Week 3: Agenda

- Quiz #2: Grades will be released before Tuesday
- This week: Quiz #3 – Loops
- Loops (review)
- Strings

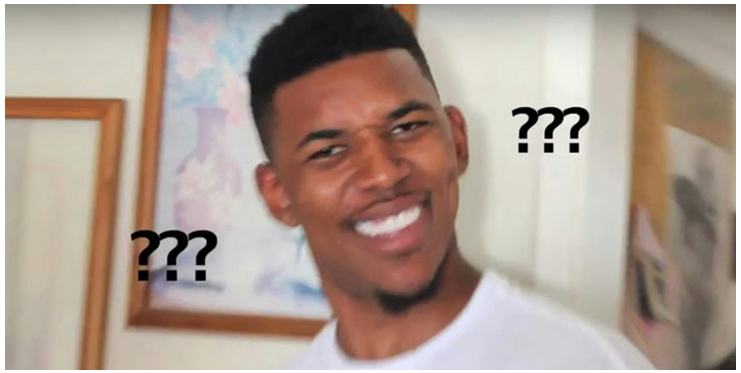
nth Number Problems

- ThreeOddy numbers
  - Multiple of 3
  - Only odd digits
- Examples:
  - is 3 a ThreeOddy? **YES**
  - is 5 a ThreeOddy? **NO**
  - is 42 a ThreeOddy? **NO**
  - is 15 a ThreeOddy? **YES**

Considering only positive integers:

What's the 0th ThreeOddy number? **3**

What's the 1st ThreeOddy number? **9**



because in computer science you start with 0th

Another example: 7ish numbers

- Non-negative number, and the sum of its digits a is multiple of 7
  - Examples:
    - 61: 6 + 1 = 7
    - 86: 8 + 6 = 14
    - 489: 4 + 8 + 9 = 21
- First 16 7ish numbers

0
7
16
25
34
43
52
59
61
68
70
77
86
95
106
115

`nth...` problems

- Other examples: `nthCircularPrime, nth7ish, ...`
- Recap: `nth7ish`
  - Part 1: Write the function `is7ish(n)`, which takes a non-negative integer **`n`** and returns **`True`** if **`n`** is a 7ish number and **`False`** otherwise.
  - Part 2: Write the function `nth7ish(n)`which takes a non-negative integer **`n`** and returns the nth *7ish* number.
    - `nth7ish(0)` should return 0, the first 7ish number.
    - `nth7ish(1)` returns 7.

0
7
16
25
34
43
52
59
61
68
70
77
86
95
106
115

7ish

Position	Number
0	0
1	7
2	16
3	25
4	34
5	43
6	52
7	59
8	61
9	68
10	70
11	77
12	86
13	95
14	106
15	115

Another loop problem: printMysteryShape(n)

- printMysteryShape(5)

```
0
1 *
2 * *
3 * * *
4 * * * *
```

- printMysteryShape(9)

```
0
1 *
2 * *
3 * * *
4 * * * *
5 * * * * *
6 * * * * * *
7 * * * * * * *
8 * * * * * * * *
```

String Operations

- Indexing:

How do we get the first character in a string?

```
s[0]
```

How do we get the last character in a string?

```
s[len(s) - 1]
```

What happens if we try an index outside of the string?

```
s[len(s)] # runtime error
```

String Operations

- Slicing:

Slices are exactly like ranges – they can have a **start**, an **end**, and a **step**. But slices are represented as numbers inside of **square brackets**, separated by **colons**.

```
s = "abcde"
print(s[2:len(s):1]) # print "cde"
print(s[0:len(s)-1:1]) # prints "abcd"
print(s[0:len(s):2]) # prints "ace"
```

Check

- Given the string `s="abcdefghij"`, what slice would we need to get the string `"cfi"`?

Announcement

Thursday sessions are now in Room 2152

Code Tracing: Loops

```
def ct3(z):
    total = 0
    for y in range(z,1,-1):
        if (y % 2 == 0):
            print('skip y =', y)
            continue
        total += y
        if (total > 20):
            print('break at y =', y)
            break
    return total
print(ct3(10))
```

Loop problems: Finding “runs”

What is a Run of Digits?

A run of digits is a sequence of consecutive, identical digits within a number.

For example:

112233 has runs: **11**, **22**, and **33**

4445551 has runs: **444**, **555**, and **1**

77777 has one run: **77777**

Each time a digit changes, a new run starts.

Example: Finding runs of a specific digit **d**

Given a number and a specific digit d, we want to find the length of the longest run of d, meaning the sequence where d appears consecutively the most times.

Examples:

Number: 100220022000, d=0

Runs of 00, 00, 000 **Longest run**: 000 (length=3)

Number: 333355533311, d=3

Runs of: 3333, 333 **Longest run**: 3333 (length=4)

Number: 456789, d=5

Runs of 5: 5 **Longest run**: 5 (length=1)

String built-in methods

String built-in methods work differently from built-in functions. Instead of writing:

```
isdigit(s)
```

we have to write:

```
s.isdigit()
```

This tells Python to call the built-in string function `isdigit()` **on the string** `s`. It will then return a result normally.

String functions

Some string functions return information about the string.

`s.isdigit()`, `s.islower()`, and `s.isupper()` return `True` if the string is all-digits, all-lowercase, or all-uppercase, respectively.

`s.count(c)` returns the number of times the character `c` occurs in `s`.

`s.find(c)` returns the index of the character `c` in `s`, or `-1` if it doesn't occur in `s`.

Loops

```
1 s = "hello 15112"
2 for c in s:
3     print(f"This is character {c}")
4
5 for w in s.split():
6     print(f'This is word {w}')
7
```

From ASCII to Unicode

- Teleprinters had to “agree” how to convert codes to characters and vice versa

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Example: `largestNumber(s)`

- Write the function `largestNumber(s)` that takes a string `s` and returns the largest integer value that occurs within that text, or None if no such value occurs.
- You may assume that numbers in the text are non-negative integers.
- You may assume that numbers are always composed of consecutive digits delimited by spaces.

```
assert(largestNumber("I saw 3 dogs, 17 cats, and 14 cows!") == 17)
assert(largestNumber("I saw four dogs") == None)
```

Example: `largestNumber(s)`

- Observation: Extracting numbers is similar to finding “runs”

Example: `leastFrequentLetters(s)`

- Write the function `leastFrequentLetters(s)`, that takes a string `s`, and ignoring case (so `"A"` and `"a"` are treated the same), returns a string containing the least-frequent alphabetic letters that occur in s, in the same order.
- Each least-frequent letter is included only once in the result and in alphabetical order.
- Digits, punctuation, and whitespace are not letters!
- If `s` has no alphabetic characters, the result should be the empty string ("").
- Example:

```
leastFrequentLetters("aDq efQ? FB'daf!!!") == "eB"
```

Example: `leastFrequentLetters(s)`

- Break down the problem into small subproblems
  a. Find the lowest frequency > 0
     - "aDq efQ? FB'daf!!!" -> lowest frequency is 1
     - "aaabBcc" -> lowest frequency is 2 (both b and c occur 2 times and no letter occurs with frequency 1)
  b. Extract the letters that occur with the lowest frequency
     - "aDq efQ? FB'daf!!!" -> find letters with count = lowest frequency (2)
       - result: “eB”
     - "aaabBcc" , lowest frequency is 2
       - result: “bBcc”