Python Programming for Novice

Malvika Sharan & Olav Vahtras

Workshop: Software Writing Skills for Young Researchers
2015.09.23 - 2015.09.25

GFZ Helmholtz-Zentrum Potsdam, Germany

Python Programming for Novice

Malvika Sharan & Olav Vahtras

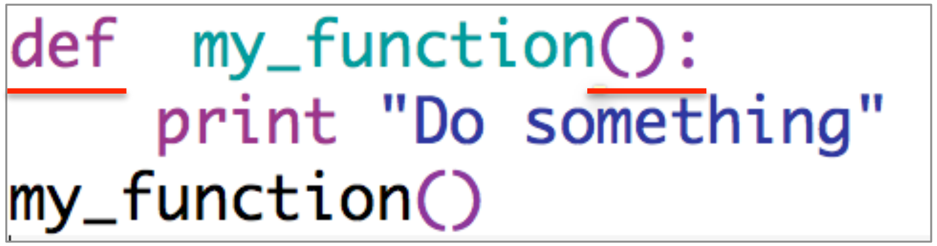
Day – 2: Session – 2

2015-09-24

Functions

- Functions are the reusable block of code that you can name
- Using the name, a function can be executed any number of times
  - This reusability is called calling the function
- You have been using built-in functions already
  - len(), range(), sorted(), max(), min(), sum() etc.
- Function is important building block of a software
- Structure of writing a function:
  - def (keyword) + function name (you choose) + ():
  - newline with 4 spaces or a tab + block of code
  - Call your function

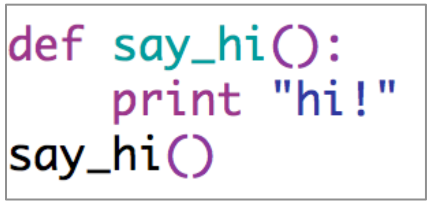
Note: Codes at the 0 position are always read



```
def  my_function():
    print "Do something"
my_function()
```

Functions

- Non parametric function



```
def say_hi():
    print "hi!"
say_hi()
```

Functions

- Non parametric function

```
def say_hi():
    print "hi!"
say_hi()
```

- Parametric function

```
def say_hi(name):
    print "Hi %s!" % name
name = 'Greg'
say_hello(name)
```

Functions

- Non parametric function

```
def say_hi():
    print "hi!"
say_hi()
```

- Parametric function

```
def say_hi(name):
    print "Hi %s!" % name
name = 'Greg'
say_hello(name)
```

- Return values

```
def say_hi(name):
    comment = "Hi %s!" % name
    return comment

name = 'Greg'
print say_hi(name)
```

Functions

- Non parametric function

```
def say_hi():
    print "hi!"
say_hi()
```

- Parametric function

```
def say_hi(name):
    print "Hi %s!" % name
name = 'Greg'
say_hello(name)
```

- Return values
  - Local vs. global variables?

```
def say_hi(name):
    comment = "Hi %s!" % name
    return comment

name = 'Greg'
print say_hi(name)

name2 = 'Wilson'
new_comment = say_hi(name2)
print "Use %s here" % new_comment
```

