IPython table, pandas

https://stackoverflow.com/questions/26873127/show-dataframe-as-table-in-ipython-notebook

Programming: NullReferenceException

- https://stackoverflow.com/questions/4660142/what-is-a-nullreferenceexception-and-how-do-i-fix-it

Memory: Stack vs Heap

- https://stackoverflow.com/questions/79923/what-and-where-are-the-stack-and-heap

Python IDE

PyCharm

Good Discussion on best Python IDE

Python Tutorials

1. Tutorials point Pandas
2. Online Terminals
3. Screen Scraping HTML table to CSV
4. Python Tutorial with Quizes
5. Python Sockets Explained
6. *args **kwargs
7. Major Python Libraries

row string

string literals

r before string

Python Packaging

https://www.programiz.com/python-programming/package

Delete Imported Module

https://stackoverflow.com/questions/437589/how-do-i-unload-reload-a-python-module

Jupyter Python - output hyperlink

In [2]:
```
from IPython.core.display import display, HTML
display(HTML("""<a href="https://google.com">Go to Google.com</a>"""))
```

Go to Google.com

In [1]:
```
#!C:\Users\bilsto1\Anaconda2\envs\py3Env\Scripts\activate
!cd
```

```
C:\Users\bilsto1\Cookbook\Tutorials
```

*args & **kwargs

- pass a variable number of arguments to a function

args

- *args is used to send a non-keyworded variable-length argument list to the function

****kwargs***

- **kwargs allows you to pass **keyworded** variable-length of arguments to a function
- You should use **kwargs if you want to handle named arguments in a function

practice

http://thepythonguru.com/python-args-and-kwargs/

Converting list to *args in Python

https://stackoverflow.com/questions/3941517/converting-list-to-args-in-python

```
timeseries_list = [timeseries1 timeseries2 ...]
r = scikits.timeseries.lib.reportlib.Report(*timeseries_list)
```

How to make a call

https://docs.python.org/2/reference/expressions.html#calls

In [10]:
```
def func(*args):
    print("***************")
    for item in args:
        print(item)
```

```
func([2,3,4])
func(*[2,3,4])
func(range(2, 5))
func(*range(2, 5))
```

```
**************
[2, 3, 4]
**************
2
3
4
**************
range(2, 5)
**************
2
3
4
```

In [11]:

```
def test_var_args(f_arg, *argv):
    print("first normal arg:", f_arg)
    for arg in argv:
        print("another arg through *argv :", arg)

test_var_args('yasoob','python','eggs','test')



def greet_me(**kwargs):
    if kwargs is not None:
        for key, value in kwargs.items():
            print("%s == %s" %(key,value))

greet_me(name="yasoob")



def greet_me_2(*args):
    if args is not None:
        print("*FIRST WAY OF UNPACKING*****")
        print(args[0])
        print(args[1])
        for item in args:
            print("*SECOND WAY OF UNPACKING*****")
            print(item)


names = {"name": "yasoob", 'lang':'python'}
status = {"1": "success", '2':'fail'}
greet_me_2(names, status)
```

```
first normal arg: yasoob
another arg through *argv : python
another arg through *argv : eggs
another arg through *argv : test
name == yasoob
*FIRST WAY OF UNPACKING*****
{'name': 'yasoob', 'lang': 'python'}
{'1': 'success', '2': 'fail'}
*SECOND WAY OF UNPACKING*****
{'name': 'yasoob', 'lang': 'python'}
```

```
*SECOND WAY OF UNPACKING*****
{'1': 'success', '2': 'fail'}
```

In [12]:

```python
def some_func(fargs,*args,**kwargs):
    print("fargs: ", fargs)
    print("********")
    for item in args:
        print("args: ", item)
    print("********")
    for item in kwargs:
        print("kwargs: ", item)
    print("********")


names = {"name": "yasoob", 'lang':'python'}

some_func(5, 1, 2, 3, names)
```

```
fargs:  5
*******
args:  1
args:  2
args:  3
args:  {'name': 'yasoob', 'lang': 'python'}
*******
*******
```

Closure in Python

Corey Schafer https://www.youtube.com/watch?v=FsAPt_9Bf3U&t=42s

In [6]:

```python
def outer_f(msg):
    def inner_f():
        print(msg)

    return inner_f

hi_func = outer_f('Hi')
bye_func = outer_f('Bye')

hi_func()
bye_func()
```

```
Hi
Bye
```

Decorator Function in Python

Corey Schafer https://www.youtube.com/watch?v=FsAPt_9Bf3U&t=42s

In [10]:

```python
def decorator_f(original_f):
    def wrapper_f():
        print('wrapper executed before {}'.format(original_f.__name__))
        return original_f()
    return wrapper_f
```

```
@decorator_f
def display_f():
    print("display function ran")

#decorated_display_f = decorator_f(display_f)
#decorated_display_f()

display_f()
```

```
wrapper executed before display_f
display function ran
```

In [11]:

```
def decorator_f(original_f):
    def wrapper_f(*args, **kwargs):
        print('wrapper executed before {}'.format(original_f.__name__))
        return original_f(*args, **kwargs)
    return wrapper_f

@decorator_f
def display_f():
    print("display function ran")

@decorator_f
def display_info(name, age):
    print("display info ran with arguments ({}, {})".format(name, age))

display_f()
display_info("John", 25)
```

```
wrapper executed before display_f
display function ran
wrapper executed before display_info
display info ran with arguments (John, 25)
```

Decorator Class in Python

Corey Schafer https://www.youtube.com/watch?v=FsAPt_9Bf3U&t=42s

In [14]:

```
class decorator_class(object):
    def __init__(self, original_f):
        self.original_f = original_f

    def __call__(self, *args, **kwargs):
        print('call method executed before {}'.format(self.original_f.__name_
        return self.original_f(*args, **kwargs)

@decorator_class
def display_f():
    print("display function ran")

@decorator_class
def display_info(name, age):
    print("display info ran with arguments ({}, {})".format(name, age))

display_f()
display_info("John", 25)
```

```
call method executed before display_f
display function ran
```

```
call method executed before display_info
display info ran with arguments (John, 25)
```

Guido Answer - not to lose self

http://neopythonic.blogspot.ca/2008/10/why-explicit-self-has-to-stay.html

Call a Python script with arguments from another script

https://stackoverflow.com/questions/1186789/what-is-the-best-way-to-call-a-python-script-from-another-python-script

```
import os
os.system("python myOtherScript.py arg1 arg2 arg3")

# our example is executing dummy3 that is calling dummy2
os.system('python dummy2.py a b c')
```

In [2]:
```
!python dummy3.py
```

In [3]:
```
import dummy3
```

In [4]:
```
!python dummy1.py
```

In [5]:
```
!python dummy2.py 1 2 3
```