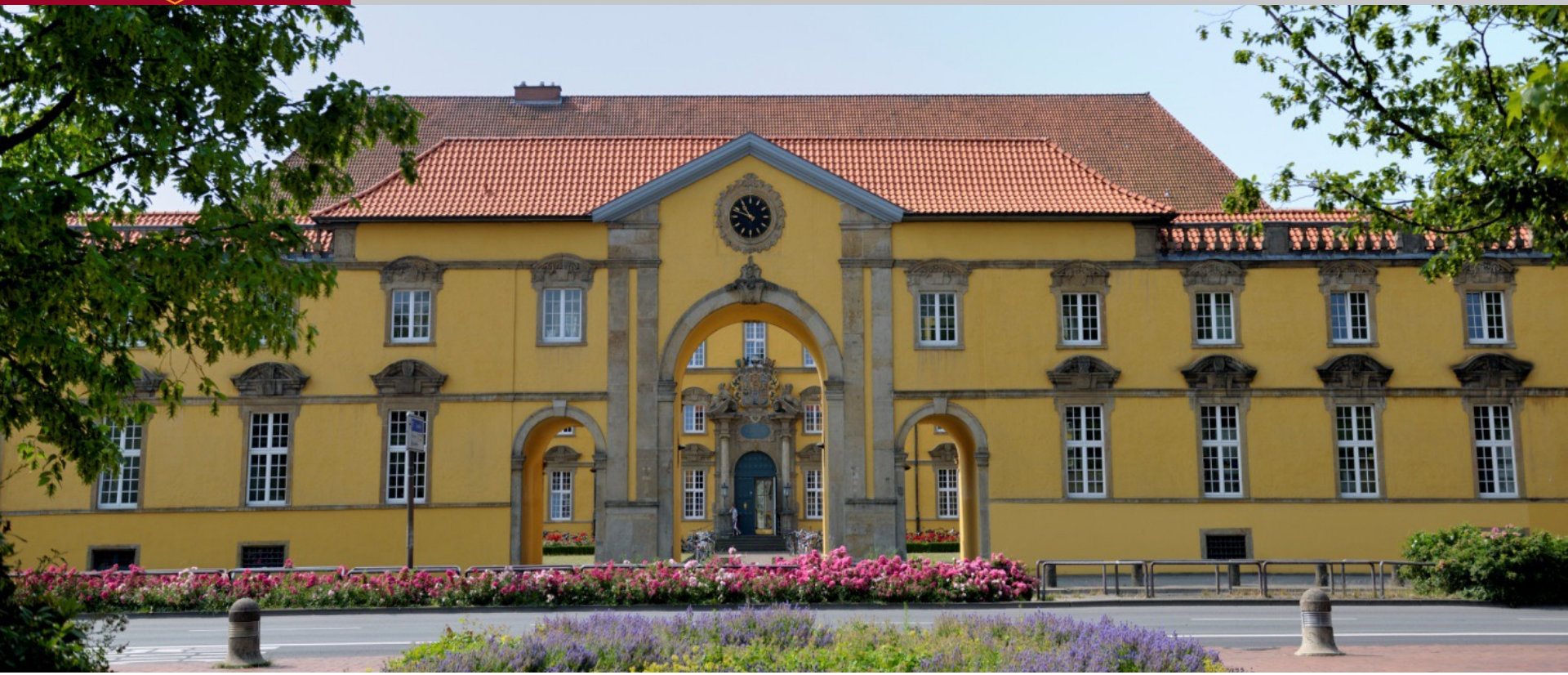

Basic Programming in Python

4. Session: Functions

Nohayr Muhammad Abdelmoneim

Summer Term 2023

May 15th, 2023

Overview

- Recap on methods
- Functions vs. Methods
- How to define Functions
- How to call Functions
- Variable Scope
- Recursive Functions

