Module inspect exploration

August 5, 2017

```
In [1]: import inspect

In [2]: def powerI(a, b = 0, *c, d, e = 1, **f ):
            pass

In [3]: pw_sig = inspect.signature(powerI)

In [4]: #inspect.signature(powerI)
        type(pw_sig)

Out[4]: inspect.Signature

In [5]: #inspect.signature(powerI)
        print(pw_sig)

(a, b=0, *c, d, e=1, **f)


In [6]: isinstance(pw_sig,str)

Out[6]: False

In [7]: pw_para = pw_sig.parameters

In [8]: #signature.parameters
        print(pw_para)

OrderedDict([('a', <Parameter at 0x48677e0 'a'>), ('b', <Parameter at 0x4867d80 'b'>), ('c', <


In [9]: #signature.parameters
        type(pw_para)

Out[9]: mappingproxy

In [10]: #def powerI(a, b = 0, *c, d, e = 1, **f ):
         #    pass
         #pw_para.items() is inspect.signature.parameters.items()
         for k, v in pw_para.items():
             print('key: {}'.format(k))
             print('type(key): {}'.format(type(k)))
             print('value: {}'.format(v))
             print('type(value): {}'.format(type(v)))
             print('inspect.Parameter.kind: \n{}'.format(v.kind))
             print('inspect.Parameter.default: \n{}\n\n'.format(v.default))
```

```
key: a
type(key): <class 'str'>
value: a
type(value): <class 'inspect.Parameter'>
inspect.Parameter.kind:
POSITIONAL_OR_KEYWORD
inspect.Parameter.default:
<class 'inspect._empty'>


key: b
type(key): <class 'str'>
value: b=0
type(value): <class 'inspect.Parameter'>
inspect.Parameter.kind:
POSITIONAL_OR_KEYWORD
inspect.Parameter.default:
0


key: c
type(key): <class 'str'>
value: *c
type(value): <class 'inspect.Parameter'>
inspect.Parameter.kind:
VAR_POSITIONAL
inspect.Parameter.default:
<class 'inspect._empty'>


key: d
type(key): <class 'str'>
value: d
type(value): <class 'inspect.Parameter'>
inspect.Parameter.kind:
KEYWORD_ONLY
inspect.Parameter.default:
<class 'inspect._empty'>
```

```
key: e
type(key): <class 'str'>
value: e=1
type(value): <class 'inspect.Parameter'>
inspect.Parameter.kind:
KEYWORD_ONLY
inspect.Parameter.default:
1


key: f
type(key): <class 'str'>
value: **f
type(value): <class 'inspect.Parameter'>
inspect.Parameter.kind:
VAR_KEYWORD
inspect.Parameter.default:
<class 'inspect._empty'>
```

```
In [11]: def powerII(a):
             pass
         pwI_para = inspect.signature(powerII).parameters
         for k, v in pwI_para.items():
             print('key: {}'.format(k))
             print('type(key): {}'.format(type(k)))
             print('value: {}'.format(v))
             print('type(value): {}'.format(type(v)))
             print('inspect.Parameter.kind: \n{}'.format(v.kind))
             print('inspect.Parameter.default: \n{}\n\n'.format(v.default))
```

```
key: a
type(key): <class 'str'>
value: a
type(value): <class 'inspect.Parameter'>
inspect.Parameter.kind:
POSITIONAL_OR_KEYWORD
inspect.Parameter.default:
<class 'inspect._empty'>
```

```
In [12]: def powerIII(a,b,*):
```

```
            pass
        pwI_para = inspect.signature(powerIII).parameters
        for k, v in pwI_para.items():
            print('key: {}'.format(k))
            print('type(key): {}'.format(type(k)))
            print('value: {}'.format(v))
            print('type(value): {}'.format(type(v)))
            print('inspect.Parameter.kind: \n{}'.format(v.kind))
            print('inspect.Parameter.default: \n{}\n\n'.format(v.default))
```

```
      File "<ipython-input-12-74fb7cf6a170>", line 1
    def powerIII(a,b,*):
                    ^
SyntaxError: named arguments must follow bare *
```

```
In [13]: help(inspect.Parameter)
```

```
Help on class Parameter in module inspect:

class Parameter(builtins.object)
 |  Represents a parameter in a function signature.
 |
 |  Has the following public attributes:
 |
 |  * name : str
 |      The name of the parameter as a string.
 |  * default : object
 |      The default value for the parameter if specified.  If the
 |      parameter has no default value, this attribute is set to
 |      `Parameter.empty`.
 |  * annotation
 |      The annotation for the parameter if specified.  If the
 |      parameter has no annotation, this attribute is set to
 |      `Parameter.empty`.
 |  * kind : str
 |      Describes how argument values are bound to the parameter.
 |      Possible values: `Parameter.POSITIONAL_ONLY`,
 |      `Parameter.POSITIONAL_OR_KEYWORD`, `Parameter.VAR_POSITIONAL`,
 |      `Parameter.KEYWORD_ONLY`, `Parameter.VAR_KEYWORD`.
 |
 |  Methods defined here:
 |
 |  __eq__(self, other)
 |
 |  __init__(self, name, kind, *, default, annotation)
 |
 |  __repr__(self)
 |
 |  __str__(self)
 |
 |  replace(self, *, name=<class 'inspect._void'>, kind=<class 'inspect._void'>, annotation=<cl
 |      Creates a customized copy of the Parameter.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  annotation
 |
 |  default
 |
 |  kind
 |
 |  name
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  KEYWORD_ONLY = <_ParameterKind: 'KEYWORD_ONLY'>
 |
 |  POSITIONAL_ONLY = <_ParameterKind: 'POSITIONAL_ONLY'>
 |
 |  POSITIONAL_OR_KEYWORD = <_ParameterKind: 'POSITIONAL_OR_KEYWORD'>
 |
 |  VAR_KEYWORD = <_ParameterKind: 'VAR_KEYWORD'>
 |
 |  VAR_POSITIONAL = <_ParameterKind: 'VAR_POSITIONAL'>
 |
 |  __hash__ = None
 |
 |  empty = <class 'inspect._empty'>
```

```
In [14]: dir(pw_sig)
```

```
Out[14]: ['__class__',
          '__delattr__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__gt__',
          '__hash__',
          '__init__',
          '__le__',
          '__lt__',
          '__module__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__setattr__',
          '__sizeof__',
          '__slots__',
          '__str__',
          '__subclasshook__',
          '_bind',
          '_bound_arguments_cls',
          '_parameter_cls',
          '_parameters',
          '_return_annotation',
          'bind',
          'bind_partial',
          'empty',
          'from_builtin',
          'from_function',
          'parameters',
          'replace',
          'return_annotation']
```