Introduction to Python

Names and Functions
(Day 1)

Aim: How can we write expressions with names and functions?

Variables

One of the most powerful features of a programming language is the ability to manipulate *variables*. A variable is a name that refers to a value.

An *assignment statement* creates new variables and gives them values:

```
>>> message = 'And now for something completely different'
>>> n = 17
>>> pi = 3.1415926535897931
```

This example makes three assignments. The first assigns a string to a new variable named `message`; the second assigns the integer `17` to `n`; the third assigns the (approximate) value of π to `pi`.

To display the value of a variable, you can use a print statement:

```
>>> print(n)
17
>>> print(pi)
3.141592653589793
```

The type of a variable is the type of the value it refers to.

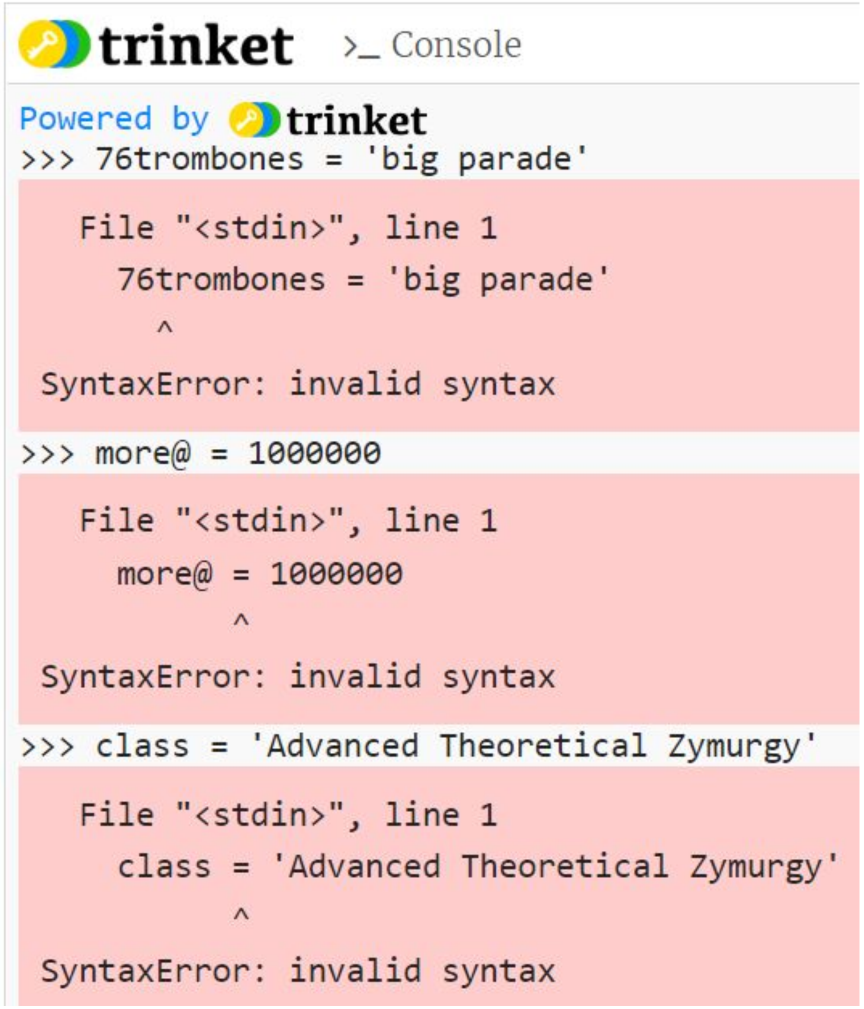
```
>>> type(message)
<class 'str'>
>>> type(n)
<class 'int'>
>>> type(pi)
<class 'float'>
```

Variable names and keywords

Programmers generally choose names for their variables that are meaningful and document what the variable is used for.

Variable names can be arbitrarily long. They can contain both letters and numbers, but they cannot start with a number. It is legal to use uppercase letters, but it is a good idea to begin variable names with a lowercase letter (you'll see why later).

