CS218 - Data Structures
FAST NUCES Peshawar Campus
Dr. Nauman (recluze.net)

September 3, 2019

1 Circular Linked List in Python

Raster images of the notebook 06-circular-linked-list

Stack

A stack is essentially free in Python. Here's what happens when we use Python's list.

In [ ]:
```
s = []

s.append(12)   # append is the same as push
s.append(5)
s.append(55)

print(s)

print(s.pop())
print(s.pop())
print(s.pop())
print(s.pop())     # <-- IndexError
```

In [ ]:
```
s = []
s.append(1)
s.append(2)
s.append(3)
print(s)
print(s[1])    # <-- in a stack, we shouldn't be able to do this!
```

Writing Our Own

In [ ]:
```
class Stack:
    def __init__(self):
        self.l = []

    def push(self, val):
        self.l.append(val)

    def pop(self):
        return self.l.pop()

    def peek(self):
        return self.l[-1]
```

In [ ]:
```
s = Stack()
s.push(1)
s.push(2)
s.push(3)
# print(s[1])    # <--- error .... which, for us, is success since we have a stack and that's what we want.

# print(s.l[1])
# print(s.pop())
```

In [ ]:
```
print(s.peek())
print(s.pop())
print(s.peek())
print(s.pop())
```

Case Study: Bracket Matching

In [ ]:
```
a = '123'
b = '456'
dict(zip(a, b))
```

In [ ]:
```
opening = '({['
closing = ')}]'
mapping = dict(zip(opening, closing))
print(mapping)
mapping['{']
```

In [ ]:
```
def is_matched(string):
    opening = '({['
    closing = ')}]'
    mapping = dict(zip(opening, closing))


    stack = []

    for c in string:
        # case 1
        if c not in mapping.values() and c not in mapping.keys():
            continue

        # case 2
        # automatically checks only starting brackets
        if c in mapping:
            stack.append(mapping[c])   # we'll be looking for corresponding closing bracket later

        # case 3: has to be closing bracket if we get here
        elif len(stack) == 0 or c != stack.pop():
            return False

    return len(stack) == 0
```

In [ ]:
```
string = "{[[]]{()}}"
is_matched(string)
```

In [ ]:
```
string = "2 + (3 * 5) * ((2 * 2) + 5)"
is_matched(string)
```

In [ ]:
```
string = "2 + (3 * 5) * ((2 * 2) + 5) )"
is_matched(string)
```

Case Study: Binary to Decimal Conversion

In [ ]:
```
def dec_to_bin(num):
    s1 = []

    while num != 0:
        remainder = num % 2   # reminder will always be 0 or 1
        num = num // 2        # division by 2 gets rid of the least significant binary digit 101 // 2 = 10
        s1.append(remainder)

    ret = ''
    while s1:
        ret += str(s1.pop())  # comes out in reverse order of course

    return ret
```

In [ ]:
```
dec_to_bin(0)
```