On your way in…(on the side table)

Pick-up:

1. Homework 01 print-out

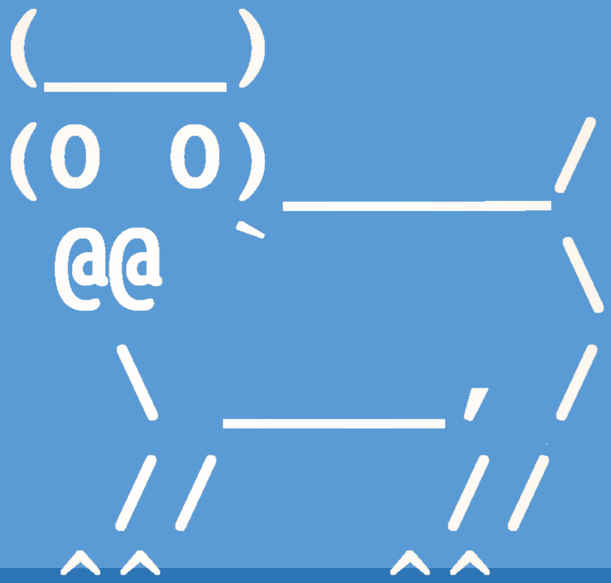
2. POGIL Activity #12
3. POGIL Activity #13
4. Day of the Week Algorithm print-out

Welcome to CS 134!

Introduction to Computer Science

Iris Howley

-Functions-



Spring 2020

Housekeeping

Homework 01

- Due Monday, February 17 (in less than a week), **in class**

- Some open-ended responses to get you to think about why we do some of the things we do, in programming
- A little bit of code reading
- A little bit of code writing

Labs are due Thursday and Friday (at noon)

- If you have Monday lab:
  - → `push` your work by Wednesday at 11pm!

- If you have Tuesday lab:
  - → `push` your work by Thursday at 11pm!

For every lab!

(unless stated otherwise)

Note:

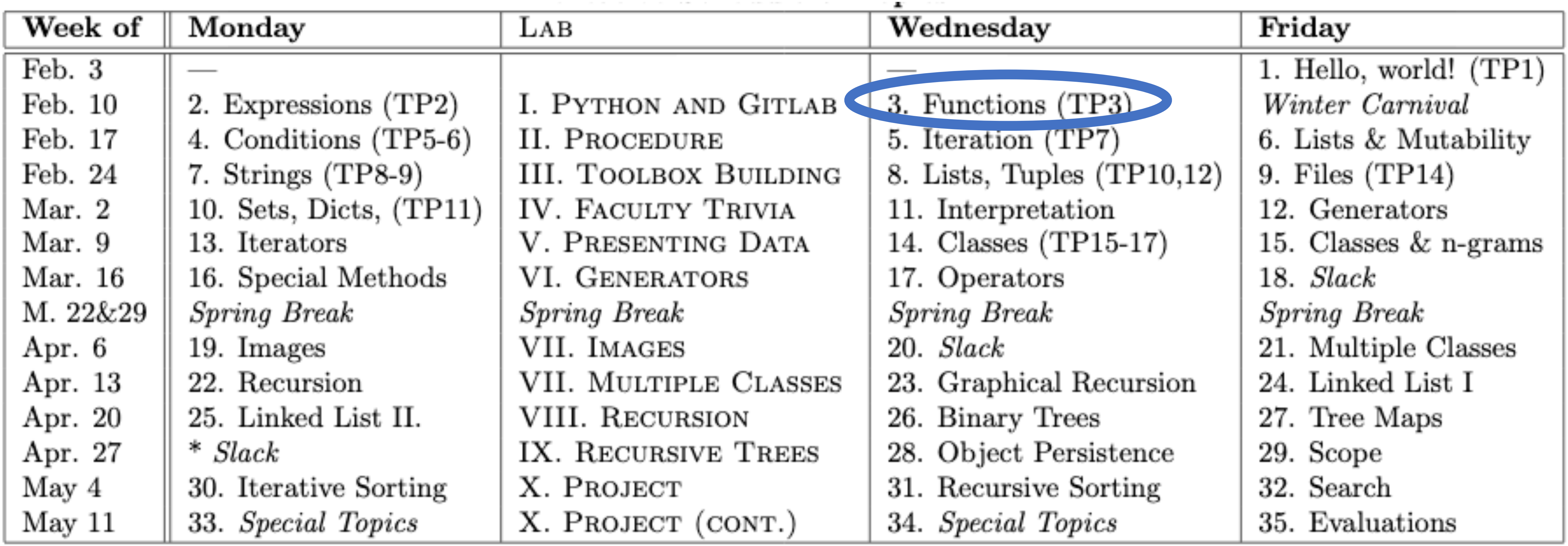
Homeworks that you turn in are marked as “Homework”

POGIL activities are in-class, optional activities that are not turned in.

(but they’re meant to assist in your learning)

