



11 things I wish I
knew when I knew 0



(11)₆ things I wish I
knew when I knew 0



Hello!

I am Orfeas



ECE @ AUTH



Python & Django



10 KΣ!

F-Strings

String formatting before and after Python 3.6

%-formatting

```
>>> name = "Eric"  
>>> age = 74  
>>> "Hello, %s. You are %s." % (name, age)  
'Hello Eric. You are 74.'
```

%-formatting

```
>>> first_name = "Eric"
>>> last_name = "Idle"
>>> age = 74
>>> profession = "comedian"
>>> affiliation = "Monty Python"
>>> "Hello, %s %s. You are %s. You are a %s. You were a member of %s.
" % (first_name, last_name, age, profession, affiliation)
'Hello, Eric Idle. You are 74. You are a comedian. You were a member
of Monty Python.'
```

str.format()

```
>>> "Hello, {}. You are {}".format(name, age)
'Hello, Eric. You are 74.'
```

```
>>> person = {'name': 'Eric', 'age': 74}
>>> "Hello, {name}. You are {age}.".format(name=person['name'], age=person['age'])
'Hello, Eric. You are 74.'
```

```
>>> person = {'name': 'Eric', 'age': 74}
>>> "Hello, {name}. You are {age}.".format(**person)
'Hello, Eric. You are 74.'
```

str.format()

```
>>> first_name = "Eric"
>>> last_name = "Idle"
>>> age = 74
>>> profession = "comedian"
>>> affiliation = "Monty Python"
>>> print(("Hello, {first_name} {last_name}. You are {age}. " +
>>>        "You are a {profession}. You were a member of {affiliation}.") \
>>>        .format(first_name=first_name, last_name=last_name, age=age, \
>>>                profession=profession, affiliation=affiliation))
```


F-strings!

```
>>> name = "Eric"
>>> age = 74
>>> f"Hello, {name}. You are {age}."
'Hello, Eric. You are 74.'
```

```
>>> def to_lowercase(input):
...     return input.lower()

>>> name = "Eric Idle"
>>> f"{to_lowercase(name)} is funny."
'eric idle is funny.'
```

Black

The uncompromising Python code formatter

No configuration!

- Εγκατάσταση:

```
pip install black
```

- Εκτέλεση:

```
black {source_file_or_directory}
```

**For with enumerate,
for/else**

enumerate

```
colors = ['red', 'green', 'blue', 'yellow']
```

```
i = 1
for color in colors:
    print i + 1, '--->', colors[i]
    i += 1
```

```
for i, color in enumerate(colors):
    print i + 1, '--->', color
```

enumerate

```
colors = ['red', 'green', 'blue', 'yellow']
```

```
i = 1
for color in colors:
    print i + 1, '--->', colors[i]
    i += 1
```

```
for i, color in enumerate(colors, 1):
    print i, '--->', color
```



for/else

```
for item in container:
    if search_something(item):
        # Found it!
        process(item)
        break
else:
    # Didn't find anything...
    not_found_in_container()
```

```
if __name__ ==  
    "__main__":
```

What is it and why is it everywhere?

What is it?

- When the python interpreter reads a file:
 - Sets special variables (including `__name__`)
 - Runs the file code
- If you run the file:
`$ python foo.py`  `__name__ = "__main__"`
- If the file is imported:
`import foo`  `__name__ = "foo"`

Why use it?

Both used by other programs/modules as a module and run as the main program. Examples:

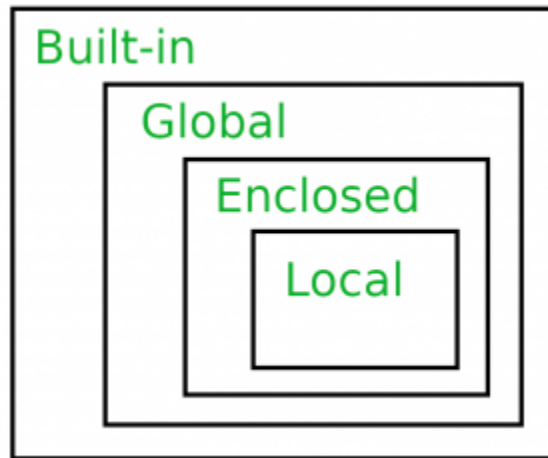
- Library, with unit tests or demo
- Main program, but testing requires importing the file
- Mostly main program, with programmer-friendly API for advanced users