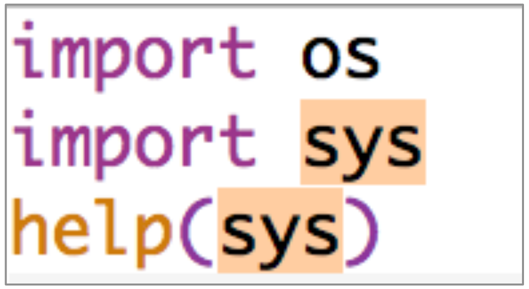
Do something with the values

Function Exercises

- Let’s take our older codes (if.py or for.py) and write them in function

Packages

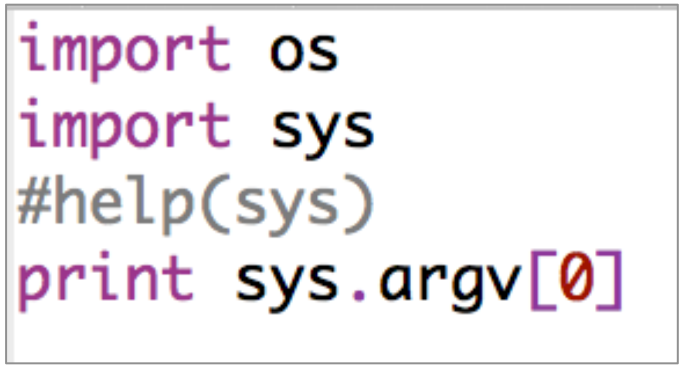
- A module is a Python file that (generally) has only definitions of variables, functions, and classes.
- Like functions, which are usable parts of a program, packages (also known as libraries) are reusable programs with several modules
- Many powerful tools are built into Python which can be imported without rewriting (or even reading) them in the current program
  - Import (keyword) + package name: e.g.: import os, import sys
  - More at: https://pypi.python.org/pypi



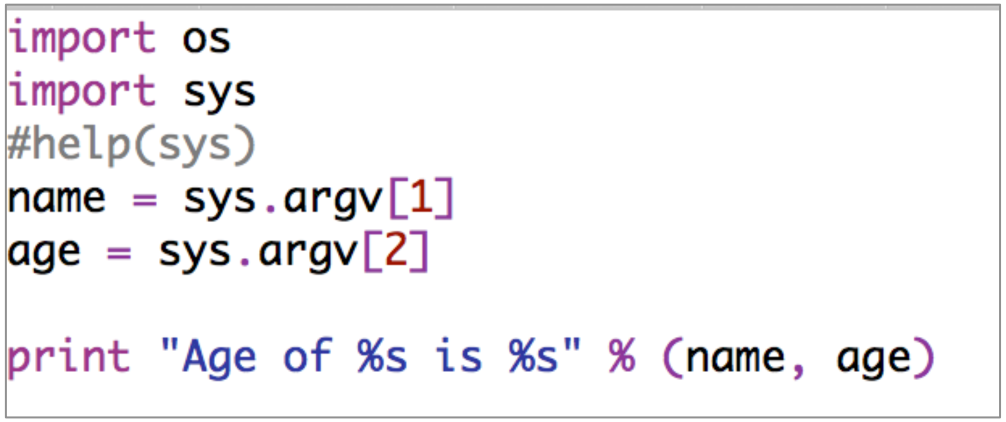
```
import os
import sys
help(sys)
```

Packages

- To known about the imported modules, type help(package name): help(sys)
  - argv -- command line arguments; argv[0] is the script pathname if known
    - python scriptname.py
  - Therefore anything written after the script pathname can be accessed as list like argv[1] and argv[2]
    - python scriptname.py input1 input2



```
import os
import sys
#help(sys)
print sys.argv[0]
```

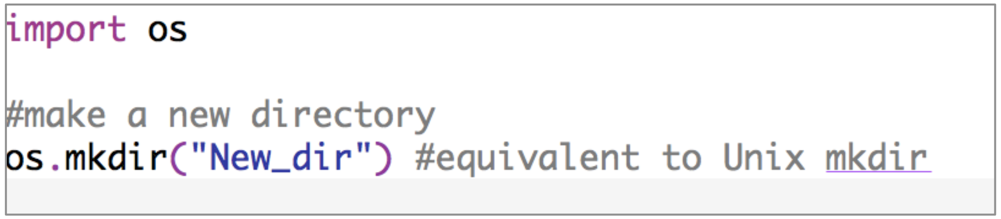


```
import os
import sys
#help(sys)
name = sys.argv[1]
age = sys.argv[2]

print "Age of %s is %s" % (name, age)
```

