```
#PROBLEM SET 1
# CONDITIONALS
```

DEEP THOUGHT PROBLEM

“All right,” said the computer, and settled into silence again. The two men fidgeted. The tension was unbearable.

“You’re really not going to like it,” observed Deep Thought.

“Tell us!”

“All right,” said Deep Thought. “The Answer to the Great Question…”

“Yes…!”

“Of Life, the Universe and Everything…” said Deep Thought.

“Yes…!”

“Is…” said Deep Thought, and paused.

“Yes…!”

“Is…”

“Yes…!!!…?”

“Forty-two,” said Deep Thought, with infinite majesty and calm.”

— The Hitchhiker’s Guide to the Galaxy, Douglas Adams

In deep.py, implement a program that prompts the user for the answer to the Great Question of Life, the Universe and Everything, outputting Yes if the user inputs 42 or (case-insensitively) forty-two or forty two. Otherwise output No.

Hints

No need to convert the user’s input to an int if you check for equality with "42", a str, rather than 42, an int!

It’s okay if your output or the user’s wraps onto multiple lines.

```
answer = input("What is the answer to the Great Question of Life, the Universe and Everything? ")
if answer.strip().casefold() in ["42","forty-two","forty two"]:
  print("Yes")
else:
  print("No")
```

```
What is the answer to the Great Question of Life, the Universe and Everything?          forty-two
Yes
```

HOME FEDERAL SAVINGS BANK PROBLEM

In season 7, episode 24 of Seinfeld, Kramer visits a bank that promises to give $100toanyonewhoisn'tgreetedwitha“hello.”Kramerisinsteadgreetedwitha“hey,”whichheinsistsisn'ta“hello,$100. The $”andsoheasksfor$
bank’s manager proposes a compromise: “You got a greeting that starts with an ‘h,’ how does $20 sound?” Kramer accepts.

In a file called bank.py, implement a program that prompts the user for a greeting. If the greeting starts with “hello”, output $0.Ifthegreetingstartswithan“h”(butnot“hello”),output$20. Otherwise, output $100. Ignore any leading whitespace in the user’s greeting, and treat the user’s greeting case-insensitively.

Hints

Recall that a str comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods.

Be sure to give $0 not only for “hello” but also “hello there”, “hello, Newman”, and the like.

```
greeting = input("Greeting: ").strip().lower()
if greeting.startswith("hello"):
  print("$0")
elif greeting.startswith("h"):
  print("$20")
else:
  print("$100")
```

```
Greeting:         What's happening?
$100
```

FILE EXTENSIONS PROBLEM

Even though Windows and macOS sometimes hide them, most files have file extensions, a suffix that starts with a period (.) at the end of their name. For instance, file names for GIFs end with .gif, and file names for JPEGs end with .jpg or .jpeg. When you double-click on a file to open it, your computer uses its file extension to determine which program to launch.

Web browsers, by contrast, rely on media types, formerly known as MIME types, to determine how to display files that live on the web. When you download a file from a web server, that server sends an HTTP header, along with the file itself, indicating the file's media type. For instance, the media type for a GIF is image/gif, and the media type for a JPEG is image/jpeg. To determine the media type for a file, a web server typically looks at the file's extension, mapping one to the other.

See developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types for common types.

In a file called extensions.py, implement a program that prompts the user for the name of a file and then outputs that file's media type if the file's name ends, case-insensitively, in any of these suffixes:

.gif .jpg .jpeg .png .pdf .txt .zip

If the file's name ends with some other suffix or has no suffix at all, output application/octet-stream instead, which is a common default.

Hints

Recall that a str comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods.

```
extensions = input("File Name: ").strip().lower().casefold()
if extensions.endswith(".gif"):
  print("image/gif")
elif extensions.endswith(".jpg"):
  print("image/jpeg")
elif extensions.endswith(".jpeg"):
  print("image/jpeg")
elif extensions.endswith(".png"):
  print("image/png")
elif extensions.endswith(".pdf"):
  print("application/pdf")
elif extensions.endswith(".zip"):
  print("application/zip")
elif extensions.endswith(".txt"):
  print("text/plain")
else:
  print("application/octet-stream")
```

```
File Name: document.pdf
application/pdf
```

MATH INTERPRETER PROBLEM

Python already supports math, whereby you can write code to add, subtract, multiply, or divide values and even variables. But let's write a program that enables users to do math, even without knowing Python.

In a file called interpreter.py, implement a program that prompts the user for an arithmetic expression and then calculates and outputs the result as a floating-point value formatted to one decimal place. Assume that the user's input will be formatted as x y z, with one space between x and y and one space between y and z, wherein:

x is an integer

y is +, -, *, or /

z is an integer

For instance, if the user inputs 1 + 1, your program should output 2.0. Assume that, if y is /, then z will not be 0.

Note that, just as python itself is an interpreter for Python, so will your interpreter.py be an interpreter for math!

Hints

Recall that a str comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods, including split, which separates a str into a sequence of values, all of which can be assigned to variables at once. For instance, if expression is a str like 1 + 1, then

x, y, z = expression.split(" ")

will assign 1 to x, + to y, and 1 to z.

```
calculation = input("Expression: ")
x, y, z = calculation.split(" ")

x = int(x)
z = int(z)

if y == "+":
  result = x + z
elif y == "-":
  result = x - z
elif y == "*":
  result = x * z
elif y == "/":
  result = x / z

print(f"Result: {result:.1f}")
```

```
Expression: 4 / 3
Result: 1.3
```

MEAL TIME PROBLEM

Suppose that you're in a country where it's customary to eat breakfast between 7:00 and 8:00, lunch between 12:00 and 13:00, and dinner between 18:00 and 19:00. Wouldn't it be nice if you had a program that could tell you what to eat when?

In meal.py, implement a program that prompts the user for a time and outputs whether it's breakfast time, lunch time, or dinner time. If it's not time for a meal, don't output anything at all. Assume that the user's input will be formatted in 24-hour time as #:## or ##:##. And assume that each meal's time range is inclusive. For instance, whether it's 7:00, 7:01, 7:59, or 8:00, or anytime in between, it's time for breakfast.

Structure your program per the below, wherein convert is a function (that can be called by main) that converts time, a str in 24-hour format, to the corresponding number of hours as a float. For instance, given a time like "7:30" (i.e., 7 hours and 30 minutes), convert should return 7.5 (i.e., 7.5 hours).

def main(): ...

def convert(time): ...

if **name** == "**main**": main()

Hints

Recall that a str comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods, including split, which separates a str into a sequence of values, all of which can be assigned to variables at once. For instance, if time is a str like "7:30", then

hours, minutes = time.split(":")

will assign "7" to hours and "30" to minutes.

Keep in mind that there are 60 minutes in 1 hour.

```
def convert(time):
    hours, minutes = time.split(":")
    hours = int(hours)
    minutes = int(minutes)
    hours += minutes / 60
    return hours

def main():
    time = input("What time is it? ")
    hours = convert(time)

    if 7.0 <= hours < 8.0:
        print("breakfast time")
    elif 12.0 <= hours < 13.0:
        print("lunch time")
    elif 18.0 <= hours < 19.0:
        print("dinner time")

main()
```

```
What time is it? 12:05
lunch time
```