COMP1021
Introduction to Computer Science

Handling Repeating Patterns

Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Use the modulus operator to find odd/even numbers
  2. Write code to capture repeating patterns using the modulus operator

The Modulus Operator

- In this presentation, we will look at using the modulus operator, %, to capture repeating patterns
- The % operator is used to find ‘the remainder after division’
- Here are some examples:
  - 10 % 2 = 0
  - 10 % 3 = 1
  - 10 % 4 = 2
  - 10 % 5 = 0
- The % operator always gives you a number between 0 and the divisor minus 1

Using % to Find Odd/Even Numbers

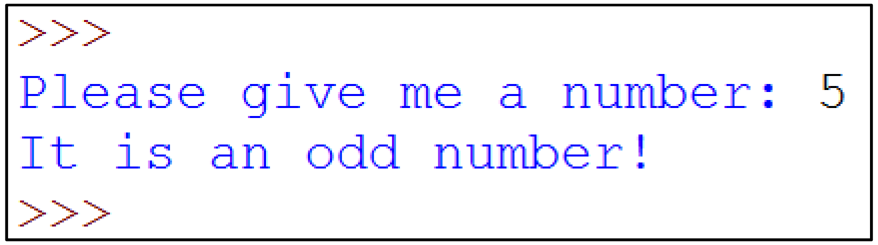
- Apart from finding the remainder of a division, a very common use of the % operator is to determine whether a number is an odd/even number
- This is because, when an odd number is divided by 2, the remainder is 1 whereas the remainder of dividing an even number by 2 is 0
- Combining an if statement and the % operator you can make a program to tell the user whether a given number is an odd/even number, as shown on the next slide

An Example of Finding Odd/Even Numbers

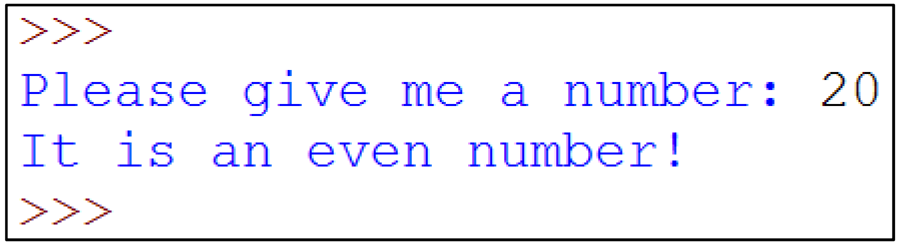
- Here is the example:

```
number = int(input("Please give me a number: "))

if number % 2 == 1:
    print("It is an odd number!")
else:
    print("It is an even number!")
```



```
>>>
Please give me a number: 5
It is an odd number!
>>>
```



```
>>>
Please give me a number: 20
It is an even number!
>>>
```

Using Numbers as Conditions

- We can simplify this code:

```
if number % 2 == 1:
        ...the number is odd, do something...
```

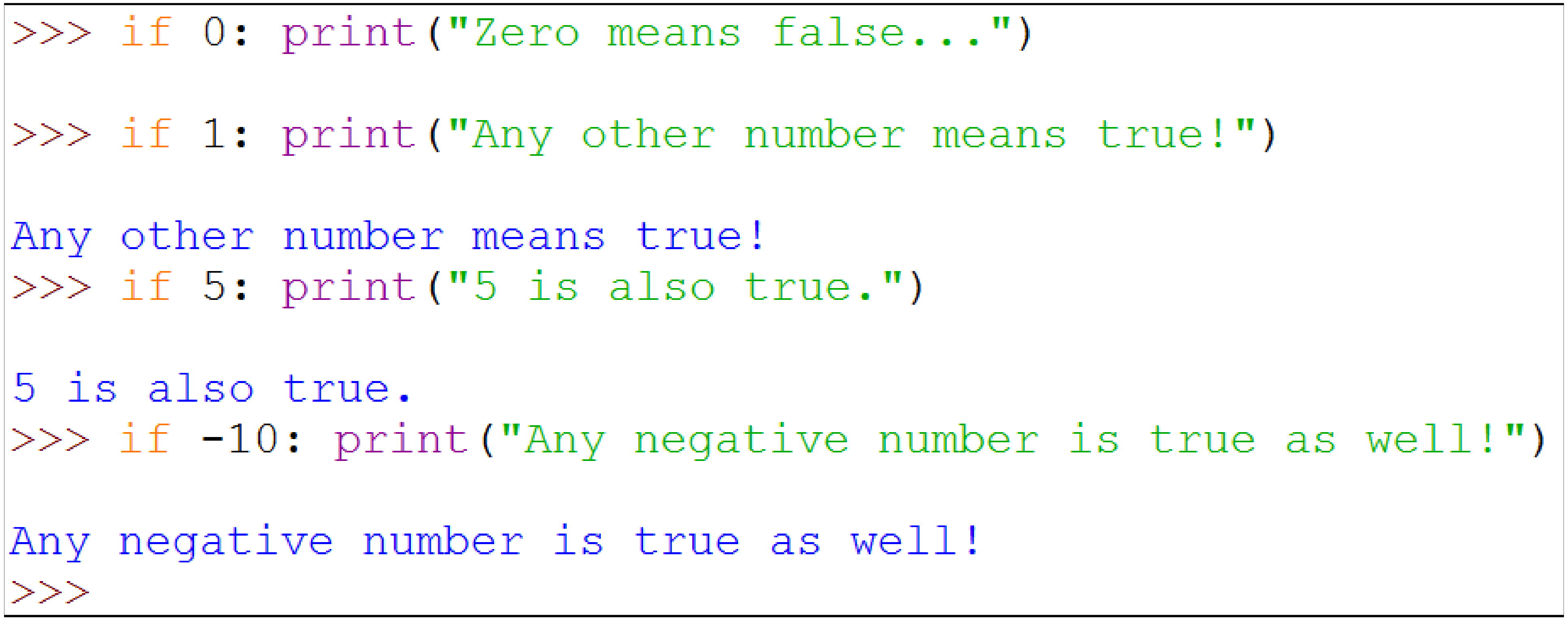
into this:

```
if number % 2:
        ...the number is odd, do something...
```

