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

>>> x = 3
A Balloon Metaphor...

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

3

x
A Balloon Metaphor...

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

NUMBERS ARE IMMUTABLE
A Balloon Metaphor...

>>> [5,16,18]
A Balloon Metaphor...

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

mylist
A Balloon Metaphor...

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

Lists are mutable.
A Tale of Two Mutabilities...

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

```python
>>> l = [1,2,3]
>>> m = [1,2,3]
>>> l == m
  True
>>> l is m
  False
```
A Balloon Metaphor...

```python
>>> l = [1,2,3]
>>> m = [1,2,3]
```
A Tale of Two Mutabilities...

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

```python
>>> l = [1, 2, 3]
>>> m = l
>>> l == m
 True
>>> l is m
 True
```
A Balloon Metaphor...

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

1,2,3
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!

- \( l = [18, 20, 5] \)
- \( o = l[:] \)

- \( l == o \)
  - True
- \( l \) is \( o \)
  - False

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

- \( l = [18, 20, 5] \)
- \( n = [18, 20, 5] \)

- \( l == n \)
  - True
- \( l \) is \( n \)
  - False

l and n are tied to different balloons!
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 >>> `mlylist` = ["pixel", "tally", 2]
1 >>> `mytup` = ("pixel", "tally", 2)
2 >>> `mlylist` == `mytup`
3 False

a. How many **elements** does the list named `mlylist` contain? _______

b. How many **elements** does the tuple named `mytup` contain? _______


d. How do the elements of `mlylist` and `mytup` differ? 

__________________________________________________________

e. Why does the comparison on line 2 return False?

__________________________________________________________

f. 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

a. Write a line of code to access the last element of mytup1 with indexing: _____________

b. How do the elements of mytup1 and mytup2 differ? ___________________________

c. How do lines 4 and line 5 differ?

______________________________________________________________

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

______________________________________________________________

e. 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'

a. What is stored in mylist after line 8? ___________

b. What is stored in mytup after line 9? 

   _______________________________________________________________________

   c. 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 \texttt{mytup} at line 2 differ from what it contains at line 0?

b. What type of object is \texttt{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 \texttt{.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

a. What type of object is mytup?

b. What type of object is 72?

c. How should we modify line 1 to append 72 to our tuple?

d. 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...

• \( l = [\text{'dog'}, \text{'cat'}, \text{'mouse'}, \text{'cheese'}] \)
• \( l[1] = \text{'dizzy'} \)
• print(l)
• \([\text{'dog'}, \text{'dizzy'}, \text{'mouse'}, \text{'cheese'}] \)

• t= (\text{'dog'}, \text{'cat'}, \text{'mouse'}, \text{'cheese'})
• t[1] = \text{'dizzy'}

• TypeError: 'tuple' object does not support item assignment

Lists are mutable (i.e., changeable)
Tuples are immutable.
A Tale of Two Mutabilities...

```python
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

```python
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`:

```python
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

- \( s = (\text{‘cheese’}) \)
- `type(s)`
- `<class 'str'>`

- \( t = (\text{‘cheese’, } \) \)
- `type(t)`
- `<class 'tuple'>`

- \( u = \text{‘cheese’, } \)
- `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

```python
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? __________________________________________________________________________
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.
QUESTIONS?
Leftover Slides
F-strings

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

This will print the same exact text:

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

A Tale of Two Sortings...

- `l = [18, 20, 5, 16]`
- `l.sort()`

.**.sort() sorts the list itself**

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

- `m = [18, 20, 5, 16]`
- `sorted(m)`

.**sorted() returns a copy of the sorted list**

- `m`
  - `[18, 20, 5, 16]`
Algorithms
How do we rotate a character by 1?

- a → b, b → c, ..., y → z, z → a, etc

Any ideas?
ord(c) and chr(n)

- ord(‘a’) 97
- ord(‘z’) 122
- ord(‘A’) 65
- ord(‘;’) 59
- chr(97) ‘a’
- chr(122) ‘z’
- chr(65) ‘A’
- chr(59) ‘;’
# ASCII Values