Math module methods

Method	Description
math.acos()	Returns the arc cosine of a number
math.acosh()	Returns the inverse hyperbolic cosine of a number
math.asin()	Returns the arc sine of a number
math.asinh()	Returns the inverse hyperbolic sine of a number
math.atan()	Returns the arc tangent of a number in radians
math.atan2()	Returns the arc tangent of y/x in radians
math.atanh()	Returns the inverse hyperbolic tangent of a number
math.ceil()	Rounds a number up to the nearest integer
math.comb()	Returns the number of ways to choose k items from n items without repetition and order
math.copysign()	Returns a float consisting of the value of the first parameter and the sign of the second parameter
math.cos()	Returns the cosine of a number
math.cosh()	Returns the hyperbolic cosine of a number
math.degrees()	Converts an angle from radians to degrees
math.dist()	Returns the Euclidean distance between two points (p and q), where p and q are the coordinates of that point

https://www.w3schools.com/python/module_math.asp

String Methods

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case

https://www.w3schools.com/python/python_strings_methods.asp

List Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

https://www.w3schools.com/python/python_lists_methods.asp

Built in functions

Function	Description
abs()	Returns the absolute value of a number
all()	Returns True if all items in an iterable object are true
any()	Returns True if any item in an iterable object is true
ascii()	Returns a readable version of an object. Replaces none-ascii characters with escape character
bin()	Returns the binary version of a number
bool()	Returns the boolean value of the specified object
bytearray()	Returns an array of bytes
bytes()	Returns a bytes object
callable()	Returns True if the specified object is callable, otherwise False
chr()	Returns a character from the specified Unicode code.
classmethod()	Converts a method into a class method
compile()	Returns the specified source as an object, ready to be executed
complex()	Returns a complex number
delattr()	Deletes the specified attribute (property or method) from the specified object
dict()	Returns a dictionary (Array)
dir()	Returns a list of the specified object's properties and methods

https://www.w3schools.com/python/python_ref_functions.asp

What is a function/method?

- A function is a block of code given a certain name, that performs a specific task.
- A function can take one or more parameters/arguments as an input.
- A function can return one or more variables as an output.
- A function is defined once and can be used many times.
- A function is only executed when it is “called”

Functions Vs. Methods

- A function doesn't need any object and is independent.
- A method is a function that is linked with an object.
- We can directly call the function with its name.
- A method is called by the object's name.

- Function

```
x=[1,2,3,4,5]
print(len(x))
```

```
5
```

```
x=[1,2,3,4,5]
print(x.len())
```

```
---------------------------------------------------------
AttributeError                            Traceback (m
Cell In[9], line 2
      1 x=[1,2,3,4,5]
----> 2 print(x.len())

AttributeError: 'list' object has no attribute 'len'
```

- Method

```
y="HeLLo"
print(lower(y))
```

```
---------------------------------------------------------
NameError                                 T
Cell In[7], line 2
      1 y="HeLLo"
----> 2 print(lower(y))

NameError: name 'lower' is not defined
```

```
y="Hello"
print(y.lower())
```

```
hello
```

User defined functions

```
def FunctionName(#optional set or parameters):
    #code
```

- Defining a function

```
def HelloWorld():
    print("This is my first function")
    print("Hello world")
```

```
def MySum(x,y):
    print(x+y)
```

- Calling a function

```
HelloWorld()
```

```
This is my first function
Hello world
```

```
MySum(2,3)
```

```
5
```

```
a=4
b=5
MySum(a,b)
```

```
9
```

```
MySum(y=7,x=2)
```

```
9
```

Return value

- A function can either perform a block of code without returning a value, or it can return one or more values
- There is a difference between printing a value within a function and returning a value

```
def MySum(x,y):
    return x+y

print("Calling a function which returns a value")
MySum(2,3)
print("Do you see an output?")
```

```
Calling a function which returns a value
Do you see an output?
```

```
print(MySum(5,6))
print("How about now?")
```

```
11
How about now?
```