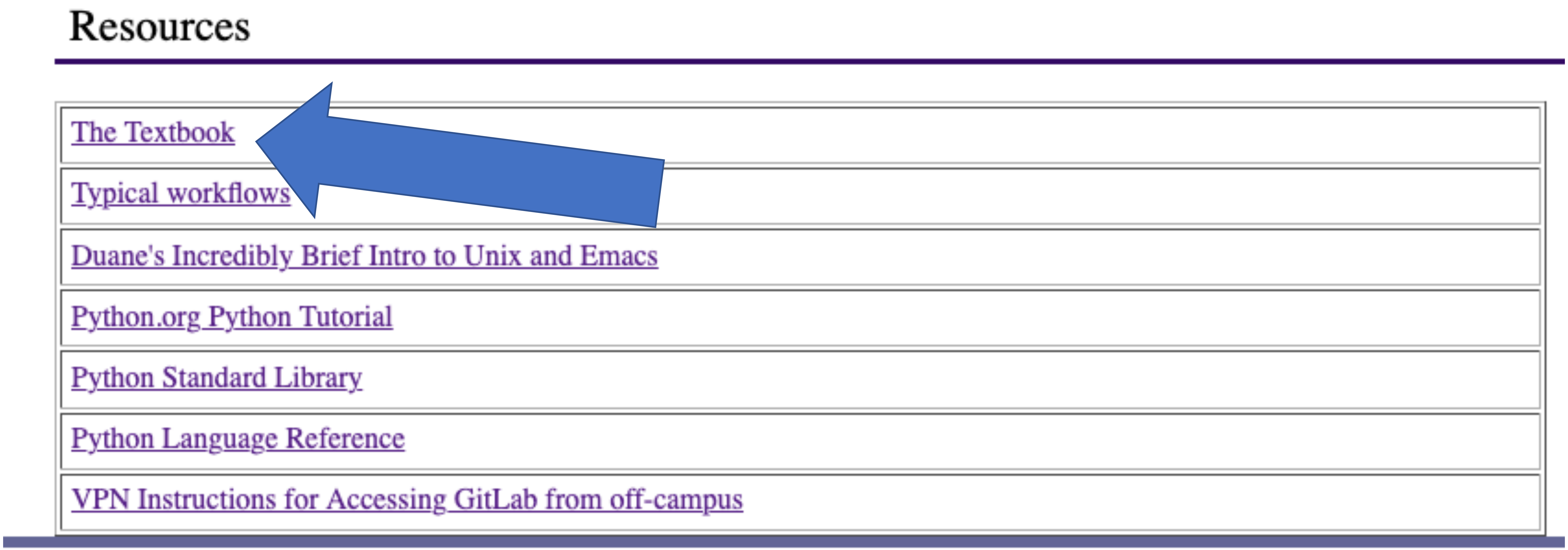
(if you are struggling with concepts in the POGIL activity, you’ll encounter the same struggles in other parts of this course)

Have you been following along in the textbook?



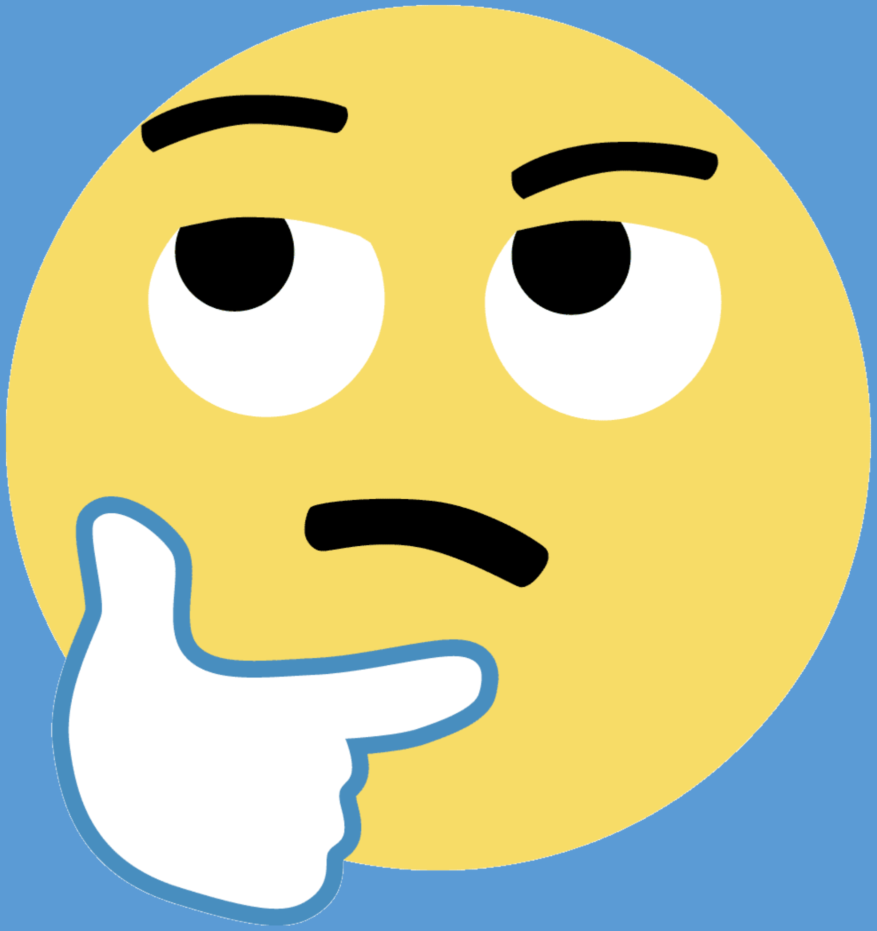
Week of	Monday	Lab	Wednesday	Friday
Feb. 3	—		—	1. Hello, world! (TP1)
Feb. 10	2. Expressions (TP2)	I. Python and Gitlab	3. Functions (TP3)	*Winter Carnival*
Feb. 17	4. Conditions (TP5-6)	II. Procedure	5. Iteration (TP7)	6. Lists & Mutability
Feb. 24	7. Strings (TP8-9)	III. Toolbox Building	8. Lists, Tuples (TP10,12)	9. Files (TP14)
Mar. 2	10. Sets, Dicts, (TP11)	IV. Faculty Trivia	11. Interpretation	12. Generators
Mar. 9	13. Iterators	V. Presenting Data	14. Classes (TP15-17)	15. Classes & n-grams
Mar. 16	16. Special Methods	VI. Generators	17. Operators	18. *Slack*
M. 22&29	*Spring Break*	*Spring Break*	*Spring Break*	*Spring Break*
Apr. 6	19. Images	VII. Images	20. *Slack*	21. Multiple Classes
Apr. 13	22. Recursion	VII. Multiple Classes	23. Graphical Recursion	24. Linked List I
Apr. 20	25. Linked List II.	VIII. Recursion	26. Binary Trees	27. Tree Maps
Apr. 27	* *Slack*	IX. Recursive Trees	28. Object Persistence	29. Scope
May 4	30. Iterative Sorting	X. Project	31. Recursive Sorting	32. Search
May 11	33. *Special Topics*	X. Project (cont.)	34. *Special Topics*	35. Evaluations

Have you been following along in the textbook?



