

## Objectives

- Exceptions
- Our own modules
- Writing to files

March 11, 2019

Sprenkle - CSCI111

1

## Check: Parts of an Algorithm

- Primitive operations
  - What data you have, what you can do to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques


- Which of these have we covered?
- How do we implement them in Python?

March 11, 2019

Sprenkle - CSCI111

2

## Midway Check: Parts of an Algorithm

- Primitive operations 

where most of the rest of the semester focuses

  - What **data** you have, what you **can do** to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques

Working toward  
no-longer-*primitive*

March 11, 2019

Sprenkle - CSCI111

3

## EXCEPTION HANDLING

March 11, 2019

Sprenkle - CSCI111


4

## Handling Exceptions

- Using try/except statements

- Syntax:

```
try:
    <body>
except [<errorType>] :
    <handler>
```



Optional: use this to handle specific error types appropriately

- Example:

```
try:
    age = int(input("Enter your age: "))
    currentyear = int(input("Enter the current year: "))
except:
    print("ERROR: Your input was not in the correct form.")
    print("Enter integers for your age and the current year")
    sys.exit(1)
```

March 11, 2019

Sprenkle - CSCI111

[yearborn.py](#)

5

## Discussion: `sys.exit()`

- What is `sys.exit()`? Where does it come from?

March 11, 2019

Sprenkle - CSCI111

6

## Discussion: `sys.exit()`

- What is `sys.exit()`? Where does it come from?

- `import sys`

- Imports the `sys` module

```
exit(...)  
exit([status])
```

Exit the interpreter by raising `SystemExit(status)`.  
If the status is omitted or `None`, it defaults to zero (i.e., success).  
If the status is an integer, it will be used as the system exit status.  
If it is another kind of object, it will be printed and the system exit status will be one (i.e., failure).

March 11, 2019

Sprenkle - CSCI111

7

## Handling Exceptions

- Other types of exceptions

- File exceptions:

- File doesn't exist
  - Don't have permission to read/write file

March 11, 2019

Sprenkle - CSCI111

`file_handle.py`

8


## CREATING MODULES

March 11, 2019

Sprenkle - CSCI111

9

### Where are Functions Defined?

- Functions can go inside of program script
  - Defined before use/called (if no `main()` function)
  - Or, below the `main()` function (*preferred*)
- Functions can go inside a separate **module** 

March 11, 2019

Sprenkle - CSCI111

10

## Creating Modules

- Modules group together related functions and constants
- Unlike functions, no special keyword to define a module
  - A module is named by its filename
- You've used modules in the past
  - `graphics.py`
  - `game.py`

Just a  
Python file!

March 11, 2019

Sprenkle - CSCI111

11

## Typical Use of Modules

- Put your reusable code in a module that can be shared with others
- Example: `game.py`
  - `rollDie(sides)`
  - `rollMultipleDice(numDice, sides)`
- Call `import game` in Python interpreter
  - What happened?

March 11, 2019

Sprenkle - CSCI111

12

## Creating Modules

- Then, to call **rollDie** function
  - `game.rollDie(6)`
- To access a defined constants
  - Example: `game.SIDES`

March 11, 2019

Sprenkle - CSCI111

13

## Creating Modules

- So that our program doesn't execute code automatically when it is **imported** in a program, at bottom, add

```
if __name__ == '__main__':  
    testRollDie()  
    testRollMultipleDice()
```

Not important how this works;  
just know when to use

- Note the sub-directories now listed in the directory

March 11, 2019

Sprenkle - CSCI111

14

## Benefits of Defining Functions in Separate Module

- Reduces code in **primary** driver script
- Easier to reuse by importing from a module
- Maintains the “black box”
  - **Abstraction**
- Isolates testing of function
- Write “test driver” scripts to test functions separately from use in script

March 11, 2019

Sprenkle - CSCI111

15

## Problem: Cleaning Up Data

- **Given:** a CSV file containing students’ names and their class according to the Registrar
- **Problem:** This file has TMI
  - Just want the last name and the class year
  - Instead of Ugr:Sophomore, say “Sophomore”
- **Solution:**
  - Read through file “data/years.csv”, clean up data
  - Write the cleaned up data to a new file called “data/roster.txt”
    - 1<sup>st</sup> iteration: lastname class
    - 2<sup>nd</sup> iteration: nice tables of data

`cleanRoster.py`

March 11, 2019

Sprenkle - CSCI111

16



## Looking Ahead

- Lab 8 tomorrow – lists! files!
  - Pre Lab 8
- Broader Issue – Net Neutrality