

On your way in...

Pick-Up:

1. HW 03
2. POGIL 19: Tuples (+ POGIL 10: Nested Loops Excerpt)

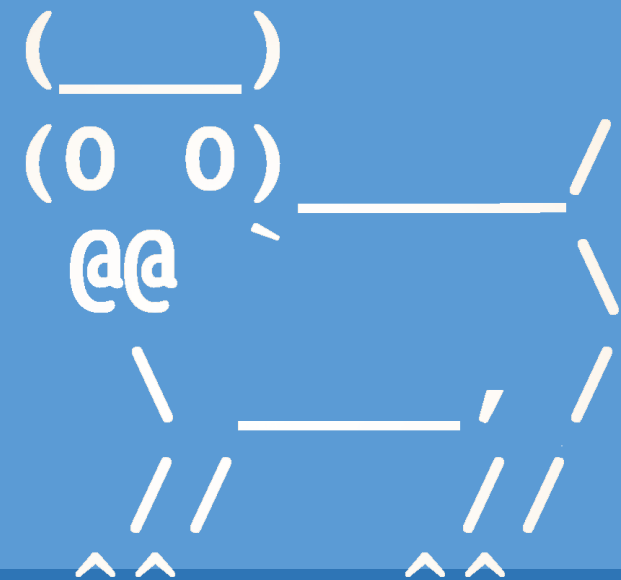
Please note that Lab02 moonAge grades are available on
GitLab



Welcome to CS 134!

Introduction to Computer Science
Iris Howley

-Tuples & Mutability-



Format Printing

```
print("{} was born on {}/{/}/{}".format("Pixel", 5, 16, 2018))
```

Pixel was born on 5/16/2018

This will print the same exact text:

```
name = "Pixel"
```

```
month = 5
```

```
day = 16
```

```
year = 2018
```

```
print("{} was born on {}/{/}/{}".format(name, month, day, year))
```

TODAY'S LESSON

Mutability

(Some objects can be modified, some cannot!)

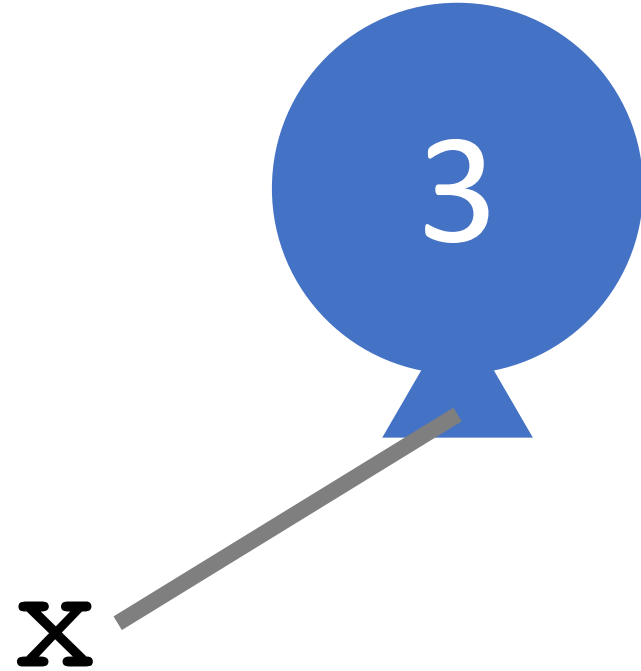
A Balloon Metaphor...

>>> 3



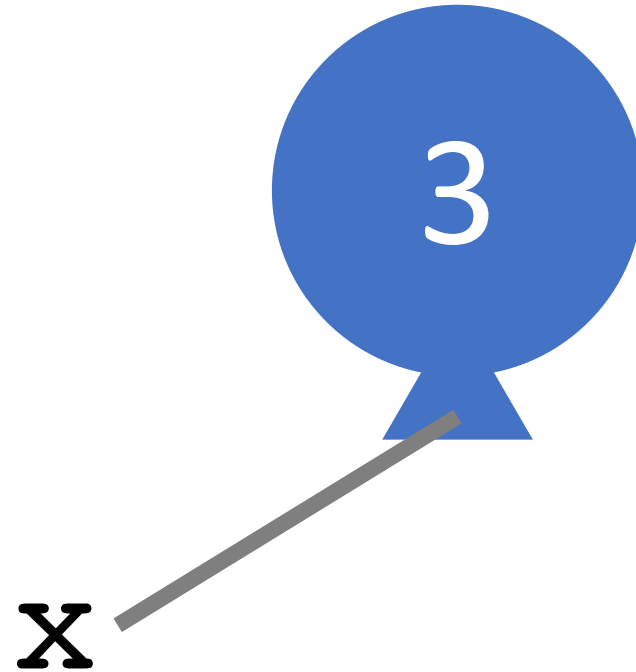
A Balloon Metaphor...

```
>>> x = 3
```