Resources

- The Textbook
- Typical workflows
- Duane's Incredibly Brief Intro to Unix and Emacs
- Python.org Python Tutorial
- Python Standard Library
- Python Language Reference
- VPN Instructions for Accessing GitLab from off-campus



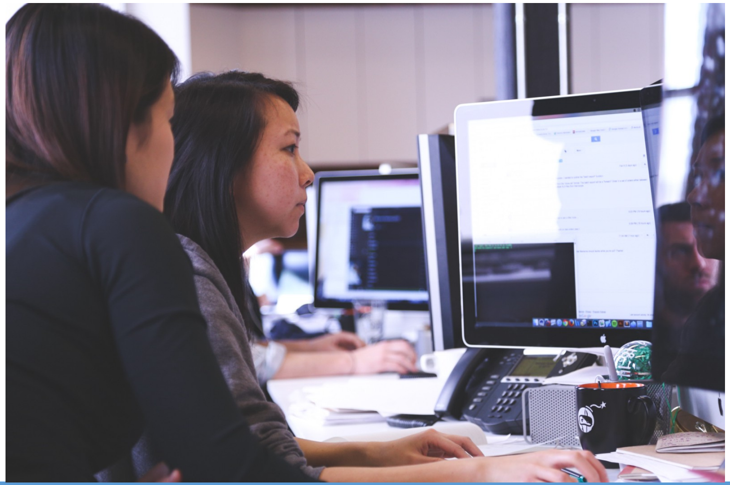
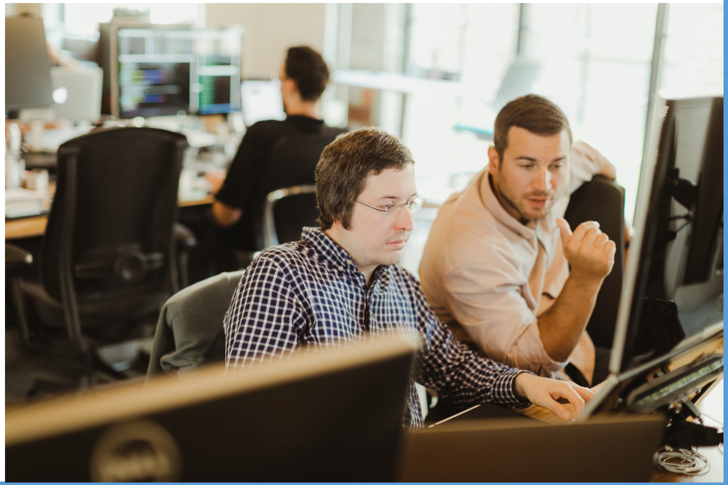
A Thought.

IT IS OKAY TO MAKE MISTAKES.
THIS IS HOW WE LEARN.

IT IS OKAY FOR ME TO MAKE MISTAKES.
I WILL MAKE A LOT OF MISTAKES.

The longer the program, the more errors!
Even for experts!

YOU ARE MY PAIR PROGRAMMING PARTNERS.

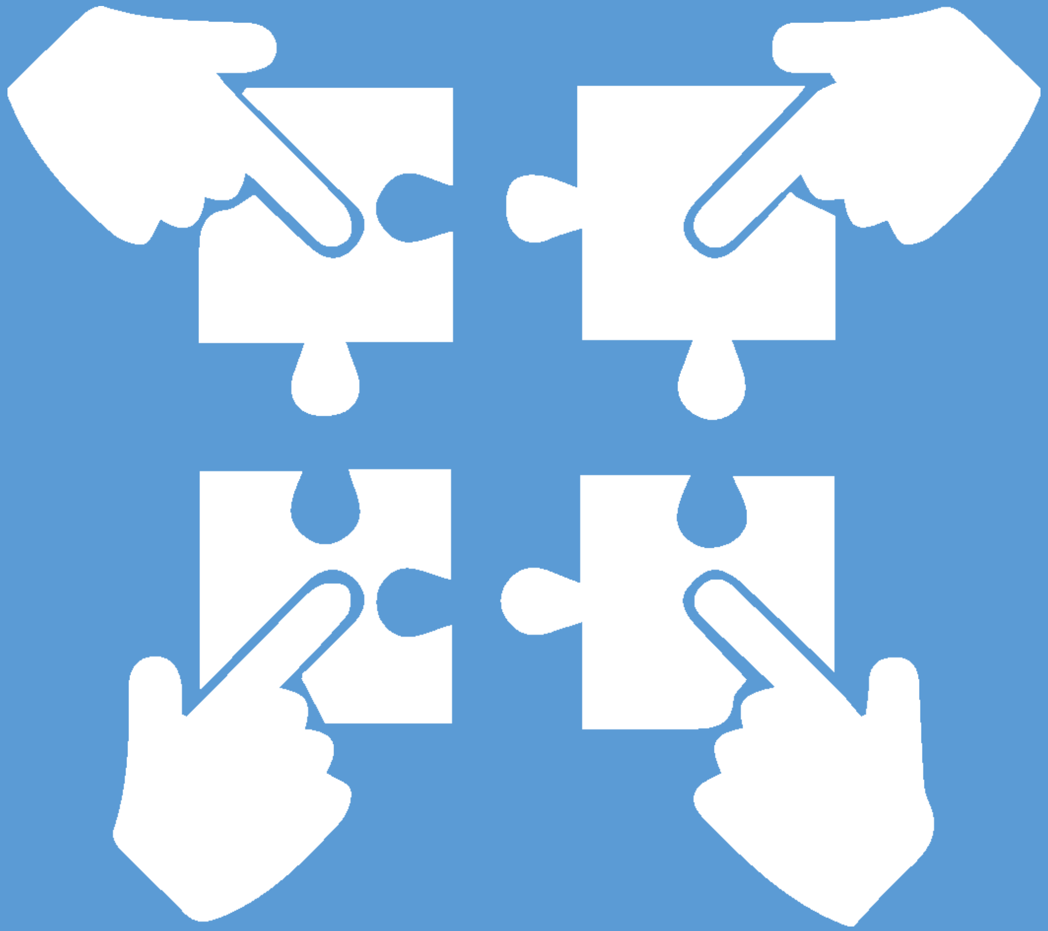



…back to the lesson…

TODAY'S LESSON

Programs are useful because they are reusable.

