COMP1021
Introduction to Computer Science

Handling Repeating Patterns

Gibson Lam and David Rossiter

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Use the modulus operator to identify 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 identify repeating patterns
- The % operator gives you '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

- A common use of the % operator is to determine whether a number is an odd/even number
- When an odd number is divided by 2 the remainder is 1, whereas the remainder of dividing an even number by 2 is 0
- By 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*...

- This is because 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 are a pattern with a cycle of two, as shown below:

```
number        0  1  2  3  4  5  6  …
number % 2    0  1  0  1  0  1  0  …
```

$\underbrace{0\ 1}\ \underbrace{0\ 1}\ \underbrace{0\ 1}\ 0\ \ldots$

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        0  1  2  3  4  5  6  7  8  …
number % 4    0  1  2  3  0  1  2  3  0  …
```

$\underbrace{0\ 1\ 2\ 3}\ \underbrace{0\ 1\ 2\ 3}\ 0\ \ldots$

Cycles in the repeating pattern

Leap Years

- A leap year is a year which has 366 days (the years with 365 days are called common years)
- We have a leap year every four years
  - *There are some exceptions to this, but we will ignore those exceptions in this presentation*
- For example, if 2008 is a leap year, 2012 will also be a leap year because it is 4 years later
- Let's 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 the 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 which means a leap year is at the start of the 4-year cycle

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
>>>
```