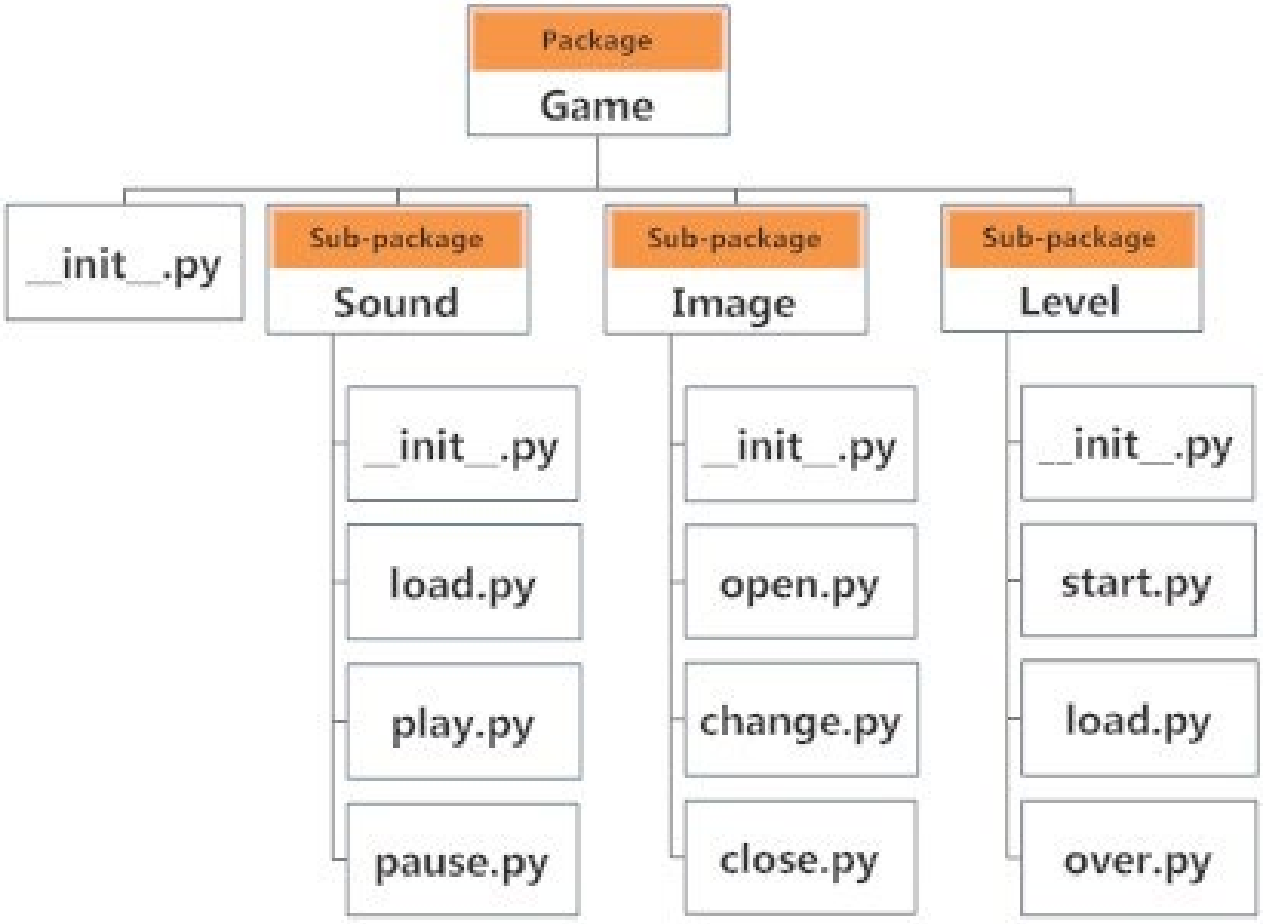
Output

```
[2, 10, 8, 12, 16, 22, 6, 24]
```

What are packages?

- We don't usually store all of our files on our computer in the same location. We use a well-organized hierarchy of directories for easier access.
- Similar files are kept in the same directory, for example, we may keep all the songs in the "**music**" directory. Analogous to this, Python has packages for directories and modules for files.
- As our application program grows larger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project (program) easy to manage and conceptually clear.
- Similarly, as a directory can contain subdirectories and files, a Python package can have sub-packages and modules.
- A directory must contain a file named __init__.py in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.

Suppose we are developing a game. One possible organization of packages and modules could be as shown in the figure below.



Importing module from a package

We can import modules from packages using the dot (.) operator.

For example, if we want to import the start module in the above example, it can be done as follows:

import Game.Level.start

- if this module contains a function named select_difficulty(), we must use the full name to reference it.

- Game.Level.start.select_difficulty(2)
- If this construct seems lengthy, we can import the module without the package prefix as follows:

- from Game.Level import start

- We can now call the function simply as follows:

- start.select_difficulty(2)
- Another way of importing just the required function (or class or variable) from a module within a package would be as follows:

- from Game.Level.start import select_difficulty
- Now we can directly call this function.

- select_difficulty(2)