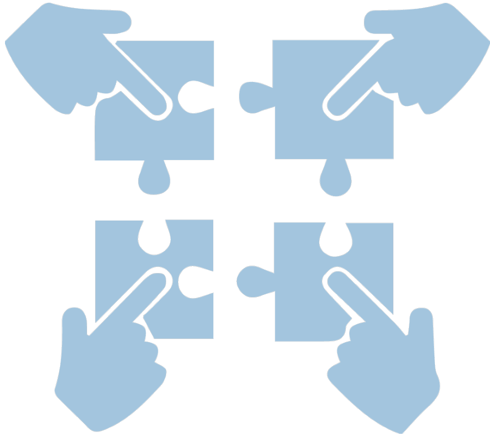
(among other reasons)



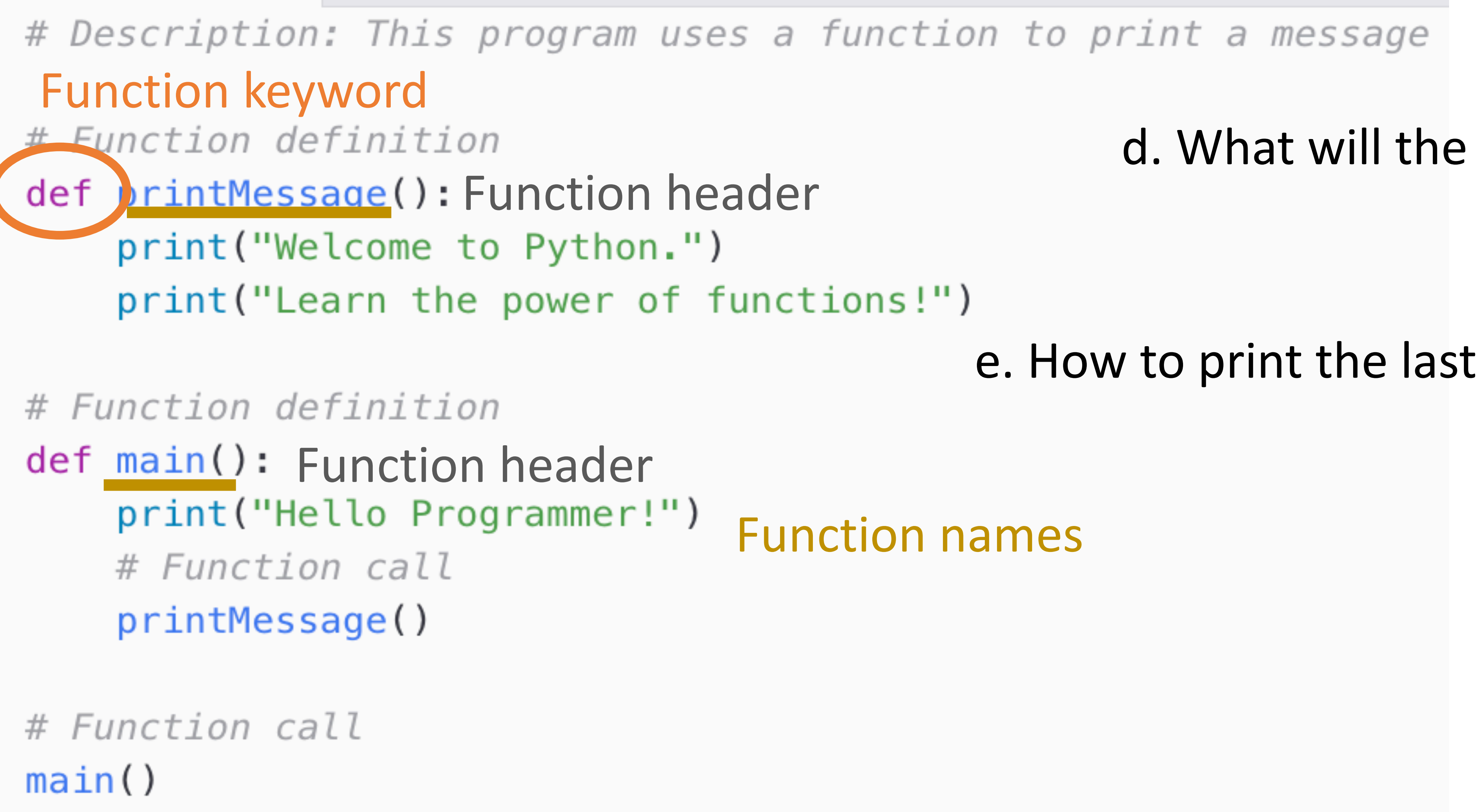
Process-Oriented Guided-Inquiry Learning (POGIL)

POGIL

- Look at Python Activity 12
- Find a partner and talk through question 1 & 2 together
  - Anything that says ‘enter and execute’, etc. we’ll do as class
- When time is up, we’ll execute the code as a class.



