Functions

Functions

A function definition defines a user-defined function object

Functions

```
>>> def func():
...     """ Dummy function. Returns nothing """
...     pass


>>> type(func)
<class 'function'>

>>> func.__name__
'func'

>>> func.__doc__ # help(func)
' Dummy function. Returns nothing '

>>> hash(func)
8783939476151
```

Functions

```
>>> def sum(a, b):
...     return a + b


>>> sum(a=1, b=2)
3

>>> sum(1, b=2)
3
```

Functions

```
>>> sum.__annotations__
{}
```

Functions

Type annotation is not mandatory but recommended for readability purposes, built-in IDE lex-analyzers

…

For mypy is mandatory

```
>>> def sum(a: int, b: int) -> int:
...     return a + b
```

```
>>> sum.__annotations__
{'a': <class 'int'>, 'b': <class 'int'>, 'return': <class
'int'>}
```

Defaults

```
def student(first_name, last_name, grade=5):
    """ Function returns tuple with first name, last name and grade """
    return first_name, last_name, grade


>>> print(student("Mark", "Walters"))
('Mark', 'Walters', 5)

>>> print(student("Hugo", "Smith", 3))
('Hugo', 'Smith', 3)

>>> print(student(first_name="Hugo", last_name="Smith", 5))
Syntax Error: positional argument follows keyword argument (<input>, line
1)
```

Defaults

```
def concat_and_multiply(lst_1, lst_2, number=1):
    """ Function concats two lists and multiply them by `number` """
    return (lst_1 + lst_2) * number


>>> concat_and_multiply([1, 2, 3, 4, 5], [11, 12, 1, 1, 1], 1)
[1, 2, 3, 4, 5, 11, 12, 1, 1, 1]
```

Defaults

```
def concat_and_multiply(lst_1, lst_2, *, number=1):
    """ Function concats two lists and multiply them by `number` """
    return (lst_1 + lst_2) * number
```

```
>>> concat_and_multiply([1, 2, 3, 4, 5], [11, 12, 1, 1, 1], 1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: concat_and_multiply() takes 2 positional arguments but 3 were
given

concat_and_multiply() takes 2 positional arguments but 3 were given
```

Defaults

```
def concat_and_multiply(lst_1, lst_2, *, number=1):
    """ Function concats two lists and multiply them by `number` """
    return (lst_1 + lst_2) * number


>>> concat_and_multiply([1, 2, 3, 4, 5], [11, 12, 1, 1, 1], number=1)
[1, 2, 3, 4, 5, 11, 12, 1, 1, 1]
```

Defaults

```
from typing import List

def append_to_list(element: int, lst: List = []) -> List:
    lst.append(element)
    return lst

>>> append_to_list(10)
[10]

>>> append_to_list(12)
[10, 12]
```

Defaults

```
from typing import List

def append_to_list(element: int, lst: List = []) -> List:
    lst.append(element)
    return lst

>>> append_to_list.__defaults__
([10, 12],)
```

*args, **kwargs

```
def sum(*args: int) -> int:
    result = 0
    for number in args:
        result += number
    return result
```

```
>>> sum(1, 2)
3
>>> sum(1, 2, 3)
6
>>> sum(1, 2, 3, 4)
10
```

*args, **kwargs

```
def get_args_kwargs(a, b, *args, c, d, **kwargs):
    return args, kwargs
```

```
>>> get_args_kwargs(1, 2, 3, 4, 5, c=1, d=2, key='value')
((3, 4, 5), {'key': 'value'})
```

Call stack

```
def fib(n: int) -> int:
    """ Returns n th Fibonacci sequence element """
    if n <= 1:
        return n
    return fib(n - 1) + fib(n - 2)

>>> fib(10)
55
```

Call stack

```
>>> fib(10000)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 5, in fib
  File "<stdin>", line 5, in fib
  File "<stdin>", line 5, in fib
  [Previous line repeated 2980 more times]
  File "<stdin>", line 3, in fib
RecursionError: maximum recursion depth exceeded in comparison

maximum recursion depth exceeded in comparison
```

Call stack

```
>>> import sys
... print(sys.getrecursionlimit())
3000

>>> sys.setrecursionlimit(100000)

>>> fib(10000)
KeyboardInterrupt
```

Call stack

```
def fib(n: int) -> int:
    """ Returns n th Fibonacci sequence element """
    if n <= 1:
        return n

    a, b = 1, 1
    for i in range(2, n):
        c = a + b
        a, b = b, c
    return b
```

Call stack

```
>>> fib(10000)
33644764876431783266621612005107543310302148460680063906564769974680081442166662368155595513633734025582065332680836159373734790483865268263040892463056431887354544369559827491606602099884183933864652731300088830269235673613135117579297437854413752130520504347701602264758318906527890855154366159582987279682987510631200575428783453215515103870818298969791613127856265033195487140214287534698187962046936097879900350962302291026368131493195275630227837628441540360584402572114334961180023091208287046088 9...
```