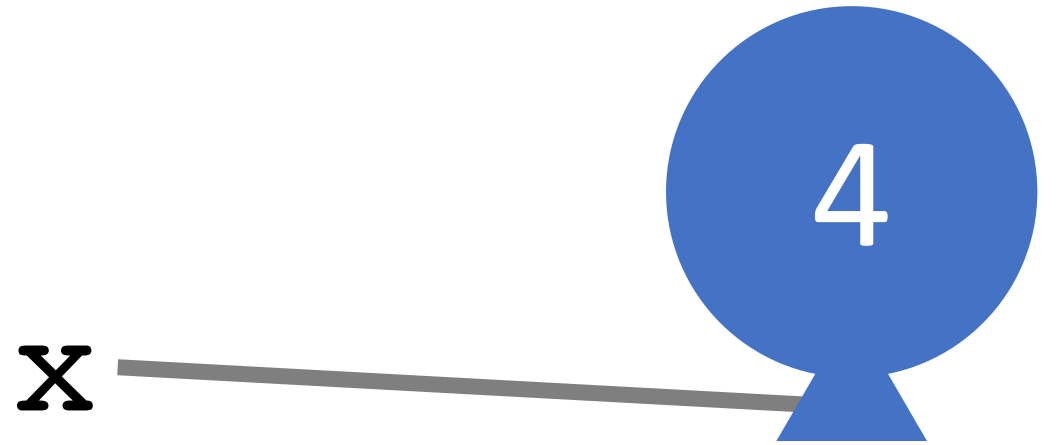
A Balloon Metaphor...

```
>>> x = x + 1
```



A Balloon Metaphor...

```
>>> x = x + 1
```



NUMBERS ARE IMMUTABLE

A Balloon Metaphor...

>>> [5,16,18]



A Balloon Metaphor...

```
>>> mylist = [5,16,18]
```

`mylist`



5,16,
18

A Balloon Metaphor...

```
>>> mylist.append('dogge')
```



mylist

LISTS ARE MUTABLE

A Tale of Two Mutabilities...

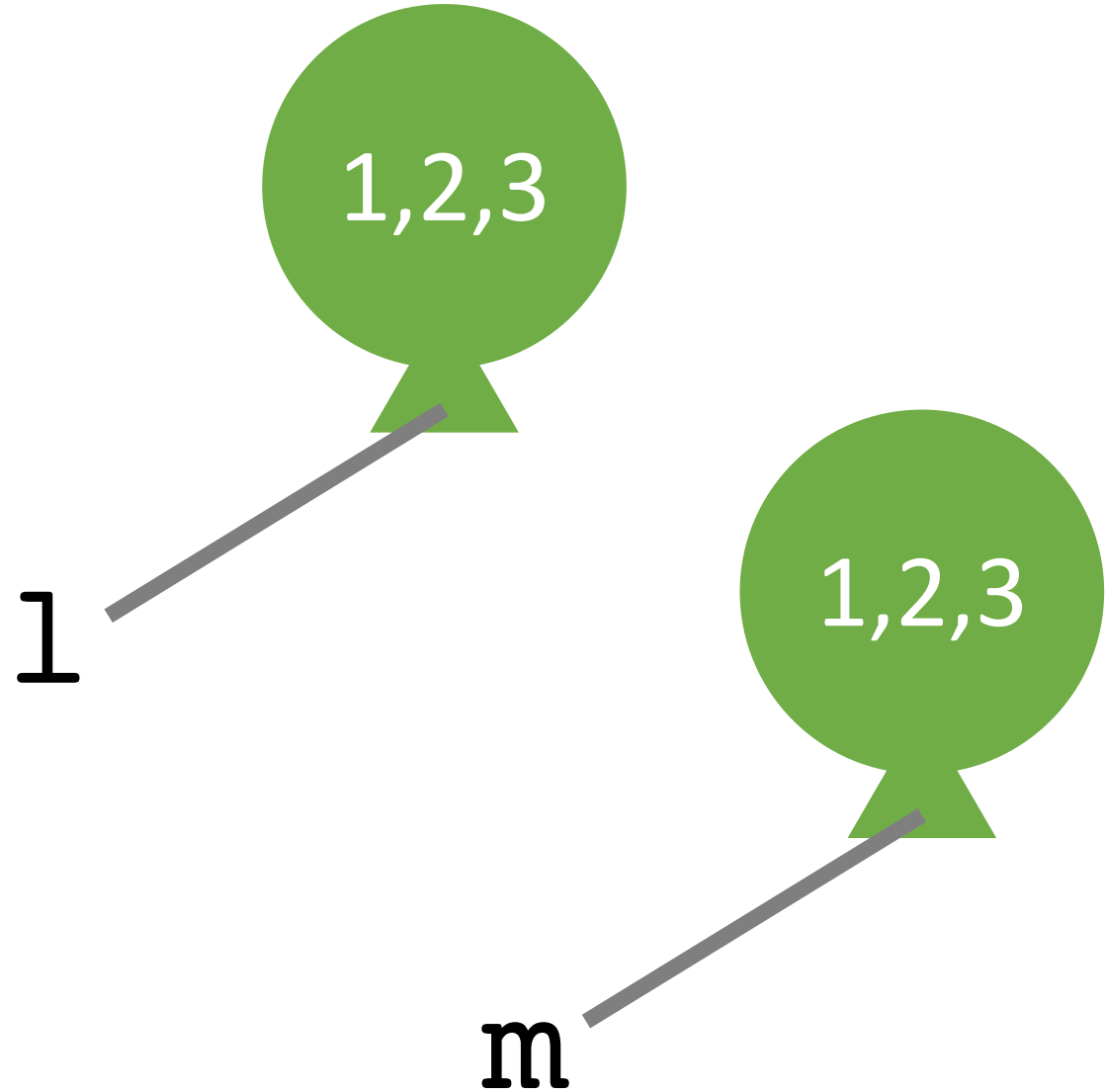
```
>>>x = 3
>>>y = 3
>>>x == y
    True
>>>x is y
    True
```

```
>>>l = [1,2,3]
>>>m = [1,2,3]
>>>l == m
    True
>>>l is m
    False
```