Look at POGIL Activity #12 Question 1



```
# Description: This program uses a function to print a message

# Function definition
def printMessage():
    print("Welcome to Python.")
    print("Learn the power of functions!")

# Function definition
def main():
    print("Hello Programmer!")
    # Function call
    printMessage()

# Function call
main()
```

Function keyword

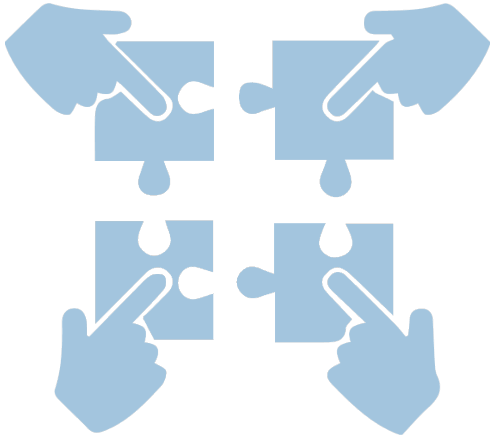
Function header

Function header

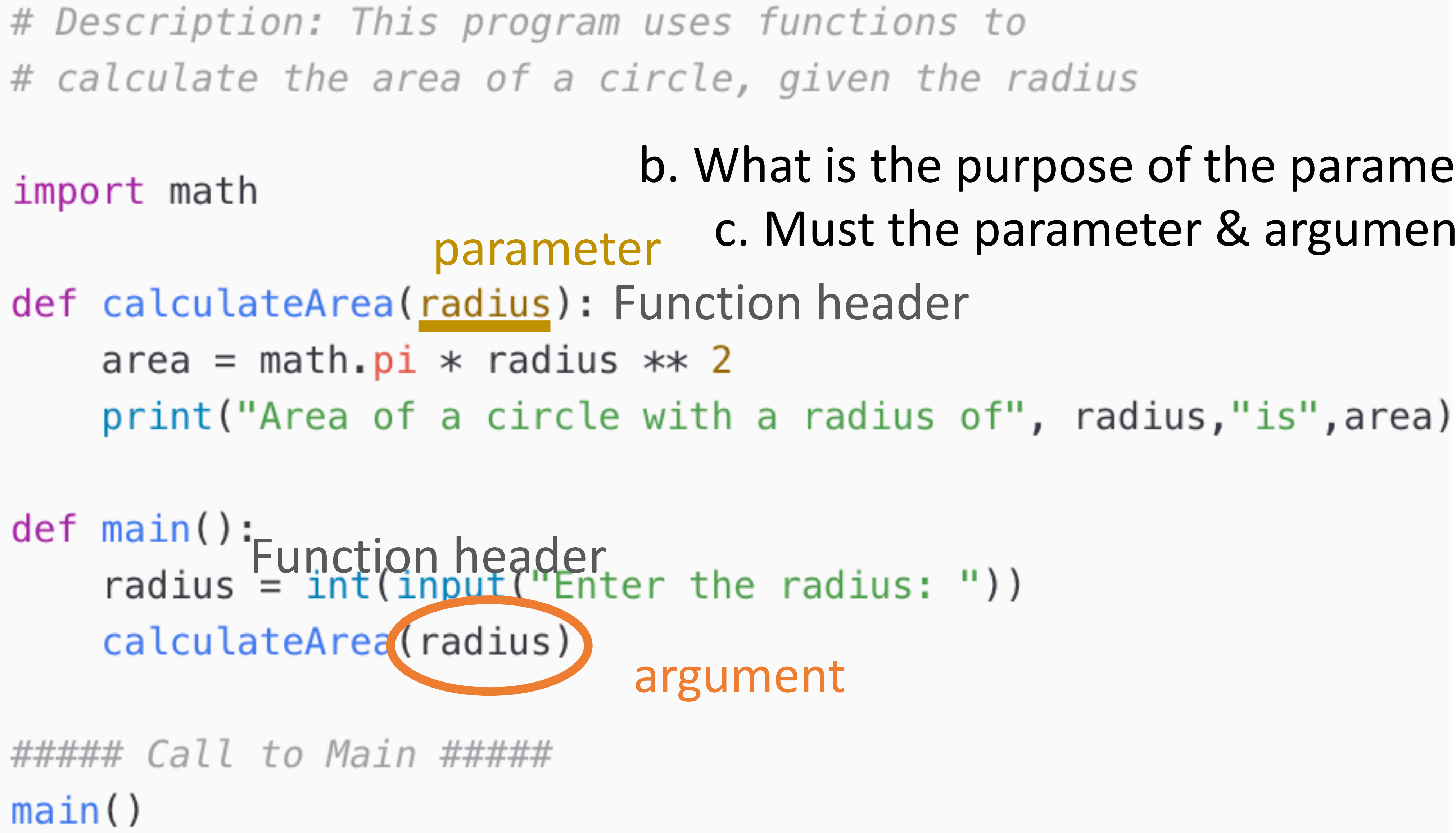
Function names

d. What will the output be?

e. How to print the last 2 lines twice?



Look at POGIL Activity #12 Question 2



```
# Description: This program uses functions to
# calculate the area of a circle, given the radius

import math

def calculateArea(radius):
    area = math.pi * radius ** 2
    print("Area of a circle with a radius of", radius,"is",area)

def main():
    radius = int(input("Enter the radius: "))
    calculateArea(radius)

##### Call to Main #####
main()
```

b. What is the purpose of the parameter/argument?

c. Must the parameter & argument be the same?

