Scientific Computing with Python Lab

12th Session(April 16th)

Kaheon Kim : https://github.com/kaheonkim/Scientific-computing-with-python-lab-material

Schedules: 3 more sessions left

HW7 : Due April 26(Fri)

- Today(April 16) : Talk about concepts for function and class
- Next week(April 23) : Talk about the HW7 and get a question
- Last session(April 30) : Talk about the final

Things to do today

HW7 : Due April 26(Fri)

- Function inputs : default value, args, functions
- Function output : defining in the big function and returning it
- Brief introduction for OOP(Objected-Oriented Programing) : Class

Function

basics

function is for the situation we have to repeat the same operation with different inputs

```
def func_name(input1, input2, …, inputn):
    operations(input1,input2, …, inputn)
    return output

func_name(input_1, input_2, …, input_n)
```

Function

Default inputs

function is for the situation we have to repeat the same operation with different inputs

def func_name(input1=default1, input2=default2, input3=default3):

operations(input1,input2, …, inputn)

return output

func_name(input1=1, input2=2, input3 =3)

func_name() # given that the defaults are set in all locations for inputs

func_name(input1=2, input2=3)

func_name(input2=3, input1=2) # you can replace the location of inputs

Lambda function

- For the function of really simple form, we can define function with one line

```
f = lambda a,b,c: a+b+c
f(1,2,3)
```

- We can use it by using the rubric : filtering, mapping, sorting

Function input

args

- You can give flexibility on the number of inputs

```
def func_name(*args):
    operation on any upcoming arguments(args)

    return output
```

- *args represent the series of inputs for the func2 and it has no limit on the length of the inputs.

- But you have to keep some idea about the form of arguments

Exercise

Function as an input

- You may have the situation that you have to do the operation with different functions.

1. In this problem, we will implement a function to carry out Euler's method to solve an ordinary differential equation (ODE) for an arbitrary real-valued function $f(x, y)$ as the right hand side over the interval $[0, b]$. We wish to approximate the solution to the ordinary differential equation:

$$y' = f(x, y)$$
$$y(0) = y_0$$

using Euler's method and N subintervals.

In a file named ode.py, write the definition of a function named eulers_method. Your eulers_method function should take four parameters: a function f (assumed to take two parameters, x and y), a float representing y_0, a float representing the endpoint b, and an integer N for the number of subintervals. The function should return a list of tuples of float values (x_i, y_i) for $i = 0, \ldots, N$.

I have provided a script that imports the euler.py module and calls the eulers_method function and then writes output to a file. I have provided sample output for the script on Canvas.

See a later page of the assignment for an explanation of Euler's method.

Function as an input

- You may have the situation that you have to do the operation with different functions.

1 Euler's Method

To approximate the solution of the differential equation

$$y' = f(x, y)$$
$$y(0) = y_0$$

over the interval $[0, b]$, divide the interval $[0, b]$ into N distinct subintervals, each of equal length $h = b/N$. The subintervals will have endpoints $[x_{i-1}, x_i]$, where $x_0 = 0$, $x_N = b$ and $x_i = b/N$. Let y_i denote the approximate solution at the point x_i, i.e. $y_i \approx y(x_i)$. To compute the values y_i, we use an iterative procedure.
Take y_0 to be the initial value $y(0)$. For $i = 0, \ldots, N-1$, define iteratively

$$y_{i+1} = y_i + f(x_i, y_i)h$$

This procedure is just repeated use of the approximation

$$y(x + h) \approx y(x) + y'(x)h = y(x) + f(x, y(x))h$$

using values at the former iteration for x_i and y_i.

Function as an input

- Then, you may take the function as an input for another function!

```
def func_name(f, input1, input2, …, inputn):
    operations(f, input1,input2, …, inputn)

    return output
```

- You can get flexibility as you don’t have to fix the form of given functions.

Get a function with two inputs(of the form f(x,y)) and two values(a,b). Return f(a,b)

Exercise

Function as an output

defining function inside of the function

- You can return a function form as an output after defining in the big function.

```
def func_name(input1, input2, …, inputn):
operations…
def f(x):
    ……
    return …
 return f
```

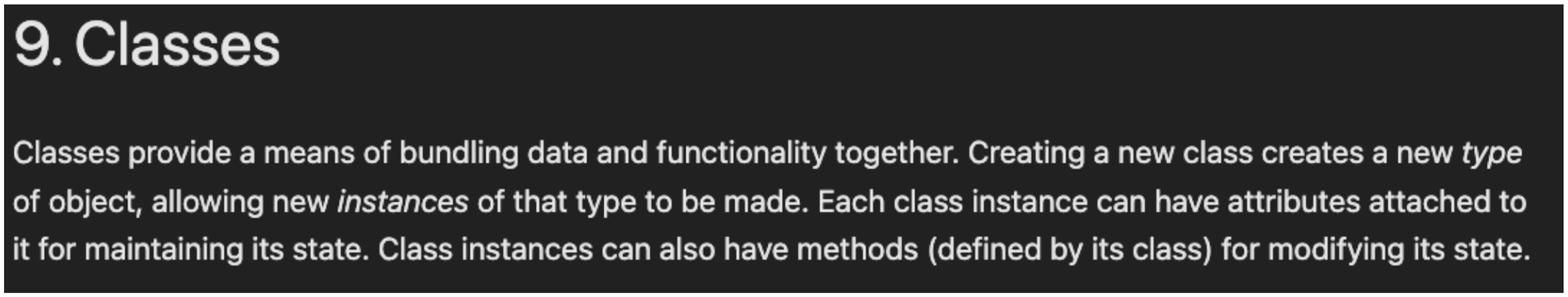
Get a two function input(f and g), and return the function f+g

Exercise

Objected Oriented Programming

Class

- Reference : https://docs.python.org/3/tutorial/classes.html



- Class is the entity of functions(operations) bound in a group. It’s like a package, such as ‘math’.

Objected Oriented Programming

Class

```
class class_name:
    def __init__(self,input1, input2, ...):
        self.x1 = input1
        self.x2 = input2
        ,
        ,
        ,

    def func1(self):
        return operation on variables(self.x1, self.x2,...)
    def func2(self):
        return operation on variables(self.x1, self.x2,...)

        ,
        ,
        ,
class_1 = class_name()
class_1.func1()
class_1.func2()
```

- class_name
- self : you assign variables and functions for the operation into the entity called self
- __init__ : assigning values to self → the inputs are for class

Simple Example

Exercise

4 Arithmetics(Addition, Subtraction, Division, Multiplication of 2 numbers) and l_p distance

Exercise

Objected Oriented Programming

Inheritance

- You can inherit properties(variables and functions) of one class to another for reusing.
- You can refer the name of class you want to inherit in the bracket of the new class

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age
  def __str__(self):
    return f"{self.name}({self.age})"
  def myfunc(self):
    print("Hello my name is " + self.name)
```

```
class Employee(Person):
    def __init__(self, name, age, salary, position):
        self.salary = salary
        self.position = position

        # invoking the __init__ of the parent class
        Person.__init__(self, name, age)

    def details(self):
        print("My name is {}".format(self.name))
        print("age: {}".format(self.age))
        print("position: {}".format(self.position))
```