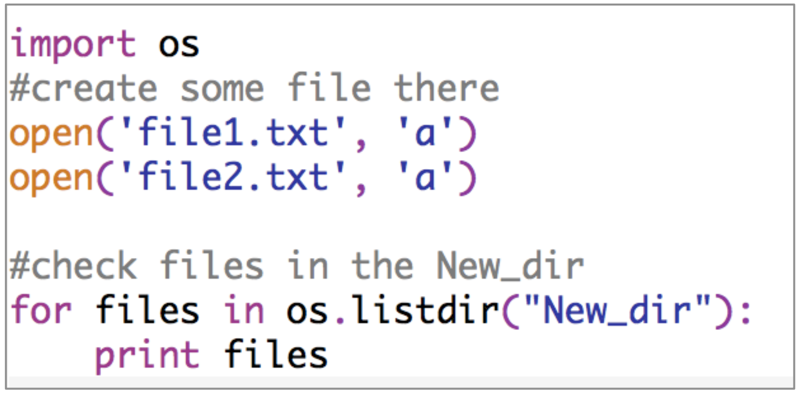
Packages

- Package os, help(os)



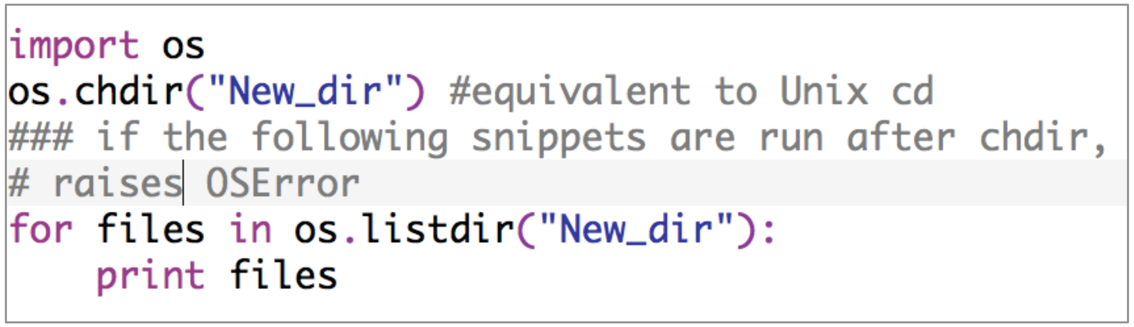
```
import os

#make a new directory
os.mkdir("New_dir") #equivalent to Unix mkdir
```



```
import os
#create some file there
open('file1.txt', 'a')
open('file2.txt', 'a')

#check files in the New_dir
for files in os.listdir("New_dir"):
    print files
```

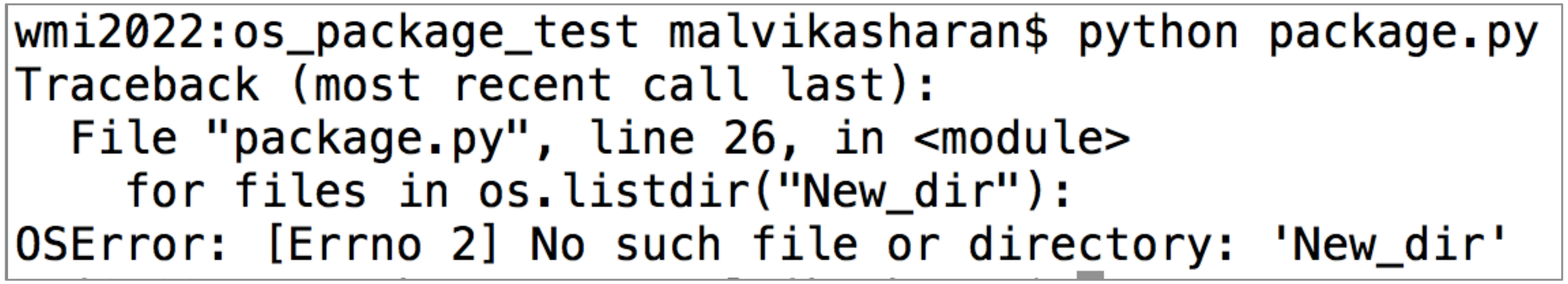


```
import os
os.chdir("New_dir") #equivalent to Unix cd
### if the following snippets are run after chdir,
# raises OSError
for files in os.listdir("New_dir"):
    print files
```