Return many values

- A function can return more than a value.
- All values can be used when calling, according to the required task

```
def ReturnMany():
    x=3
    y=4
    z=5
    return x,y,z
```

```
print(ReturnMany())
```

```
(3, 4, 5)
```

```
a,b,c=ReturnMany()
print(a,b,c)
```

```
3 4 5
```

Return Note

- Return statement terminates the function and returns the value.
- I.e., a code written after return will not be executed (unless within a condition)

```
def MySum():
    return 2+3
    print("A statement after return")
```

```
print(MySum())
```

```
5
```

```
def MySum(x,y):
    if x==2:
        return x+y
    else:
        return "A statement after return"
```

```
print(MySum(4,5))
```

```
A statement after return
```

Print vs. return

```
def ReturnSum():
    return 2+3
```

```
def PrintSum():
    print(2+3)
```

```
print("Calling ReturnSum:")
x=ReturnSum()
print("x=: ",x)
print("Calling PrintSum:")
y=PrintSum()
print("y=: ",y)
```

```
Calling ReturnSum:
x=:  5
Calling PrintSum:
5
y=:  None
```

Default arguments

- Default arguments are used in case the user didn't input parameter values.

```
def MySum(x,y):
    print(x+y)

MySum()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 4
      1 def MySum(x,y):
      2     print(x+y)
----> 4 MySum()

TypeError: MySum() missing 2 required positional arguments: 'x' and 'y'
```

```
def MySum(x,y):
    print(x+y)

MySum(5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recen
Cell In[30], line 4
      1 def MySum(x,y):
      2     print(x+y)
----> 4 MySum(5)

TypeError: MySum() missing 1 required positional argument: 'y'
```

Default arguments

- Default arguments are used in case the user didn't input parameter values.

```
def MySum(x,y=3):
    print(x+y)

MySum(5)
```

```
8
```

```
def MySum(x=1,y=3):
    print(x+y)

MySum()
```

```
4
```

```
def MySum(x=2,y):
    print(x+y)

MySum(5)
```

```
  Cell In[31], line 1
    def MySum(x=2,y):
                  ^
SyntaxError: non-default argument follows default argument
```

Variable scope (local vs. global)

- Variable scope determines which part of the program it's defined and can be used at.
- A variable which is created inside a function, is only defined within the function.

- Local variable

```
def my_sum():
    x=5
    y=7
    print(x+y)
```

```
my_sum()
print(x,y)
```

```
12
---------------------------------------------------------------------------
NameError
Cell In[8], line 2
      1 my_sum()
----> 2 print(x,y)

NameError: name 'x' is not defined
```

- Global variable

```
x=5
y=7
def my_sum():
    print(x+y)
```

```
my_sum()
print(x,y)
```

```
12
5 7
```

global keyword

The global keyword is used to define a global variable that is defined throughout the program.

```
def my_sum():
    global x
    x=5
    y=7
    print(x+y)
```

```
my_sum()
print(x)
print(y)
```

```
12
5

---------------------------------------
NameError
Cell In[4], line 3
      1 my_sum()
      2 print(x)
----> 3 print(y)

NameError: name 'y' is not defined
```

Recursive functions

- A recursive function is a function that calls itself inside its body.
- A recursive function can be considered as a loop. Hence, a condition must be specified to stop execution.

```
def MyFunction():
    MyFunction()
```

Recursive factorial

- $n! = n \times n-1 \times n-2 \times \cdots \times 1$
- $n! = n \times (n-1)!$

```
def RecFact(n):
    return n*(RecFact(n-1))
```

```
RecFact(5)
```

```
---------------------------------------------------------------------------
RecursionError                                Traceback (most recent c
Cell In[41], line 1
----> 1 RecFact(5)

Cell In[40], line 2, in RecFact(n)
      1 def RecFact(n):
----> 2     return n*(RecFact(n-1))

Cell In[40], line 2, in RecFact(n)
      1 def RecFact(n):
----> 2     return n*(RecFact(n-1))

    [... skipping similar frames: RecFact at line 2 (2970 times)]

Cell In[40], line 2, in RecFact(n)
      1 def RecFact(n):
----> 2     return n*(RecFact(n-1))

RecursionError: maximum recursion depth exceeded
```