- In Python, a value of 0 is equivalent to `False` and any other number is equivalent to `True`, see examples on the next slide

Examples of Using Numbers as Conditions

- Using various numbers as an if condition:



```
>>> if 0: print("Zero means false...")

>>> if 1: print("Any other number means true!")

Any other number means true!
>>> if 5: print("5 is also true.")

5 is also true.
>>> if -10: print("Any negative number is true as well!")

Any negative number is true as well!
>>>
```

Using % for Patterns

- Odd/even numbers is a kind of pattern with a cycle of two, like this:

`number` $\quad 0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ \ldots$

`number % 2` $\quad \underbrace{0\ \ 1}\ \ \underbrace{0\ \ 1}\ \ \underbrace{0\ \ 1}\ \ 0\ \ \ldots$

Cycles in the repeating pattern

- If we use other numbers as the divisor, we can find repeating patterns with a different size, e.g.:

`number` $\quad 0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ \ldots$

`number % 4` $\quad \underbrace{0\ \ 1\ \ 2\ \ 3}\ \ \underbrace{0\ \ 1\ \ 2\ \ 3}\ \ 0\ \ \ldots$

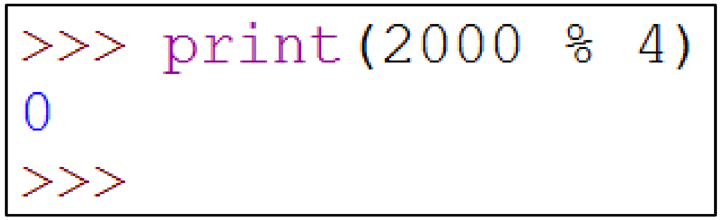
Cycles in the repeating pattern

Leap Years

- Leap year is a year with 366 days (the years with 365 days are called common years)
- A simple rule is that we have a leap year every four year
  - There are some exceptions to this rule but we will ignore these exceptions in our example
- For example, if 2008 is a leap year, 2012 will also be a leap year
- We will make a program to show the leap years between 2000 and 2015

Finding Leap Years

- Leap years happen in a pattern with a 4-year cycle so let's use the % operator to find leap years
- We know that year 2000 is a leap year and we can determine the location of a leap year within the 4-year cycle using 2000 % 4, i.e.:



```
>>> print(2000 % 4)
0
>>>
```

- The result is 0 and that means a leap year is at the start of the 4-year cycle given by the % operator

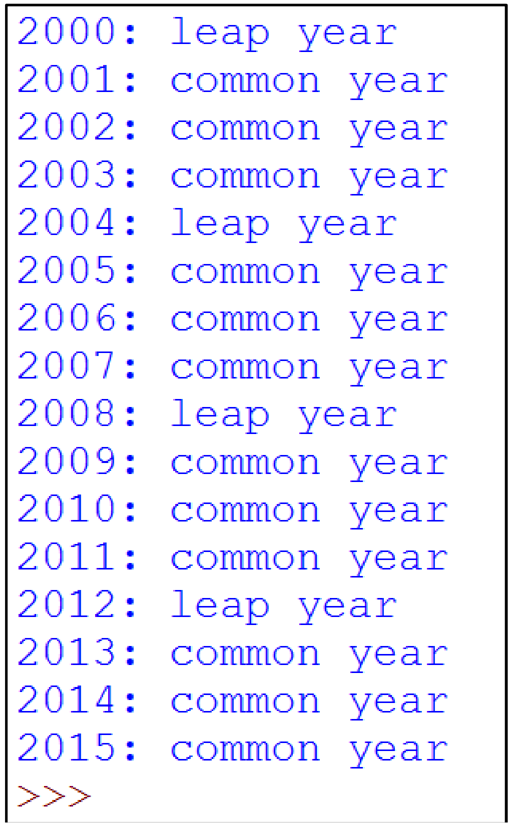
The Program

- Since year 2000 is at the start of the 4-year cycle, any other year at the start of the cycle is also a leap year
- Using this, we can build a program to find the leap years between 2000 and 2015 like this:

```
startyear = 2000
endyear = 2015

for year in range(startyear, \
                  endyear + 1):
    print(year, end=": ")

    if year % 4 == 0:
        print("leap year")
    else:
        print("common year")
```



```
2000: leap year
2001: common year
2002: common year
2003: common year
2004: leap year
2005: common year
2006: common year
2007: common year
2008: leap year
2009: common year
2010: common year
2011: common year
2012: leap year
2013: common year
2014: common year
2015: common year
>>>
```