Objectives

- String Formatting
- Data Representations, continued

1

Review

- What is the special name for sequences, like newlines, tabs, …?
  - How do we represent them in strings?
- How does the computer represent data (e.g., numbers and text)?
- What are the various things we can do with strings?

2

FORMATTING STRINGS

Formatting Strings: `format` Method

- How to use:
  - `"templatestring".format(<replacementvalues>)`
- Semantics: returns a **formatted string**
  - Means "format the `templatestring`, using the format(s) specified by **format specifiers** on the corresponding replacement values"
  - Returned as the `str` data type
- Typically used with print statements

Formatting Strings: format Method

- How to use:
  - `"templatestring".format(<replacementvalues>)`
- `templatestring` allows us to control how output is displayed to user
  - Right, left justification
  - Number of decimals to display

5

Formatting Strings: format Method

- **`templatestring`** is a template for the resulting string with *format specifiers* instead of the values
  - For each format specifier in templatestring, need a corresponding **`replacement value`**
  - Example:

```
"{:.2f}".format(3.14159)
```

Evaluates to `"3.14"`

One format specifier in template string

Corresponding replacement value

- Throws **`IndexError`** if not enough replacements for specifiers in `templatestring`

6

Format Specifiers

[ ] mean optional

- General format:
  `{[field_name]:conversion}`

index number of the argument,
i.e., which field in the template string

- `conversion`
  - conversion code of the data type

Code	Type
s	string
d	integer
f	float
e	floating point with exponent

Default if code isn't given

(There are more…)

Format Specifiers

[ ] mean optional

Conversion options `:[flags][width][.precision][code]`

- flags:

Flag	Meaning
<	Left-justification • Default for strings
>	Right-justification • Default for numbers
^	Centered
0	Zero fills
+	Adds a + sign before positive values

- width:
  - *Minimum* number of character spaces reserved to display the entire value
  - Includes decimal point, digits before and after the decimal point and the sign
- precision:
  - Number of digits after the decimal point for **floating point** values

Example using Format Operator

Format specifier

```
print("Your item that cost ${:.2f}".format(value)
print("costs ${:.2f} with tax".format(tax))
```

Alternative:

```
print(
 "Your item that cost ${:.2f} costs ${:.2f} with tax".format(value, tax))
```

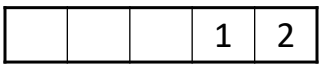
sales_tax2.py

9

Example Format Specifiers

```
"{:5d}".format(12)
```

→ " 12"



Field width is 5

```
"{:9.2f}".format(23.1999)
```

→ " 23.20"

Precision is 2

Right-justified

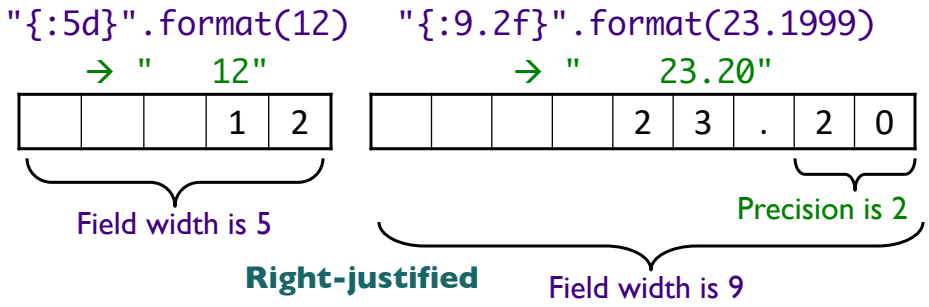
Field width is 9

- What if precision is bigger than the decimal places?
- What if field width is smaller than the length of the value?

Any guesses? Try out in interpreter.

10

Example Format Specifiers



- What if precision is bigger than the decimal places?
  - Fills decimal with 0s
- What if field width is smaller than the length of the value?
  - String contains entire value

Formatting Practice

- `x = 10`
- `y = 3.5`
- `z = "apple"`
- `"{:6d}".format(x)`
- `"{:6.2f}".format(x)`
- `"{:6.2f}".format(y)`
- `"{:06.2f}".format(y)`
- `"{:^11s}".format(z)`
- `"{:5d} {:<7.3f}".format(x,y)`

Example: Printing Out Tables

- A table of temperature conversions

```
Temp F      Temp C      Temp K
------      ------      ------
-459.7      -273.1         0.0
   0.0       -17.8       255.2
  32.0         0.0       273.1
```

- If we want to print data in rows, what is the template for what a row looks like?
  - How do we make the column labels line up?