A Balloon Metaphor...

```
>>> l = [1,2,3]
```

```
>>> m = [1,2,3]
```



A Tale of Two Mutabilities...

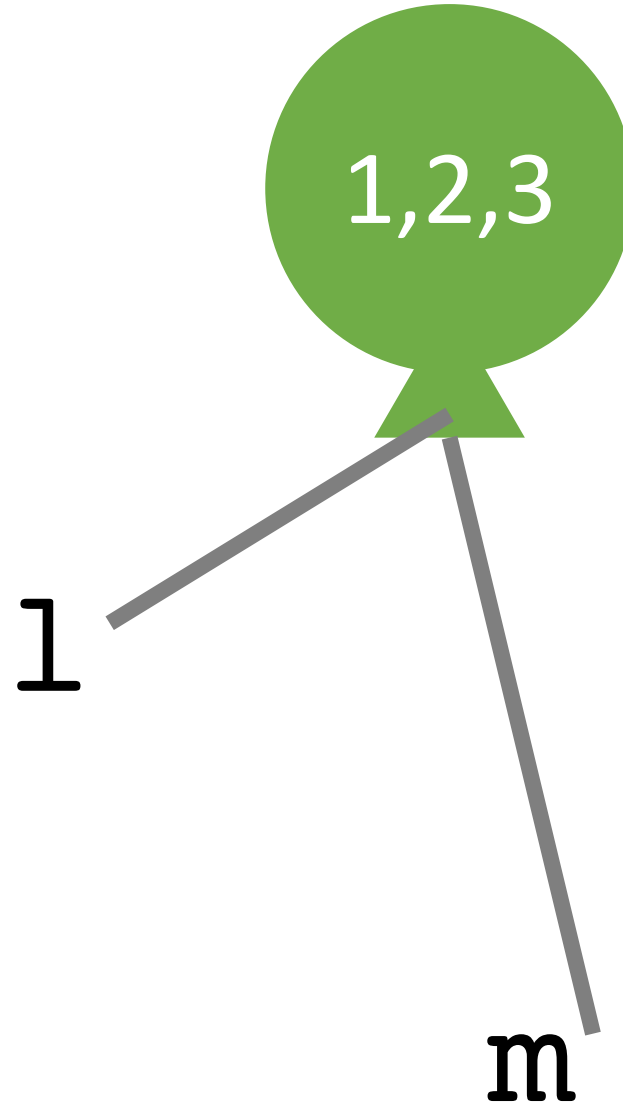
```
>>>x = 3
>>>y = x
>>>x == y
    ■ True
>>>x is y
    ■ True
```

```
>>>l = [1,2,3]
>>>m = l
>>>l == m
    ■ True
>>>l is m
    ■ True
```

A Balloon Metaphor...

```
>>> l = [1,2,3]
```

```
>>> m = l
```



A Tale of Two Sequences

Slice notation – copying a sequence

- `l = [18, 20, 5]`
- `m = l`

- `l == m`
 - `True`
- `l is m`
 - `True`

`l` and `m` are tied to the same balloon!

- `l = [18, 20, 5]`
- `n = [18, 20, 5]`

- `l == n`
 - `True`
- `l is n`
 - `False`

`l` and `n` are tied to different balloons!