The underscore character (`_`) can appear in a name. It is often used in names with multiple words, such as `my_name` or `airspeed_of_unladen_swallow`. Variable names can start with an underscore character, but we generally avoid doing this unless we are writing library code for others to use.



```
>>> 76trombones = 'big parade'

  File "<stdin>", line 1
    76trombones = 'big parade'
      ^

SyntaxError: invalid syntax
>>> more@ = 1000000

  File "<stdin>", line 1
    more@ = 1000000
          ^

SyntaxError: invalid syntax
>>> class = 'Advanced Theoretical Zymurgy'

  File "<stdin>", line 1
    class = 'Advanced Theoretical Zymurgy'
          ^

SyntaxError: invalid syntax
```

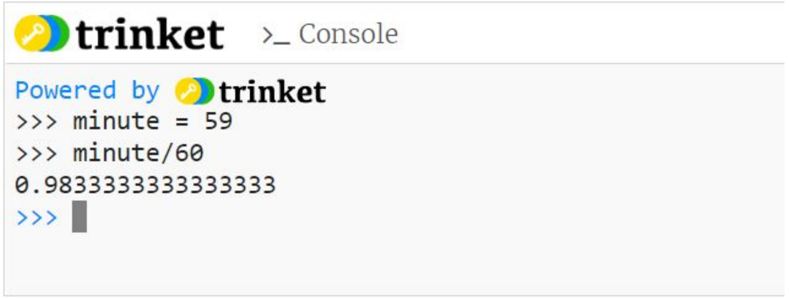
Take a guess:
What is the “Syntax Error” for the name in each statement?

It turns out that `class` is one of Python's *keywords*. The interpreter uses keywords to recognize the structure of the program, and they cannot be used as variable names.

Python reserves 33 keywords:

```
and       del       from      None      True
as        elif      global    nonlocal  try
assert    else      if        not       while
break     except    import    or        with
class     False     in        pass      yield
continue  finally   is        raise
def       for       lambda    return
```

Operators and operands

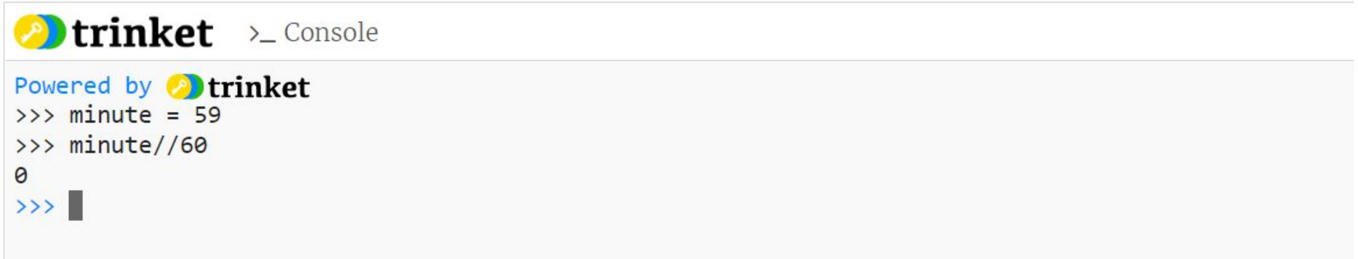


```
Powered by trinket
>>> minute = 59
>>> minute/60
0.9833333333333333
>>> 
```

The division operator in Python 2.0 would divide two integers and truncate the result to an integer:

```
>>> minute = 59
>>> minute/60
0
```

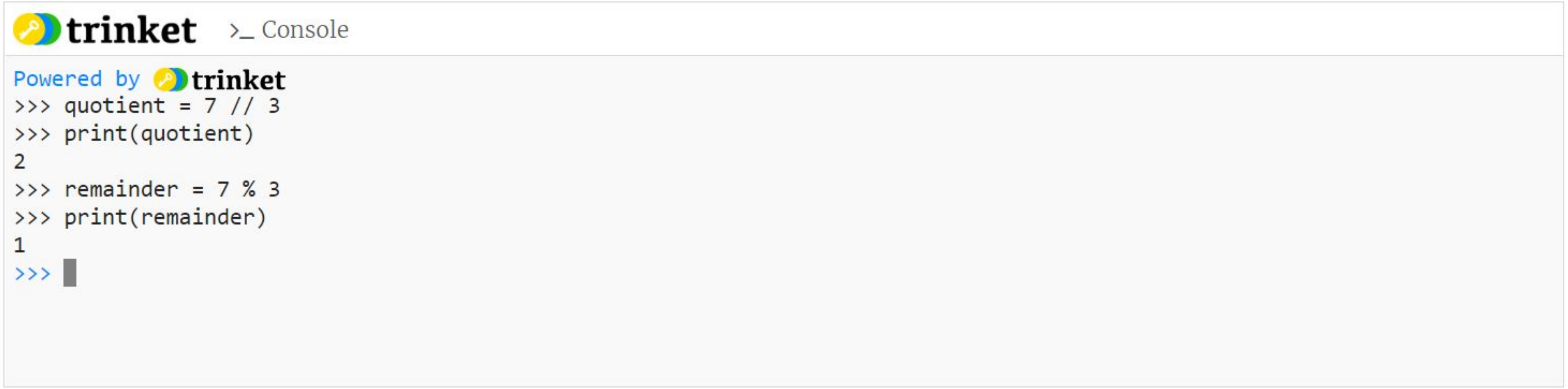
To obtain the same answer in Python 3.0 use floored (// integer) division.



```
Powered by trinket
>>> minute = 59
>>> minute//60
0
>>> 
```

