

Objectives

- String Formatting
- Data Representations, continued

1

Review

- How do you call a method?

2

FORMATTING STRINGS

Mar 6, 2023

Sprenkle - CSC1111

3

3

Solution: format Method

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

Mar 6, 2023

Sprenkle - CSC1111

4

4

Solution: format Method

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

Mar 6, 2023

Sprenkle - CSC1111

5

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, should have a *replacement value*
- `"{: .2f}".format(3.14159)` Evaluates to "3.14"
- ↑ ↑
One format specifier Corresponding replacement value
in template string
- Throws **IndexError** if not enough replacements for specifiers in templatestring

Mar 6, 2023

Sprenkle - CSC1111

6

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

Mar 6, 2023

Sprenkle - CSC1111

7

7

Format Specifiers

[] mean optional

Conversion options

: [flags] [width] [.precision] [code]

- flags:

Flag	Meaning
0	Zero fill to width
+	Adds a + sign before positive values
<	Left justify (default for strings)
>	Right justify (default for numbers)
^	center

- 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

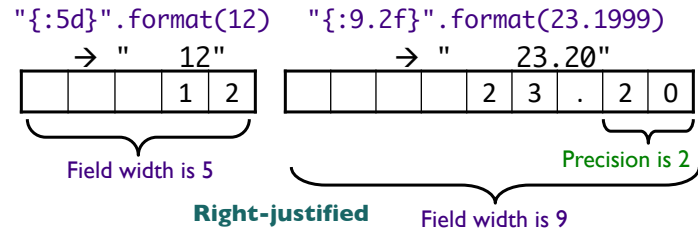
Mar 6, 2023

Sprenkle - CSC1111

8

8

Example Format Specifiers



- 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.

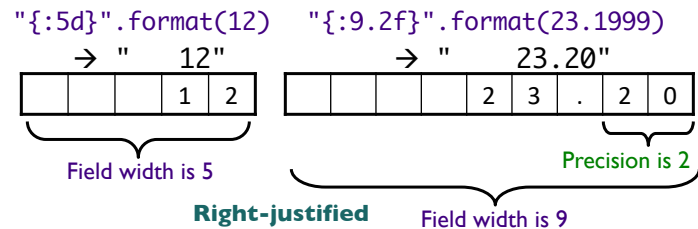
Mar 6, 2023

Sprenkle - CSC1111

9

9

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

Mar 6, 2023

Sprenkle - CSC1111

10

10

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)`

What is the resulting string?

Mar 6, 2023

Sprenkle - CSCI111

11

11

String Formatting

- There is a lot more you can do with string formatting
 - This is 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?
- The answer to these questions help guide your creation of format specifiers

Mar 6, 2023

Sprenkle - CSCI111

12

12

Using format Method in print

- You often want to format data within a broader context.
- Example: printing out money values
 - How do you want that data formatted?

Mar 6, 2023

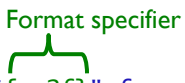
Sprenkle - CSC1111

13

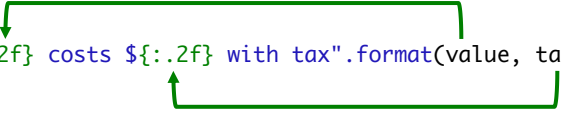
13

Using format Method in print

- Printing money values


`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))`


Mar 6, 2023

Sprenkle - CSC1111

14

14

Using format Method in print

- Printing money values

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))
```

How is this different from using the round function?

15

15

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?
 - For above, not as simple as using tabs. Why not?

Mar 6, 2023

Sprenkle - CSC1111

temp_table.py

16

16

String Formatting Conclusion

- There is a lot more you can do with string formatting
 - This is a subset of the most commonly used functionality
- When formatting strings, create the format specifiers by asking:
 - What is the data's type?
 - How do I want it to look?

Mar 6, 2023

Sprenkle - CSCI111

17

17

Review

- What is the special name for one way that computers encode strings?
 - How can we convert from characters to their numerical representation?
 - How can we convert from the numerical representation to the character?
- How does the Caesar Cipher work?

Mar 6, 2023

Sprenkle - CSCI111

18

18

Review: 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'

Mar 6, 2023

Sprenkle - CSC1111

[ascii_table.py](#)
[ascii.py](#)

19

19

Review: 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

Original Letter	Key	Encrypted Letter
'a'	1	'b'
'b'	1	'c'
'z'	1	'a'

Mar 6, 2023

Sprenkle - CSC1111

20

20

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	

Mar 6, 2023

Sprenkle - CSC1111

21

21

Caesar Cipher

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

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

Mar 6, 2023

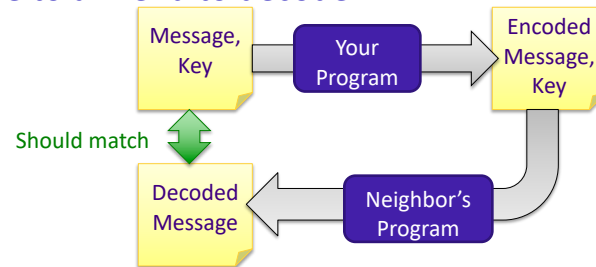
Sprenkle - CSC1111

22

22

Next Lab

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



Mar 6, 2023

Sprenkle - CSCI111

23

23

Caesar Cipher: encryptLetter

- 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 encrypted letter

Mar 6, 2023

Sprenkle - CSCI111

24

24

Caesar Cipher (Partial) Algorithm

- Given a message and key
- For each character in the message
 - Check if the character is a space or punctuation
 - if it is, it stays that character
 - Otherwise
 - encrypt letter
- Return the message

Mar 6, 2023

Sprenkle - CSCI111

25

25

Looking Ahead

- Lab 7 prep
 - Repeat sections on simple tables (with escape characters), string methods (which includes the subsection on format method), and character classifications
 - Think about how to implement the Caesar Cipher
- Lab 7

Mar 6, 2023

Sprenkle - CSCI111

26

26