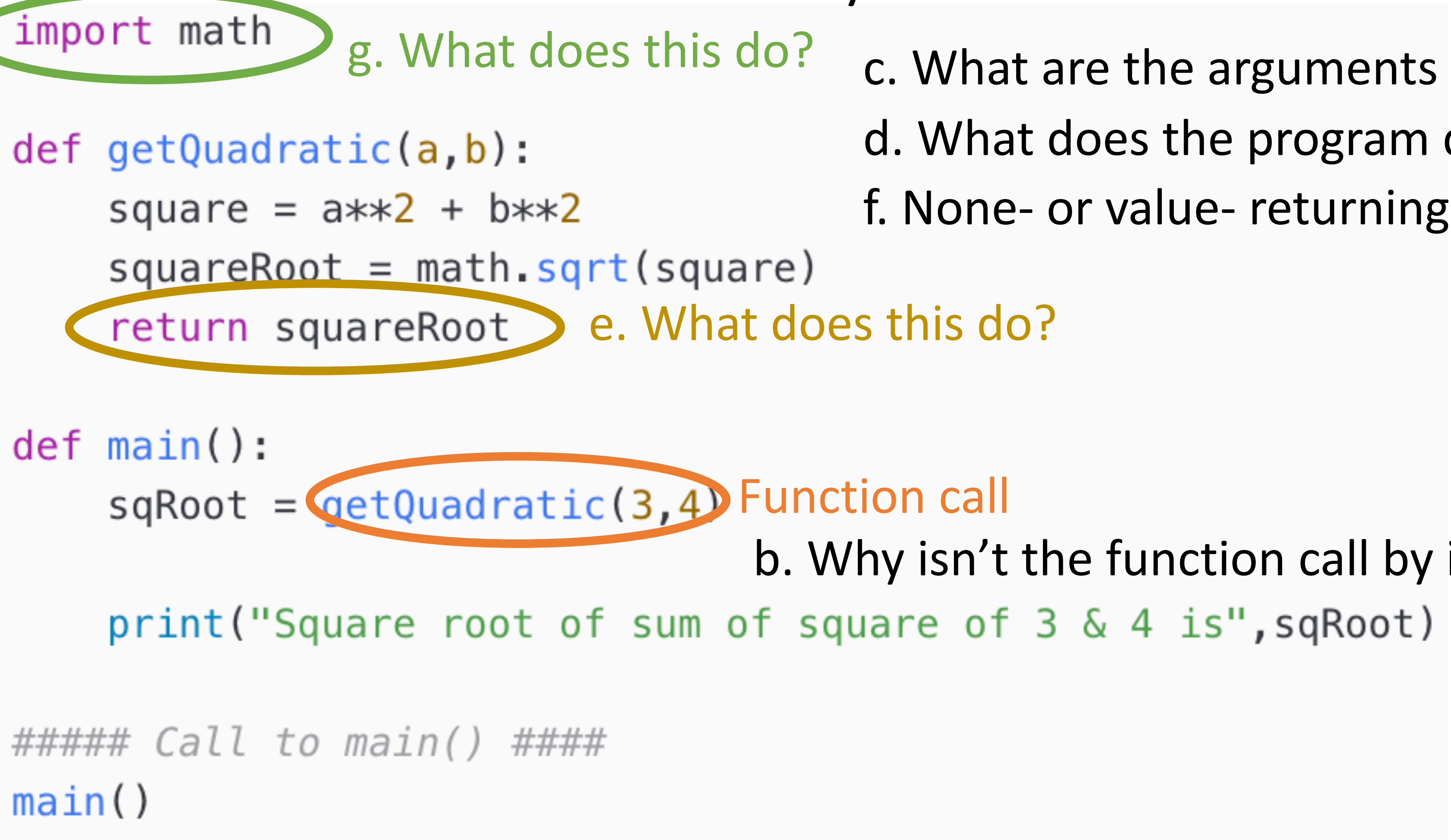
parameter

Function header

Function header

argument

Look at POGIL Activity #13 Question 1



```
import math

def getQuadratic(a,b):
    square = a**2 + b**2
    squareRoot = math.sqrt(square)
    return squareRoot

def main():
    sqRoot = getQuadratic(3,4)

    print("Square root of sum of square of 3 & 4 is",sqRoot)

##### Call to main() ####
main()
```

g. What does this do?

c. What are the arguments for?

d. What does the program do?

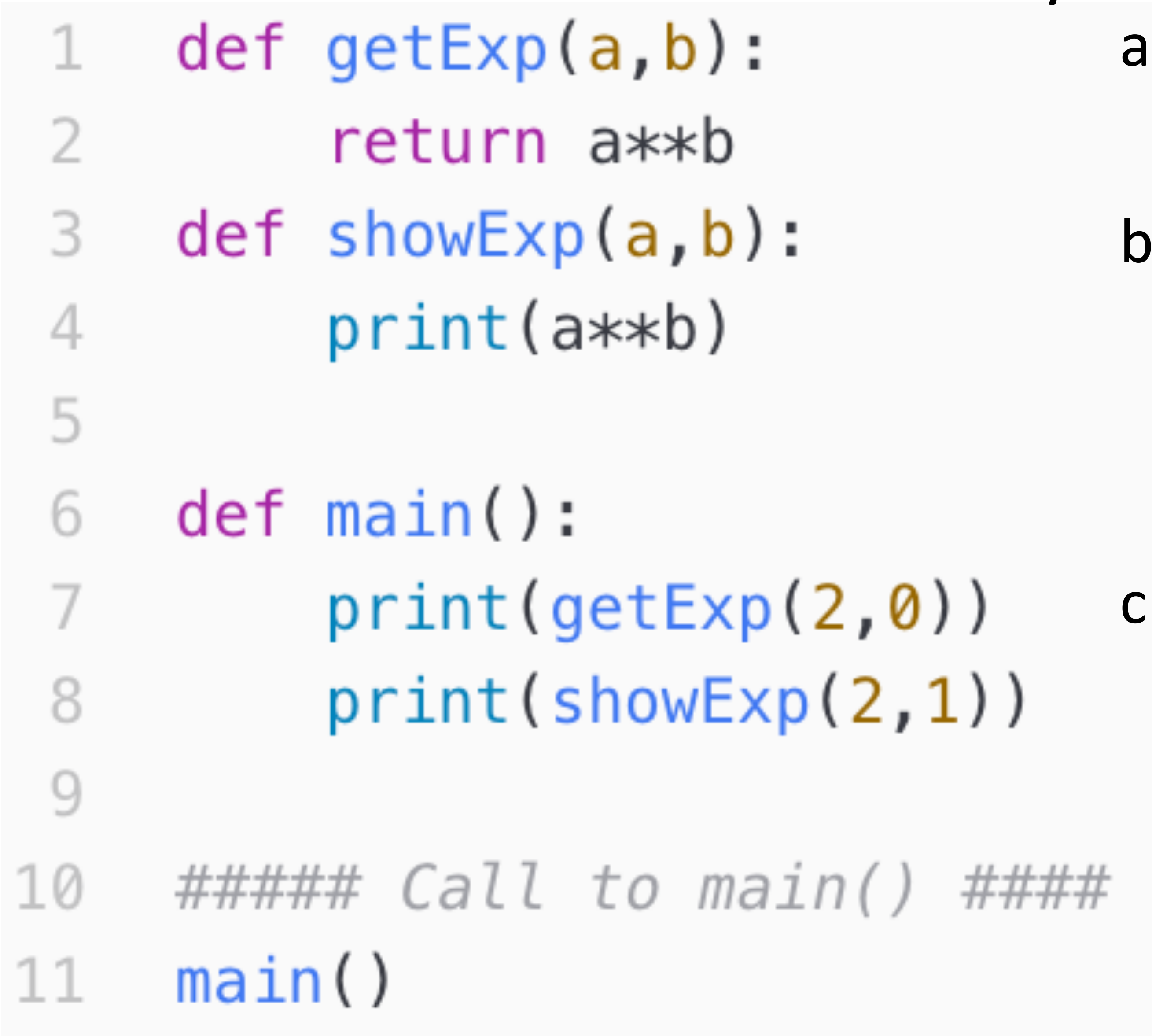
f. None- or value- returning?