Moving to error handling for a while

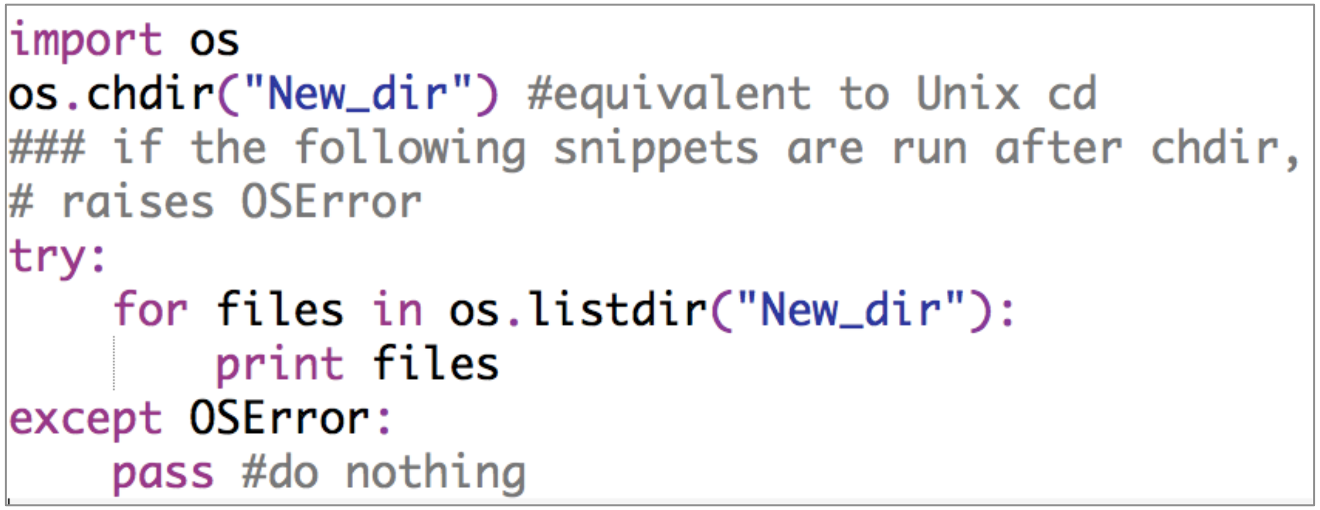
Errors

- Errors and error handlers



```
wmi2022:os_package_test malvikasharan$ python package.py
Traceback (most recent call last):
  File "package.py", line 26, in <module>
    for files in os.listdir("New_dir"):
OSError: [Errno 2] No such file or directory: 'New_dir'
```

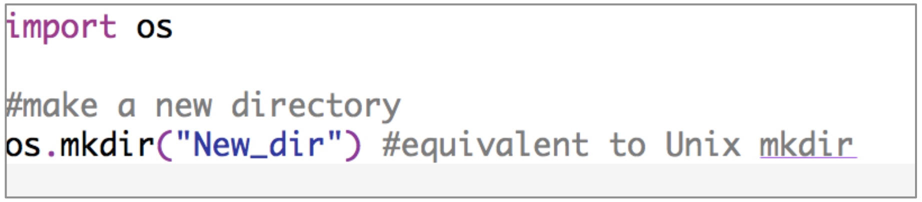
- Handle exception by try and except



```
import os
os.chdir("New_dir") #equivalent to Unix cd
### if the following snippets are run after chdir,
# raises OSError
try:
    for files in os.listdir("New_dir"):
        print files
except OSError:
    pass #do nothing
```