Modulus operator

The *modulus operator* works on integers and yields the remainder when the first operand is divided by the second. In Python, the modulus operator is a percent sign (`%`). The syntax is the same as for other operators:

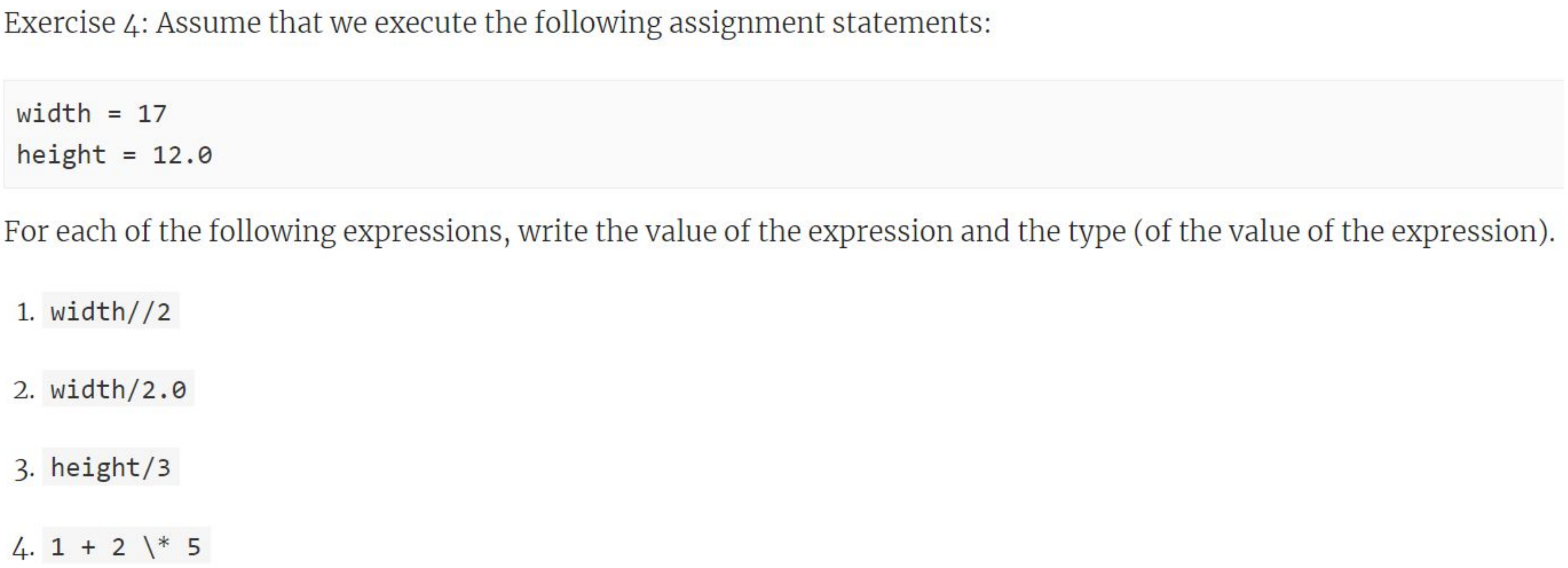

trinket >_ Console

```
Powered by trinket
>>> quotient = 7 // 3
>>> print(quotient)
2
>>> remainder = 7 % 3
>>> print(remainder)
1
>>> 
```

So 7 divided by 3 is 2 with 1 left over.