e. What does this do?

Function call

b. Why isn’t the function call by itself?

Look at POGIL Activity #13 Question 2



```
def getExp(a,b):
    return a**b
def showExp(a,b):
    print(a**b)

def main():
    print(getExp(2,0))
    print(showExp(2,1))

##### Call to main() ####
main()
```

a. *getExp*: None- or value- returning?

b. *showExp*: None- or value- returning?

c. What will be printed?

Value-Returning Functions

- Once you `return` inside a function, you don’t continue on!

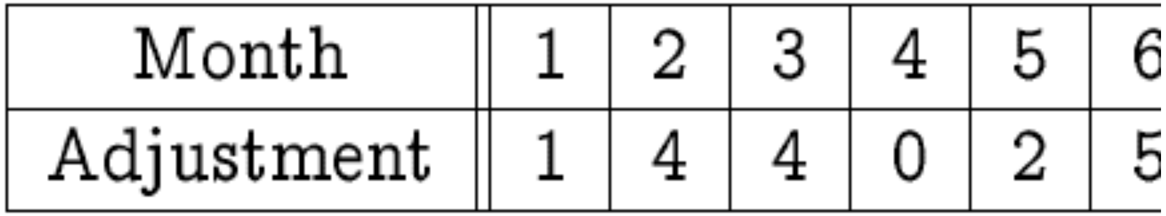
- You leave that function!

- Suggestions: only have one `return` statement that is reachable
  - With `if` statements, can have multiple!

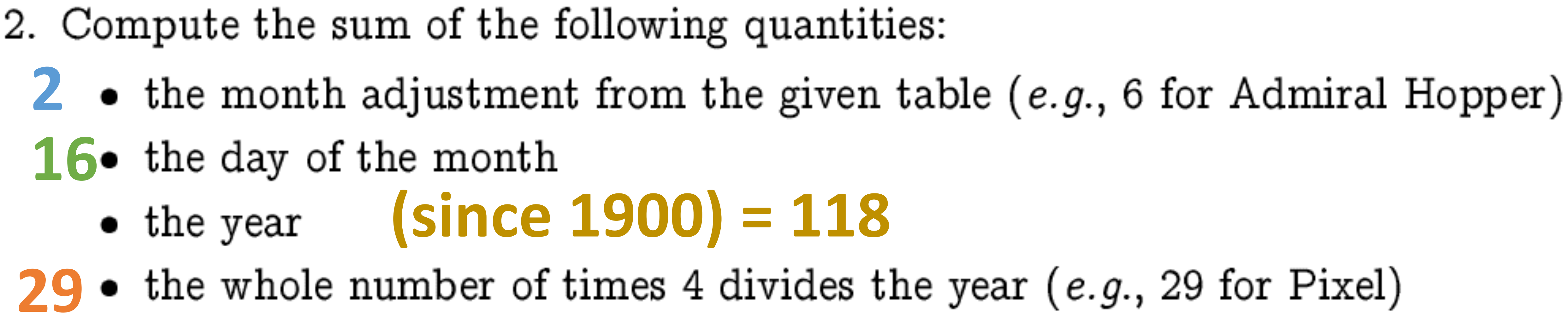
Interpreting an Algorithm

Pixel, the Sentient Snowball

Pixel, The Sentient Snowball, May 16, 2018



Month	1	2	3	4	5	6
Adjustment	1	4	4	0	2	5



2. Compute the sum of the following quantities:

2 • the month adjustment from the given table (*e.g.*, 6 for Admiral Hopper)