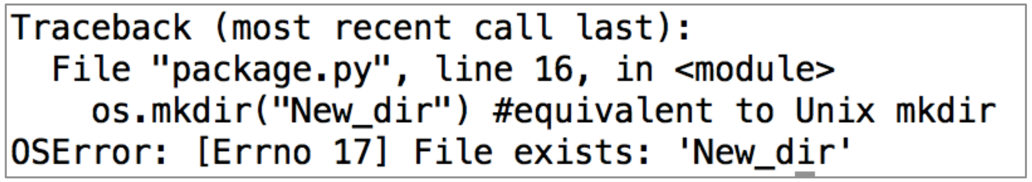
Errors

- Try creating existing folder



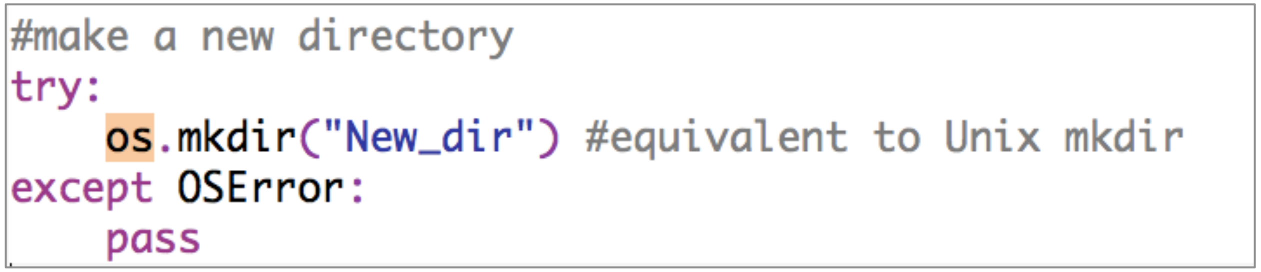
```python
import os

#make a new directory
os.mkdir("New_dir") #equivalent to Unix mkdir
```



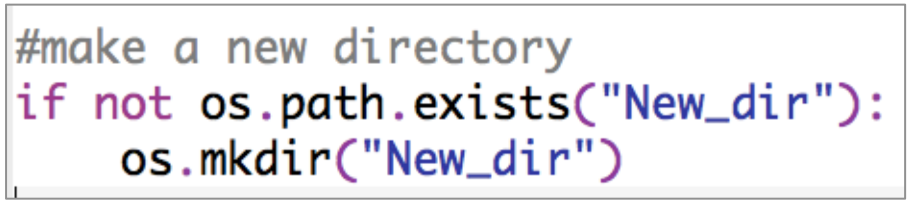
```
Traceback (most recent call last):
  File "package.py", line 16, in <module>
    os.mkdir("New_dir") #equivalent to Unix mkdir
OSError: [Errno 17] File exists: 'New_dir'
```

- try and except



```python
#make a new directory
try:
    os.mkdir("New_dir") #equivalent to Unix mkdir
except OSError:
    pass
```

- Alternate option:



```python
#make a new directory
if not os.path.exists("New_dir"):
    os.mkdir("New_dir")
```

Back to package

Packages

- The package matplotlib is widely used to visualize data as plots
  - import matplotlib
- Specific modules (program) from the packages are imported as
  - from matplotlib import pyplot
  - import matplotlib.pyplot
- Plot some data

Python Discussion

- Is there something we need to revisit?
- Do we have any trivial problem that everyone faces?
- Was there something you wanted to learn and we did not cover so far?

Python Programming for Novice

Malvika Sharan & Olav Vahtras

Day – 2: Session – 3

2015-09-24

Task for the last session on Sept 24

- From your current scientific interests identify/create a task
  - That involves repeated tasks like reading one or multiple files of same format
  - Requires you to extract certain information
  - Requires processing of the data like using certain formula for calculations
  - Requires you to create a new file with the processed information

- Hands-on practice on your own project