- `l = [18, 20, 5]`
- `o = l[:]`

- `l == o`
 - `True`
- `l is o`
 - `False`

`l[:]` returns a *copy* of `l` (similar to `l` and `n`)

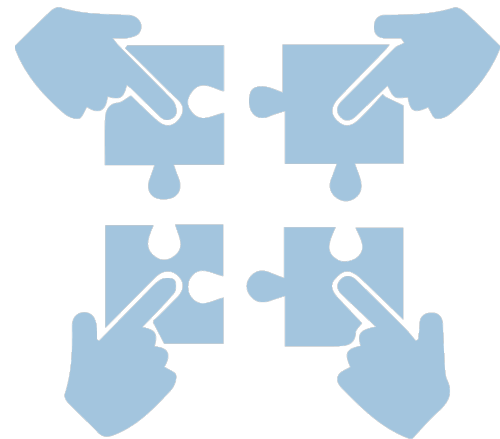
TODAY'S LESSON

Tuples

(like lists, but immutable)

POGIL – Activity 18: Tuples

- More data structures for sequences of objects
- Look at Python Activity 19, Questions 1-8
- Find a partner and talk through the questions together



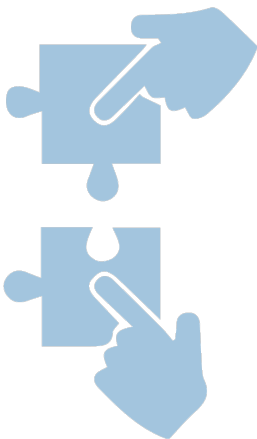
POGIL – Activity 19: Question 1

```
0 >>> mylist = ["pixel", "tally", 2]
1 >>> mytup = ("pixel", "tally", 2)
2 >>> mylist == mytup
3 False
```

- How many **elements** does the list named mylist contain? _____
- How many **elements** does the tuple named mytuple contain? _____
- What element is at mytuple[1]? _____ And at mytuple[2]? _____
- How do the elements of mylist and mytuple differ?

- Why does the comparison on line 2 return **False**?

- How does the syntax for defining a tuple differ from the syntax for a list?



tuples

- An immutable sequence of objects
- Declared with commas without square brackets
 - `>>> tup1 = ('hello', 'goodbye', 'goodmorning')`
 - `>>> tup1 = 'hello', 'goodbye', 'goodmorning'`
- Parentheses is the preferred notation (makes it clearly a tuple)
- Other than that, it looks pretty much like a list!

POGIL – Activity 19: Question 2

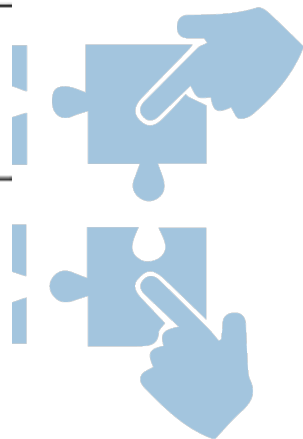
```
0 >>> mylist = ["pixel", "tally", 2]
1 >>> mytup = ("pixel", "tally", 2)
2 >>> mylist == mytup
3 False

4 >>> mytup1 = ("pixel", "tally", 2)
5 >>> mytup2 = "pixel", "tally", 2
6 >>> mytup1 == mytup2
7 True
```

- Write a line of code to access the last element of `mytup1` with indexing: _____
- How do the elements of `mytup1` and `mytup2` differ? _____
- How do lines 4 and line 5 differ?

- Write a line of code to create a new tuple using the parentheses notation style:

- Write some code that iterates through two tuples, `t1` and `t2`, and compares values at each index. It prints "Not Equal!" when it encounters two values that are different, and "Equal!" when the 2 values are equivalent:

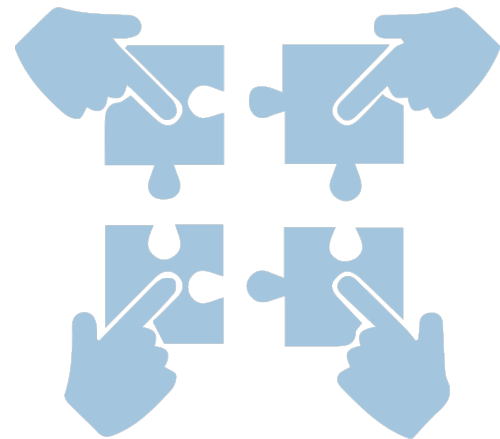


POGIL – Activity 19: Question 3

```
8 >>> mylist.append(42)
9 >>> mytup.append(42)
10 AttributeError: 'tuple' object has no attribute 'append'
```


- What is stored in mylist after line 8? _____
- What is stored in mytup after line 9?

- What might the AttributeError on line 10 mean?