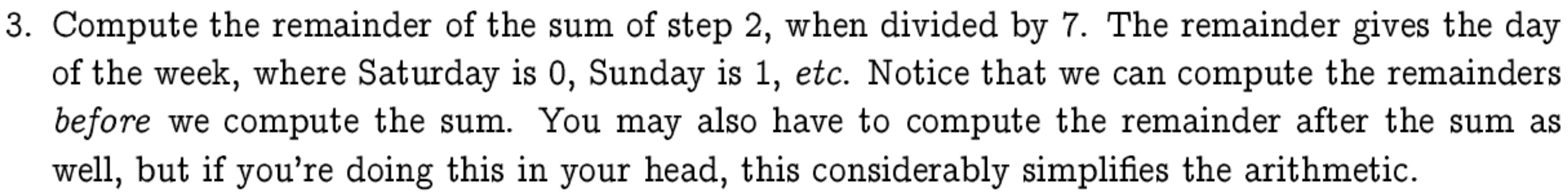
16 • the day of the month

• the year **(since 1900) = 118**

29 • the whole number of times 4 divides the year (*e.g.*, 29 for Pixel)

2+16+118+29 = 165

Pixel, The Sentient Snowball, May 16, 2018



3. Compute the remainder of the sum of step 2, when divided by 7. The remainder gives the day of the week, where Saturday is 0, Sunday is 1, *etc.* Notice that we can compute the remainders *before* we compute the sum. You may also have to compute the remainder after the sum as well, but if you're doing this in your head, this considerably simplifies the arithmetic.

165%7 = 4

Sat. = 0; Sun. = 1; Mon = 2, Tues = 3, Wed = 4

Pixel was born on a Wednesday

DayOfWeek “Lecture 3” Hand-out

- Look at the algorithm on one side
- Can you see where it is represented in the python code on the other side?

(There are some more advanced topics in the python code, like lists & if statements we haven’t yet covered)

```
month = int(input("Month (1-12): "))
day = int(input("Day (1-31): "))
year = int(input("Year (1900-2099): "))

# this is a *list* containing 12 integers.
adjustments = [ 1,4,4, 0,2,5, 0,3,6, 1,4,6 ]

# the integers in the adjustment list are indexed 0 through 11
# madj is the adjustment based on the particular month
madj = adjustments[month-1]

# it's best to think of the year as a value between 0 and 200
year -= 1900

# this is the main calculation:
sum = madj + day + (year//4) + year

# this is a correction for early in leap years
if (year%4 == 0) and (month <= 2) :
     sum -= 1

# a *list of strings*, indexed between 0 and 6 (remainders, mod 7)
dayName = ["Saturday", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
print(dayName[sum%7])
```

Recall: to *execute* this script, we can type:

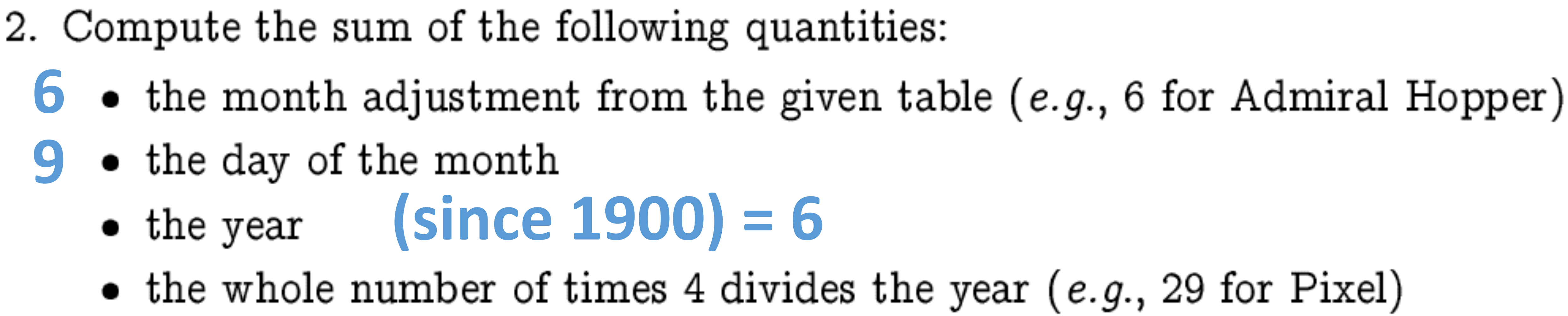
```
python3 dow.py
```

We can now *reuse* the code in the script, without re-typing the commands.

QUESTIONS?

Leftover Slides

Grace Hopper, December 9 1906



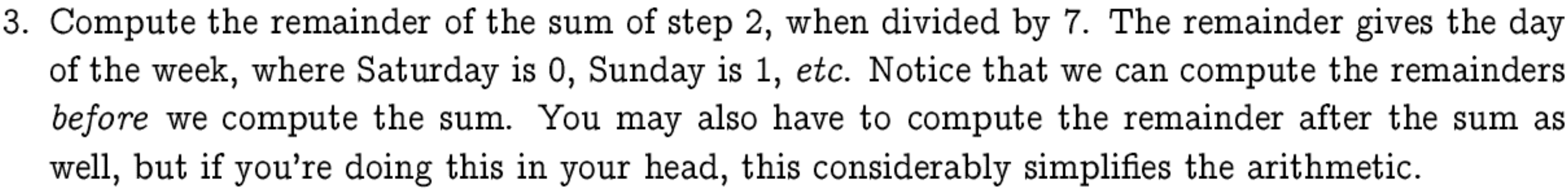
2. Compute the sum of the following quantities:

- 6 the month adjustment from the given table (*e.g.*, 6 for Admiral Hopper)
- 9 the day of the month
- the year (since 1900) = 6
- the whole number of times 4 divides the year (*e.g.*, 29 for Pixel)

1

6+9+6+1 = 22

Grace Hopper, December 9 1906



3. Compute the remainder of the sum of step 2, when divided by 7. The remainder gives the day of the week, where Saturday is 0, Sunday is 1, *etc*. Notice that we can compute the remainders *before* we compute the sum. You may also have to compute the remainder after the sum as well, but if you're doing this in your head, this considerably simplifies the arithmetic.

22%7 = 1

Saturday = 0
Sunday = 1

Admiral Grace Murray Hopper was born on a Sunday