PEP 202 Python List Comprehension

- Alex Cordero
- Zak Collins
- Matt Cirbo
- Team Pep 202

About PEP 202

- List Comprehensions make creating lists easier
- Provides a quick way to initialize a list.

About PEP 202

```
list = [i for i in range (2, 100)]
```

- This produces a list with all the numbers between 2 and 100

About PEP 202

- Can create complex lists in a single line
- Helps reduce total lines of code written

Advantage: Easy Initialization

- Format allows you to easily initialize a new list
- List comprehensions save many lines of code

Advantage: List Comprehension Vs. For Loop

```
list = [i for i in range (2, 100)]

list = []
for i in range(2,100):
    list.append(i)
```

Advantage: Easy Initialization

- Creating a list from a previously defined list is much easier
- It is much easier to apply a filter to a list through a comprehension then through a for loop

Advantage: Easy Initialization From a Previous List

```
squarelist = [x*x for x in prevlist if (x%2 != 0)]
```

vs.

```
prevlist = []
squarelist = []
for x in prevlist:
   if(x%2 != 0):
        squarelist.append(x * x)
```

Advantage

- “Part of the motivation for list comprehensions as I recall is that we’d like to get away from such a strong reliance on map, reduce and filter.” -Skip Montanaro

```
list = map(someFunction, ValuesToList)
```

Disadvantage: Less Readable

- Not immediately apparent what the code is doing
- Code is very dense
- ListA = [j for i in range(2, 8) for j in range(i*2, 50, i)]
- ListB = [x for x in range(2, 50) if x not in ListA]

Disadvantage: Confusing For Learning Python

- Another syntax to learn when learning python
- Syntax isn't self-explanatory

Disadvantage: Commitment To Backwards Compatibility

- “You can’t take it(List Comprehension) out, because you’re commited to backwards compatability.” - Moshe Zadka
- Python is committed to backwards compatibility
- Features can never be removed once added

Case Match

- A change we think would be good would be adding case matches
- Case Matches would make it easy to implement a list with multiple filters

Current Syntax

```
list = [i*i for i in range(100) if i % 3 if i % 2]
```

- The current syntax with multiple if will only create a list of numbers less than one hundred that is divisible by 6

New Proposed Syntax

- The new proposed syntax will act as a logical or and will make it easy to add multiple filters to each case.
- This syntax will square all numbers divisible by 3 and cube all numbers divisible by 2 list = [i*i, i*i*i for i in range(100) case i%3 == 0 case i%2 ==0]

Disadvantage

- An obvious disadvantage of our new proposed syntax is that it will make the list comprehension even less readable than it previously was.
- Another obvious is problem is Python does not support case matches outside of switch statements.

Conclusion: Pros

- Fast way to create complex lists
- Create lists from other lists
- Less reliance on less powerful map and filter functions

Conclusion: Cons

- Less readable
- Hard for beginners
- Can never be removed

Conclusion

- Proposal was accepted
- Pros outweigh the cons
- Making the language more powerful and versatile is more important

Credits

- https://docs.python.org/2/library/functions.html#filter
- http://markmail.org/message/qguevwxeprbg75mn#query:+page:1 +mid:ctfezaiyl3iy3b47+state:results
- http://www.secnetix.de/olli/Python/list _ comprehensions.hawk
- http://www.pythonforbeginners.com/basics/list-comprehensions-in-python