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>40</td>
<td>100</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>140</td>
<td>`</td>
</tr>
<tr>
<td>65</td>
<td>41</td>
<td>101</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>141</td>
<td>a</td>
</tr>
<tr>
<td>66</td>
<td>42</td>
<td>102</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>142</td>
<td>b</td>
</tr>
<tr>
<td>67</td>
<td>43</td>
<td>103</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>143</td>
<td>c</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>104</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>144</td>
<td>d</td>
</tr>
<tr>
<td>69</td>
<td>45</td>
<td>105</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>145</td>
<td>e</td>
</tr>
<tr>
<td>70</td>
<td>46</td>
<td>106</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>146</td>
<td>f</td>
</tr>
<tr>
<td>71</td>
<td>47</td>
<td>107</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>147</td>
<td>g</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>108</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>150</td>
<td>h</td>
</tr>
<tr>
<td>73</td>
<td>49</td>
<td>111</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>151</td>
<td>i</td>
</tr>
<tr>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>152</td>
<td>j</td>
</tr>
<tr>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>153</td>
<td>k</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>154</td>
<td>l</td>
</tr>
<tr>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>155</td>
<td>m</td>
</tr>
<tr>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>156</td>
<td>n</td>
</tr>
<tr>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>157</td>
<td>o</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>120</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>160</td>
<td>p</td>
</tr>
<tr>
<td>81</td>
<td>51</td>
<td>121</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>161</td>
<td>q</td>
</tr>
<tr>
<td>82</td>
<td>52</td>
<td>122</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>162</td>
<td>r</td>
</tr>
<tr>
<td>83</td>
<td>53</td>
<td>123</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>163</td>
<td>s</td>
</tr>
<tr>
<td>84</td>
<td>54</td>
<td>124</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>164</td>
<td>t</td>
</tr>
<tr>
<td>85</td>
<td>55</td>
<td>125</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>165</td>
<td>u</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>a</th>
<th>...</th>
<th>v</th>
<th>w</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>{</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>...</td>
<td>118</td>
<td>119</td>
<td>120</td>
<td>121</td>
<td>122</td>
<td>123</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>a</th>
<th>...</th>
<th>v</th>
<th>w</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>{</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>...</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
</tbody>
</table>

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

- Fill in the blank:
  - For ‘{’ → ‘a’:  
    - `<character-ascii> <operator> <number> = <rotated-ascii>`
      - 26 `<operator> <number> = 0`
      - 26 `<operator> <num-letters-in-alphabet> = 0`
      - 26 ___ 26 = 0
      - 26 % 26 = 0
  - For ‘|’ → ‘b’:  
    - 27 `<operator> <num-letters-in-alphabet> = 1`
      - 27 % 26 = 1
  - For ‘}’ → ‘c’:  
    - 28 % 26 = 2
Rotating Letters

\[
\begin{array}{ccccccc}
\text{a} & \ldots & \text{v} & \text{w} & \text{x} & \text{y} & \text{z} \\
0 & \ldots & 21 & 22 & 23 & 24 & 25 & 26
\end{array}
\]

• Our formula is: \(<\text{rotated-ascii}> = \langle\text{character-ascii}\rangle \% 26\>

• But that’s for 0+, how do we convert to 97+?

\[
\begin{array}{ccccccc}
\text{a} & \ldots & \text{v} & \text{w} & \text{x} & \text{y} & \text{z} \\
97 & \ldots & 118 & 119 & 120 & 121 & 122 & 123
\end{array}
\]

• Subtract 97 to adjust to ‘a’ starting at 0:
  • \(<\text{rotated-ascii}> = (\langle\text{character-ascii}\rangle-97) \% 26\>

• And then add 97 back so we can convert to real ASCII
  • \(<\text{rotated-ascii}> = 97 + (\langle\text{character-ascii}\rangle-97) \% 26\>
Rotating Numbers

• \(<\text{rotated-ascii}> = 97 + (\text{<character-ascii>}-97) \mod 26\)
• Generalizes to:
  ▪ \(\text{rotAscii} = \text{ord}(\text{`a'}) + (\text{ord(givChar)}-\text{ord(`a')}) \mod 26\)
• Convert from ASCII to character:
  ▪ \(\text{rotChar} = \text{chr}(\text{ord(`a') + (ord(givChar)-ord(`a')}) \mod 26\)
• ...Are we missing anything?
Rotating Numbers

• We need to rotate by ‘n’:
  - \( \text{rotChar} = \text{chr}(\text{ord}('a') + (((\text{ord}('givChar') - \text{ord}('a')) + n) \mod 26)) \)

• ...Are we missing anything?

• What about...

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>...</td>
<td>y</td>
<td>z</td>
<td>[</td>
<td>\</td>
</tr>
<tr>
<td>65</td>
<td>66</td>
<td>...</td>
<td>89</td>
<td>90</td>
<td>91</td>
<td>92</td>
</tr>
</tbody>
</table>

• \( \text{rotChar} = \text{chr}(\text{ord}('A') + (((\text{ord}('givChar') - \text{ord}('A')) + n) \mod 26)) \)
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