How is “Modulus” useful ?

You Try it
(Predict the output for each expression. If you are not sure, run the code.)



Exercise 4: Assume that we execute the following assignment statements:

```
width = 17
height = 12.0
```

For each of the following expressions, write the value of the expression and the type (of the value of the expression).

1. `width//2`
2. `width/2.0`
3. `height/3`
4. `1 + 2 \* 5`

Asking the user for input

Sometimes we would like to take the value for a variable from the user via their keyboard. Python provides a built-in function called `input` that gets input from the keyboard[1]. When this function is called, the program stops and waits for the user to type something. When the user presses `Return` or `Enter`, the program resumes and `input` returns what the user typed as a string.

```
>>> input = input()
Some silly stuff
>>> print(input)
Some silly stuff
```

Before getting input from the user, it is a good idea to print a prompt telling the user what to input. You can pass a string to `input` to be displayed to the user before pausing for input:

```
>>> name = input('What is your name?\n')
What is your name?
Chuck
>>> print(name)
Chuck
```

The sequence `\n` at the end of the prompt represents a *newline*, which is a special character that causes a line break. That's why the user's input appears below the prompt.

If you expect the user to type an integer, you can try to convert the return value to `int` using the `int()` function:

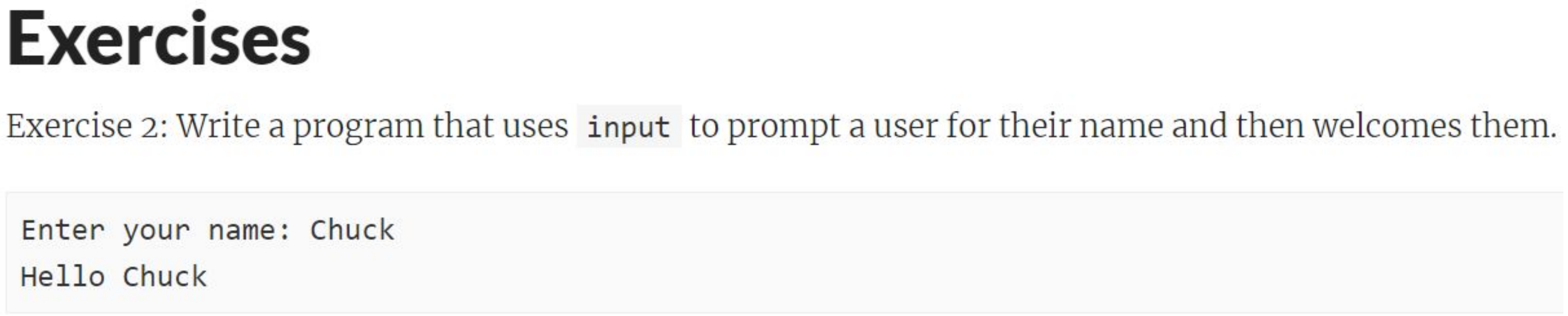
```
>>> prompt = 'What...is the airspeed velocity of an unladen swallow?\n'
>>> speed = input(prompt)
What...is the airspeed velocity of an unladen swallow?
17
>>> int(speed)
17
>>> int(speed) + 5
22
```

But if the user types something other than a string of digits, you get an error:

```
>>> speed = input(prompt)
What...is the airspeed velocity of an unladen swallow?
What do you mean, an African or a European swallow?
>>> int(speed)
ValueError: invalid literal for int() with base 10:
```

We will see how to handle this kind of error later.

You Try it
(post a screenshot of your program from Python.org)



Exercises

Exercise 2: Write a program that uses `input` to prompt a user for their name and then welcomes them.

```
Enter your name: Chuck
Hello Chuck
```

You Try it
(post a screenshot of your program from Python.org)

Exercise 3: Write a program to prompt the user for hours and rate per hour to compute gross pay.

```
Enter Hours: 35
Enter Rate: 2.75
Pay: 96.25
```

We won't worry about making sure our pay has exactly two digits after the decimal place for now. If you want, you can play with the built-in Python `round` function to properly round the resulting pay to two decimal places.

Challenge
(post a screenshot of your program from Python.org)

Write a program which prompts the user for a Celsius temperature, convert the temperature to Fahrenheit, and print out the converted temperature.