Recursive factorial

- $n! = n \times n-1 \times n-2 \times \cdots \times 1$
- $n! = n \times (n-1)!$

```
def RecFact(n):
    if n==1:
        return 1
    return n*(RecFact(n-1))

print(RecFact(5))
```

```
120
```

- Tracing:

$$RecFact(5) \quad =120$$

Is n==1? No

$$RecFact(5) = 5 \times RecFact(4) \quad \text{(=120 / =24)}$$

Is n==1? No

$$RecFact(4) = 4 \times RecFact(3) \quad \text{(=24 / =6)}$$

Is n==1? No

$$RecFact(3) = 3 \times RecFact(2) \quad \text{(=6 / =2)}$$

Is n==1? No

$$RecFact(2) = 2 \times RecFact(1) \quad \text{(=2 / =1)}$$

Is n==1? Yes

$$RecFact(1) = 1$$

Recursive factorial Notes

- $n! = n \times n-1 \times n-2 \times \cdots \times 1$
- $n! = n \times (n-1)!$

```
def RecFact(n):
    if n==1:
        return 1
    return n*(RecFact(n-1))

print(RecFact(5))
```

```
120
```

- Notes:
  - Can you see the difference between print and return?
  - What would happen if we replace return with print?
  - Can you modify the function so that it only accepts positive numbers?
  - Does the order within the function matter?
  - What would happen if we switch the order of if statement and return?

Print vs. return

- Print: only prints certain values/messages to the user.
- Print doesn't return an actual value. I.e., printed values are not usable.
- return: returns an actual value that can be used in any operation.
- Using return replaces the entire function with the returned value

```
def RecFact(n):
    if n==1:
        print(1)
    print(n*(RecFact(n-1)))
```

```
RecFact(5)
```

```
1
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent c
Cell In[47], line 1
----> 1 RecFact(5)

Cell In[46], line 4, in RecFact(n)
      2 if n==1:
      3     print(1)
----> 4 print(n*(RecFact(n-1)))

Cell In[46], line 4, in RecFact(n)
      2 if n==1:
      3     print(1)
----> 4 print(n*(RecFact(n-1)))

    [... skipping similar frames: RecFact at line 4 (2969 times)]

Cell In[46], line 4, in RecFact(n)
      2 if n==1:
      3     print(1)
----> 4 print(n*(RecFact(n-1)))

Cell In[46], line 2, in RecFact(n)
      1 def RecFact(n):
----> 2     if n==1:
      3         print(1)
      4     print(n*(RecFact(n-1)))

RecursionError: maximum recursion depth exceeded in comparison
```

Order of code

- When switching orders, the function keeps looping and never executes the if statement.

```
def RecFact(n):
    return n*(RecFact(n-1))
    if n==1:
        return 1
```

```
RecFact(5)
```

```
---------------------------------------------------------------------------
RecursionError                                Traceback (most recent ca
Cell In[45], line 1
----> 1 RecFact(5)

Cell In[44], line 2, in RecFact(n)
      1 def RecFact(n):
----> 2     return n*(RecFact(n-1))
      3     if n==1:
      4         return 1

Cell In[44], line 2, in RecFact(n)
      1 def RecFact(n):
----> 2     return n*(RecFact(n-1))
      3     if n==1:
      4         return 1

    [... skipping similar frames: RecFact at line 2 (2970 times)]

Cell In[44], line 2, in RecFact(n)
      1 def RecFact(n):
----> 2     return n*(RecFact(n-1))
      3     if n==1:
      4         return 1

RecursionError: maximum recursion depth exceeded
```

QUESTIONS?