POGIL – Activity 19: Question 4

```
0 >>> mytup = "pixel", "tally", 2
1 >>> mytup += 72,
2 >>> mytup
3 ('pixel', 'tally', 2, 72)
```

- a. How does what is stored in mytup at line 2 differ from what it contains at line 0?
-  _____
- b. What type of object is mytup? _____
- c. What type of object is 72, ? _____
- d. Rewrite 72, in its alternative format: _____
- e. Why does line 1 append an item to a tuple, while .append(obj) throws an error? _____

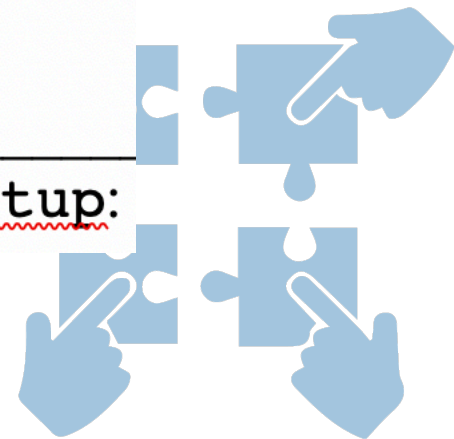


POGIL – Activity 19: Question 5

```
0 >>> mytup = "pixel", "tally", 2
1 >>> mytup += 72
2 TypeError: can only concatenate tuple (not 'int') to tuple
```

- What type of object is mytup? _____
- What type of object is 72? _____
- How should we modify line 1 to append 72 to our tuple?

- Write a line of code to append the string "second" to the tuple, mytup: _____

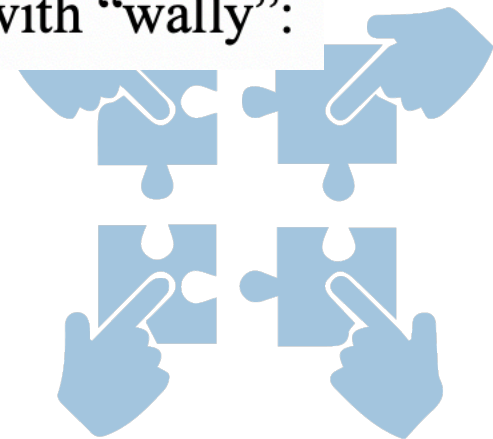


POGIL – Activity 19: Question 6

```
0 >>> mytup = ("pixel", "tally", 2)
1 >>> mytup[1] = "wally"
2 TypeError: 'tuple' object does not support item assignment
```

- a. What is the programmer trying to do on line 1?

- b. Write some lines of code to replace the second element of mytup with "wally":



A Tale of Two Mutabilities...

- `l = ['dog', 'cat', 'mouse', 'cheese']`
- `l[1] = 'dizzy'`

- `print(l)`
- `['dog', 'dizzy', 'mouse', 'cheese']`

- `t = ('dog', 'cat', 'mouse', 'cheese')`
- `t[1] = 'dizzy'`

- `TypeError: 'tuple' object does not support item assignment`

Lists are mutable (i.e., changeable)

Tuples are immutable.

A Tale of Two Mutabilities...

- `l = ['d', 'c', 'm']`
- `l.append('cheese')`

- `print(l)`
- `['d', 'c', 'm', 'cheese']`

- `t = ('d', 'c', 'm')`
- `t = t + ('cheese',)`

- `print(t)`
- `('d', 'c', 'm', 'cheese')`

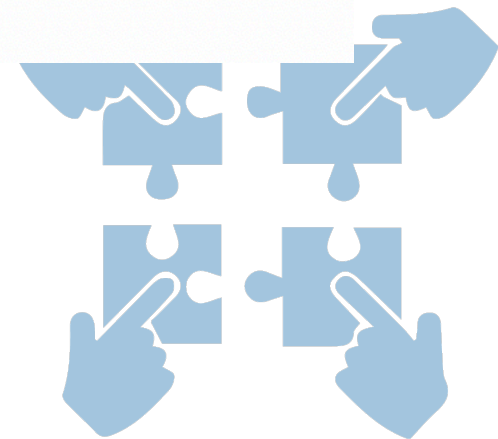
You can't modify tuples, but you can replace them!

POGIL – Activity 19: Question 7

```
0 >>> etup = ()  
1 >>> len(etup)
```

- a. How do we know that etup is a tuple?

- b. What will the output of line 1 be? _____
- c. Write some code that *iterates* through the list, mylist, and adds the items to a new tuple, mytup:
mylist = range(0,100)



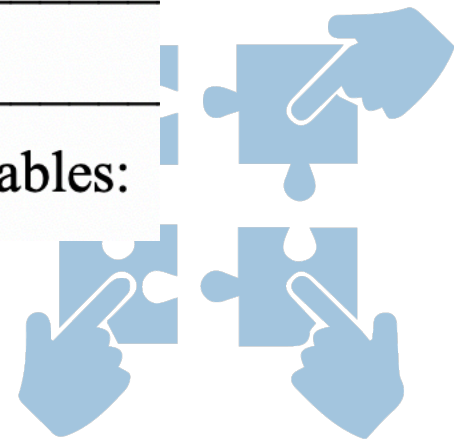
POGIL – Activity 19: Question 8

```
0 >>> a, b, c = 99, 77, 55
1 >>> a
2 99
3 >>> c
55
```

a. _____
What value is stored in the variable, b?

b. _____
Explain what is occurring on line 0:

c. _____
Write *one* line of code to assign 5 different values to 5 different variables:



POGIL – Activity 19: Question 9

- `s = ('cheese')`

- `type(s)`

- `<class 'str'>`

- `t = ('cheese',)`

- `type(t)`

- `<class 'tuple'>`

- `u = 'cheese',`

- `type(u)`

- `<class 'tuple'>`

Parentheses, no comma!

