python3 in one pic

Function

Decorator

```
# Bye log~
# **********
# Hey log~
#
This is fb!
# Bye log~
```

```
def
fa():
    print("This is fa!")

# Equal to...
def fb():

    print("This is fb!")
fb = log(fb)

fa()
print("*"*10)
fb()
```

Lambda

```
pairs = [(1, 'one'), (2, 'two'), (3, 'three'),
(4, 'four')]

pairs.sort(key=lambda pair: pair[1])
print(pairs)
"[(4,
'four'), (1, 'one'), (3, 'three'), (2, 'two')]"

pairs.sort(key=lambda
pair: pair[0])
print(pairs)
"[(1, 'one'), (2,
'two'), (3, 'three'), (4, 'four')]"
```

Arguments

```
def f(*args, **kargs):
    print("args ", args)

    print("kargs ", kargs)
    print("FP: {} &
Scripts: {}".format(kargs.get("fp"), "/".join(args)))

f("Python",
"Javascript", ms = "C++", fp = "Haskell")
# args  ('Python',
'Javascript')
# kargs  {'ms': 'C++', 'fp': 'Haskell'}
#
FP: Haskell and Scripts: Python/Javascript
```

arbitrary arguments

```
def f(*args, con = " & "):
    print(isinstance(args,
tuple))
    print("Hello", con.join(args))

f("Python",
"C", "C++", con = "/")
# True
# "Hello
Python/C/C++"
```

keyword arguments

```
def f(v, l = "Python"):
    """return
'$v, $l'"""
    return "{}, {}!".format(v,
l)
```

Native Datatypes

Casting

```
print(str(3.14))    # "3.14"
print(str(3))       # "3"
print(str([1,2,3]))
  # "[1,2,3]"
print(str((1,2,3)))  # "(1,2,3)"
print(str({1,2,3}))
  # "{1,2,3}"
print(str({'python':
'*.py', 'javascript': '*.js'}))
"{'python':
'*.py', 'javascript': '*.js'}"
```

Byte

list of ascii character

```
byt = b'abc'
print(type(byt))   # <class 'bytes'>
print(byt[0]
== 'a')# False
print(byt[0] == 97) # True
```

Length

```
print(len(byt))    # 3
```

Boolean

True

False

```
print(type(True))      # <class 'bool'>
```

None

```
print(None is None)      # True
print(type(None))        # <class 'NoneType'>
```

List

```
l = ['python', 3, 'in', 'one']
print(type(l))
      # <class 'list'>
```

Length

```
print(len(l))      # 4
```

Slicing

```
print(l[0])      # 'python'
print(l[-1])     # 'one'
print(l[1:-1])
    # [3, 'in']
```

```
Lappend('pic')     # None
# l == ['python',
3, 'in', 'one', 'pic']
Linsert(2,
'.4.1')  # None
# l == ['python', 3, '.4.1', 'in',
'one', 'pic']
Lextend(['!', '1'])
#
l == ['python', 3, '.4.1', 'in',
```