String Formatting

- There are a lot more things you can do with string formatting
- Presenting just a subset of the most commonly used functionality
- When formatting strings, consider
  - What is the data type of your data?
    - If a float, how many decimal places do you want?
  - How wide do you want the data to be?
  - What justification? Zero fill? Other flags?

String Representations

- A **string** is a *sequence* of characters
- Each character is stored as a binary number
- **ASCII** (American Standard Code for Information Interchange) is one standard encoding for characters
  - Limitation: ASCII is based on the English language
  - Cannot represent other types of characters
  - Handout is just a subset
- Unicode is a new standard

Translating to/from ASCII

- Translate a character into its ASCII numeric code using **built-in function** `ord`
  - `ord('a') ==> 97`
- Translate an ASCII numeric code into its character using **built-in function** `chr`
  - `chr(97) ==> 'a'`

`ascii_table.py`
`ascii.py`

ASCII Questions

- Lowercase letters are represented by what range of numbers?
- Uppercase letters are represented by what range of numbers?
- What is the difference between the decimal encoding of 'M' and 'N'?
  - Between 'm' and 'n'?
- Explain the result of `"Zebra" < "aardvarks"` being `True`

17

ASCII Questions

- Lowercase letters are represented by what range of numbers?
  - 97—122
- Uppercase letters are represented by what range of numbers?
  - 65—90
- What is the difference between the decimal encoding of 'M' and 'N'?
  - Between 'm' and 'n'?
  - 1
- Explain the result of `"Zebra" < "aardvarks"` being `True`
  - `ord("Z") < ord("a")`

18

Translating to/from ASCII

- Translate a character into its ASCII numeric code using **built-in function** `ord`
  - `ord('a') ==> 97`
- Translate an ASCII numeric code into its character using **built-in function** `chr`
  - `chr(97) ==> 'a'`

`ascii_table.py`
`ascii.py`

Encryption

- Process of encoding information to keep it secure
- One technique: Substitution Cipher
  - Each character in message is replaced by a new character

Caesar Cipher

- Replace character with a character X places away
  - X is called the ***key***
- Julius Caesar used technique to communicate with his generals
- "Wrap around" within the lowercase letters
- Write program(s) to do this in next lab

Caesar Cipher

- Using the ASCII handout, what would be the encoded messages?

Message	Key	Encoded Message
apple	5	
zebra	5	
the eagle flies at midnight	-5	

Caesar Cipher

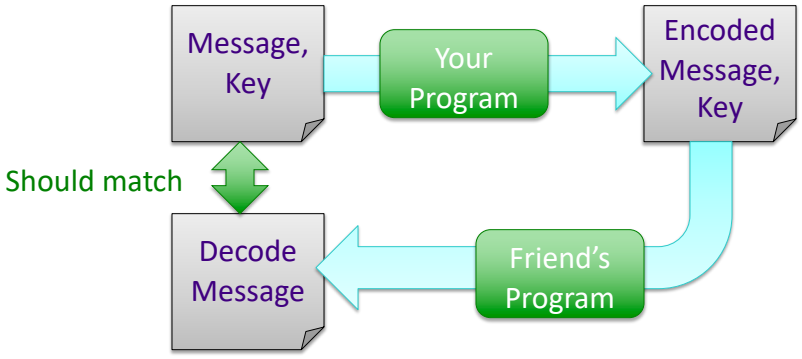
Message	Key	Encoded Message
apple	5	fuuqj
zebra	5	ejgwf
the eagle flies at midnight	-5	ocz zvbgz agdzn vo hdyidbco

What is your algorithm for the encoding process? How would you *decode* an encrypted message?

23

Next Lab

- Write an encoding/decoding program
  - Encode a message
  - Give to a friend to decode



24

Caesar Cipher: `translateLetter`

- Given a letter and key
- Convert the character to its ASCII value
- Add the key to that value
- Make sure that the new value is a "valid" ASCII value, i.e., that that new value is in the range of lowercase letter ASCII values
  - If not, "wrap around" to adjust that value so that it's in the valid range
- Convert the ASCII value into a character
- Return the letter

Caesar Cipher (Partial) Algorithm

- Given a message and key
- For each character in the message
  - Check if the character is a space; if it is, it stays a space
  - Otherwise
    - Translate Letter
- Return the message

Looking Ahead

- Lab 7 prep
  - Assignment: Repeat the section on string methods, which includes the subsection on format method
  - Think about how to implement the Caesar Cipher
- Lab 7
- Broader Issue: Cryptocurrency