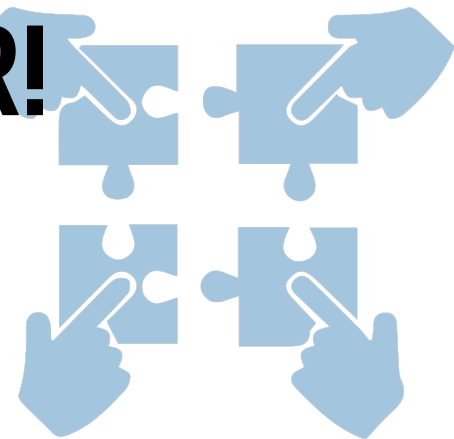
Comma, no parentheses!

“Declared with commas without square brackets”

**YOU SHOULD COMPLETE THE REST OF
ALL POGILS OUTSIDE OF CLASS.**

BEST DONE WITH A PARTNER OR STUDY GROUP.

CHECK YOUR ANSWERS ON A COMPUTER!



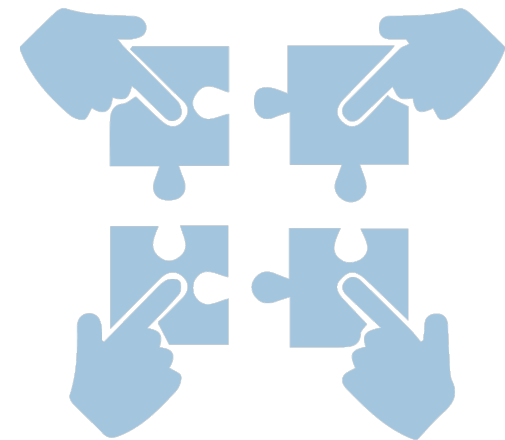
TODAY'S LESSON

Nested Loops

(Iterating over sequences of sequences)

POGIL – Activity 10: Nested Loops

- Using these in lab this week and next (and all semester)
- Look at Python Activity 10, Questions 1-2
 - Last page of your Tuple POGIL
- Find a partner and talk through the questions together

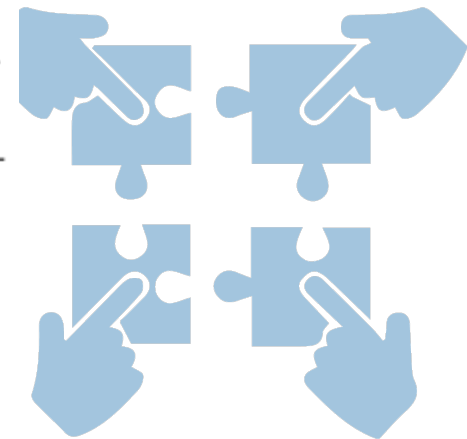


POGIL – Activity 10: Question 1

```
name = input("What is your name: ")
for x in range(5):
    for x in range(3):
        print(name + " ", end=" ")
    print()
```

- a. What does the program display?

- b. How many FOR loops are in this code? _____ Is one loop completely executed before the next loop begins? _____ What do you call this type of loop? _____
- c. How many times is the following line of code executed in the program? _____
`print(name + " ", end=" ")`
- d. Label the **inner loop** and the **outer loop**.
- e. What does the **inner loop** do? _____
- f. What does the **outer loop** do? _____



POGIL – Activity 10: Question 2

If you were asked to create a Python program that displayed the adjacent rectangle, you could easily do it with a set of print statements. You can also create it with a FOR loop and a print statement. This exercise will go through the steps to create a program that will print similar output but allows the user to determine the length and width of the figure when they execute the program.