Python scopes

LEGB and common mistakes

LEGB Rule

1. **L**ocal
2. **E**nclosing-function
3. **G**lobal (module)
4. **B**uilt-in (Python)



Mistake w/ variable

```
>>> x = 10
>>> def bar():
...     print(x)
>>> bar()
10
```

```
>>> x = 10
>>> def foo():
...     print(x)
...     x += 1
>>> foo()
```

Traceback (most recent call last):

...

UnboundLocalError: local variable 'x'
referenced before assignment

Mistake w/ list

```
>>> lst = [1, 2, 3]
>>> def foo1():
...     lst.append(5)
...
>>> foo1()
>>> lst
[1, 2, 3, 5]
```

```
>>> lst = [1, 2, 3]
>>> def foo2():
...     lst += [5]
...
>>> foo2()
Traceback (most recent call last):
...
UnboundLocalError: local variable 'lst' referenced before assignment
```

Counting with dictionaries

`setdefault()`, `get()` and `collections.defaultdict`

Basic way

```
message = 'It was a bright cold day in April,'\n          'and the clocks were striking thirteen.'
```

```
count = {}  
for character in message:  
    if character not in count.keys():  
        count[character] = 1  
    else:  
        count[character] += 1
```


• get()

```
count = {}  
for character in message:  
    count[character] = count.get(character, 0) + 1
```

- Returns a default value if key doesn't exist
- Doesn't alter the dictionary

.setdefault()

```
count = {}  
for character in message:  
    count.setdefault(character, 0)  
    count[character] += 1
```

- If the key doesn't exist, the value is added to the dictionary
- Doesn't overwrite

defaultdict()

```
import collections
count = collections.defaultdict(int)
for character in message:
    count[character] += 1 # all keys have a default already
```

- Creates keys on explicit access

Which to choose?

`.setdefault()`

- Works with dynamic default values
- Expression always evaluated

`defaultdict()`

- Only creates keys on explicit access
- Overwrites keys
- Great for static default values

Mutable default function arguments

As a function argument

The problem

```
def foo(bar=[]):  
    bar.append("baz")  
    return bar
```

```
>>> foo()  
["baz"]  
>>> foo()  
["baz", "baz"]  
>>> foo()  
["baz", "baz", "baz"]
```

Workaround

```
>>> def foo(bar=None):  
...     if bar is None:  
...         bar = []  
...     bar.append("baz")  
...     return bar  
...  
>>> foo()  
["baz"]  
>>> foo()  
["baz"]  
>>> foo()  
["baz"]
```



Thanks!

Any questions?

You can find me at orfeas@orfeasa.com

Further reading

1. F-strings
 - <https://realpython.com/python-f-strings/>
 - <https://www.python.org/dev/peps/pep-0498/>
2. Black
 - <https://github.com/psf/black>
3. `__name__ = "__main__"`
 - <https://stackoverflow.com/questions/419163/what-does-if-name-main-do>
 - https://www.geeksforgeeks.org/what-does-the-if-__name__-__main__-do/
 - https://thepythonguru.com/what-is-if-__name__-__main__/

Further reading

4. Scopes
 - <https://www.toptal.com/python/top-10-mistakes-that-python-programmers-make>
mistake 4
 - <https://www.geeksforgeeks.org/scope-resolution-in-python-legb-rule/>
5. For with enumerate/else
 - <https://pythontips.com/2015/04/19/nifty-python-tricks/>
 - <https://book.pythontips.com/en/latest/enumerate.html>
 - https://book.pythontips.com/en/latest/for_-_else.html
6. Setdefault/Defaultdict
 - <https://automatetheboringstuff.com/chapter5/>
 - <https://stackoverflow.com/questions/3483520/use-cases-for-the-setdefault-dict-method>
 - <https://docs.python.org/2/library/collections.html#collections.defaultdict>
7. Empty mutable default arguments
 - <https://www.toptal.com/python/top-10-mistakes-that-python-programmers-make>
common mistake 1
 - <https://effbot.org/zone/default-values.htm>