Sid Bhatia
CS 115
Professor Nicolaou
10/14/21

CS 115 Study Sheet

"I pledge my honor that I have abided by the Stevens Honor System."

① Python Basics: String arithmic: 'yum' + 'my' = 'yummy' → concatenation
'yum' * 3 = 'yumyumyum'

· Lists: len(list) = length · last index + 1 · myList = [1,2,3] myList += [4,5,6]
→ list concatenation. myList = [1,2,3,4,5,6] ; myList * 3 = [1,2,3,1,2,3,1,2,3]
[list[-1] = last element] · list[:-1] = returns list · [listEl] = casts element as a list
· range(start, stop, step) → [inclusive, exclusive]

② · Please · // = floor or integer division (9 // 2 = 4)
· % = mod = remainder · Head of list = L[0], tail of list = L[1:]
· range() gives range object so need to use list() function

③ Binary: · Decimal → Binary: divide by 2 | remainder · Russian Peasant
e.s. 181_{10} to $?_2$:

90	1
45	0
22	1
11	0
5	1
2	1
1	0
0	1

∴ $181_{10} = (1011\ 0101)_2$

· Binary → Decimal:
· e.s. 0111, 0011 $_2$ to $?_{10}$
$2^7\ 2^6\ 2^5\ 2^4\ \ 2^3\ 2^2\ 2^1\ 2^0$
: 64+32+16+2+1
= 112+2+1 : 115_{10}

· Binary Addition: if 2 → place 0, carry 1
e.s. 0101 1001 + 0111 1011
· if 3 → place 1, carry 1
· if 4 → place 0, carry 2, carry 1
: 1101 0100$_2$
: 128+64+16−4 = 212_{10}

· Russian Peasant: on left side: add odd #s on right side
· e.s. (21 * 14) ×2
21 14
10 28
5 56
2 112
1 224
= 112 + 28 + 7
= 147

④ Map/Reduce/Filter: list(map(function, list)) → one argument
↳ applies function to every element in list
= list output
· reduce from functools
· reduce(binary function, list) → e.s. reduce(lambda x,y: x+y, [1,2,3,4,5])
↳ function w/ two arguments → # output
3+3=6 → 6+4=10 → 10+5=15
· list(map(lambda x: x**2, [2,4,6,8,10])) → [4, 16, 36, 64, 100]
· filter(function, list) → true or false function e.s., list(filter(lambda x: x % 2 == 0, range(0,11)))
= [0,2,4,6,8,10] → returns even #s in list from [0,10]

⑤ Lambda expressions: lambda x: x**2 → function
↳ variable

⑥ Recursion: ⓐ Base Case ⓑ Recursive Call

use-it-on-lose-it = head-end tail recursion

```
def fac(n):
    if n == 1:
        return 1
    else:
        return n * fac(n-1)
# 5 * 4 * 3 * 2 * 1 = 120
```

```
def use_it_on_lose_it(L):
    if not L:  # len(L) == 0 or L == []
        return []
    else:
        head = L[0]
        tail = L[1:]
        if (condition...):
            head_contribution = use_it_on_lose_it(head)
        ... return head_contribution + tail_contribution
```

```
else:
    head_contribution =
    tail_contribution = use_it_on_lose_it(tail)
    return head_contribution + tail_contribution
```