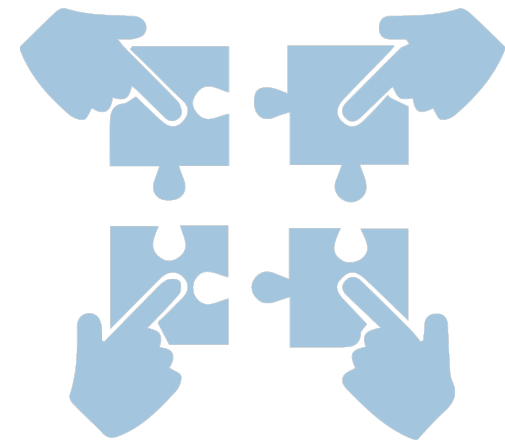
```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

- a. Create a code segment that prompts the user for a number between 1 and 10 and then prints that many asterisks (*) on one line. Use a FOR loop.

- b. You want the program to create several lines of asterisks. Extend the code in “a.” to also prompt the user for how many rows to print. Use an “outer” loop to print that many lines of asterisks. Write the revised code below.

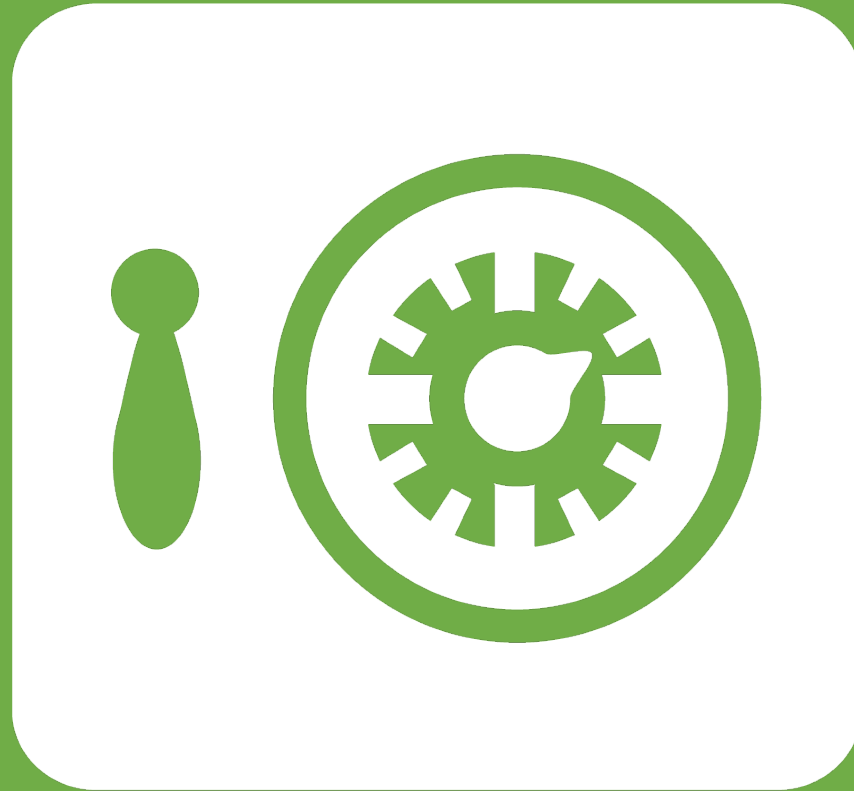
- c. Edit the program so that it prints numbers instead of asterisks. Write the line of code that was changed.

```
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
```



QUESTIONS?





Leftover Slides

F-strings

```
>>>name = 'Pixel'  
>>>age = 2  
>>>f"Hello, {name}. You are {age}."  
'Hello, Pixel. You are 2.'
```

This will print the same exact text:

```
>>>name = "Pixel"  
>>>age = 2  
>>>print("Hello, {}. You are{}".format(name, age))
```

A Tale of Two Sortings...

- `l = [18, 20, 5, 16]`

- `l.sort()`

- `l`

- `[5, 16, 18, 20]`

`.sort()` sorts the list itself

- `m = [18, 20, 5, 16]`

- `sorted(m)`

- `m`

- `[18, 20, 5, 16]`

`sorted()` returns a copy of the sorted list

Algorithms

How do we rotate a character by 1?

- $a \rightarrow b, b \rightarrow c, \dots, y \rightarrow z, z \rightarrow a, \text{ etc}$

Any ideas?

ord(c)

- `ord(' a ')`
 - 97
- `ord(' z ')`
 - 122
- `ord(' A ')`
 - 65
- `ord(' ; ')`
 - 59

and

chr(n)

- `chr(97)`
 - ' a '
- `chr(122)`
 - ' z '
- `chr(65)`
 - ' A '
- `chr(59)`
 - ' ; '

ASCII Values

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
64	40	100	@	96	60	140	`
65	41	101	A	97	61	141	a
66	42	102	B	98	62	142	b
67	43	103	C	99	63	143	c
68	44	104	D	100	64	144	d
69	45	105	E	101	65	145	e
70	46	106	F	102	66	146	f
71	47	107	G	103	67	147	g
72	48	110	H	104	68	150	h
73	49	111	I	105	69	151	i
74	4A	112	J	106	6A	152	j
75	4B	113	K	107	6B	153	k
76	4C	114	L	108	6C	154	l
77	4D	115	M	109	6D	155	m
78	4E	116	N	110	6E	156	n
79	4F	117	O	111	6F	157	o
80	50	120	P	112	70	160	p
81	51	121	Q	113	71	161	q
82	52	122	R	114	72	162	r
83	53	123	S	115	73	163	s
84	54	124	T	116	74	164	t
85	55	125	U	117	75	165	u
86	56	126	V	118	76	166	v
87	57	127	W	119	77	167	w
88	58	130	X	120	78	170	x
89	59	131	Y	121	79	171	y
90	5A	132	Z	122	7A	172	z
91	5B	133	[123	7B	173	{
92	5C	134	\	124	7C	174	
93	5D	135]	125	7D	175	}
94	5E	136	^	126	7E	176	~
95	5F	137	_	127	7F	177	

Rotating Letters

- `chr(ord('a') + 1)`
 - `'b'`
- `chr(ord('z') + 1)`
 - `'{'`

What happened here?

How do we wrap around back to 'a' after 'z'?

v	w	x	y	z	{
118	119	120	121	122	123

Rotating Letters

a	...	v	w	x	y	z	{
97	...	118	119	120	121	122	123

- What we want is for ASCII 123 to wrap around back to 97...
- Let's solve a simpler problem:

a	...	v	w	x	y	z	{
0	...	21	22	23	24	25	26

- Any number over 25 should wrap back around to 0, 1, 2, etc.
- What might we use to do that?

Rotating Numbers

a	b	...	x	y	z	{		}
0	1	...	23	24	25	26	27	28

- Fill in the blank:
 - For '{' → 'a': $\langle \text{character-ascii} \rangle \langle \text{operator} \rangle \langle \text{number} \rangle = \langle \text{rotated-ascii} \rangle$
 - $26 \langle \text{operator} \rangle \langle \text{number} \rangle = 0$
 - $26 \langle \text{operator} \rangle \langle \text{num-letters-in-alphabet} \rangle = 0$
 - $26 \underline{\quad} 26 = 0$
 - $26 \% 26 = 0$
 - For '|' → 'b': $27 \langle \text{operator} \rangle \langle \text{num-letters-in-alphabet} \rangle = 1$
 - $27 \% 26 = 1$
 - For '}' → 'c': $28 \% 26 = 2$

Rotating Letters

a	...	v	w	x	y	z	{
0	...	21	22	23	24	25	26

- Our formula is: $\langle \text{rotated-ascii} \rangle = \langle \text{character-ascii} \rangle \% 26$
- But that's for 0+, how do we convert to 97+?

a	...	v	w	x	y	z	{
97	...	118	119	120	121	122	123

- Subtract 97 to adjust to 'a' starting at 0:
 - $\langle \text{rotated-ascii} \rangle = (\langle \text{character-ascii} \rangle - 97) \% 26$
- And then add 97 back so we can convert to real ASCII
 - $\langle \text{rotated-ascii} \rangle = 97 + (\langle \text{character-ascii} \rangle - 97) \% 26$

Rotating Numbers

- $\langle \text{rotated-ascii} \rangle = 97 + (\langle \text{character-ascii} \rangle - 97) \% 26$
- Generalizes to:
 - $\text{rotAscii} = \text{ord}('a') + (\text{ord}(\text{givChar}) - \text{ord}('a')) \% 26$
- Convert from ASCII to character:
 - $\text{rotChar} = \text{chr}(\text{ord}('a') + (\text{ord}(\text{givChar}) - \text{ord}('a')) \% 26)$
- ...Are we missing anything?

Rotating Numbers

- We need to rotate by 'n':
 - $\text{rotChar} = \text{chr}(\text{ord}('a') + (((\text{ord}(\text{givChar}) - \text{ord}('a')) + n) \% 26))$
- ...Are we missing anything?
- What about...

A	B	...	y	z	[\
65	66	...	89	90	91	92
- $\text{rotChar} = \text{chr}(\text{ord}('A') + (((\text{ord}(\text{givChar}) - \text{ord}('A')) + n) \% 26))$

Rotating Letters

- `result = ''`
- `for c in s:`
 - `if c.islower():`
 - `c = chr(ord('a')+(((ord(c)-ord('a'))+n)%26))`
 - `elif c.isupper():`
 - `c = chr(ord('A')+(((ord(c)-ord('A'))+n)%26))`
 - `result += c`
- `return result`

See `crypt.py` in examples: `rot('xyz',1)`

Stdin

< for stdin

> for pushing output to a file

- `python3 crypt.py < crypt.py > whatever`

- Encrypts stdin into whatever file
- Rotates by 13 (by default, see program)

- `python3 crypt.py < whatever > second.py`

- Rotates 'whatever' by 13, stores in second.py
- What happens when you rotate by 13 twice?

| to run a second command

- `python3 crypt.py < crypt.py | python3 crypt.py`

- Rotates by 13, then rotates by 13 again