# Module 2: Introductory Python

Dr. Foo Yee Loo

Dr. Sarina Mansor

FUTURE READY YOU | YOUR NEXT IS NOW

A Subsidiary of MMU MULTIMEDIA UNIVERSITY

1 Basic Python

2 If-else, for and while loops

3 Functions

4 Classes, Instances

5 Methods, Inheritance

## Python

- Invented in the Netherlands, early 90s by Guido van Rossum
  - Named after Monty Python (British Comedy)

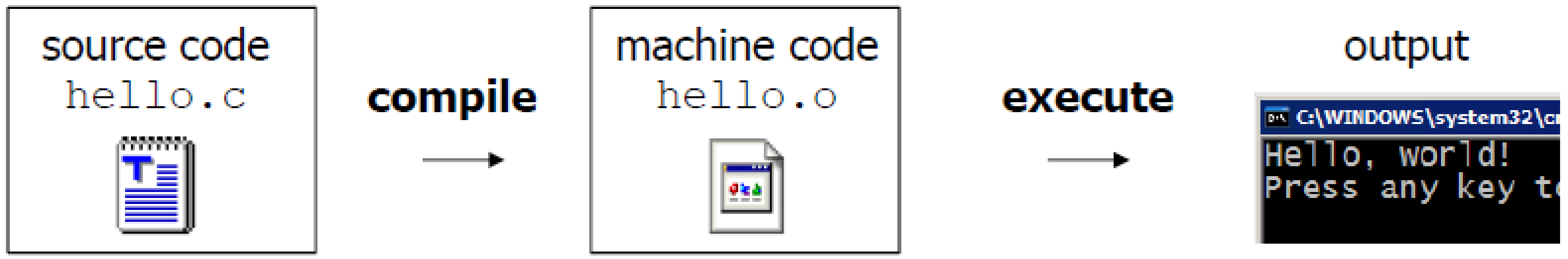
- Python is a scripting/interpreted language

- It is relatively easy to get started.

- It comes with interactive shell to experience how programs run.

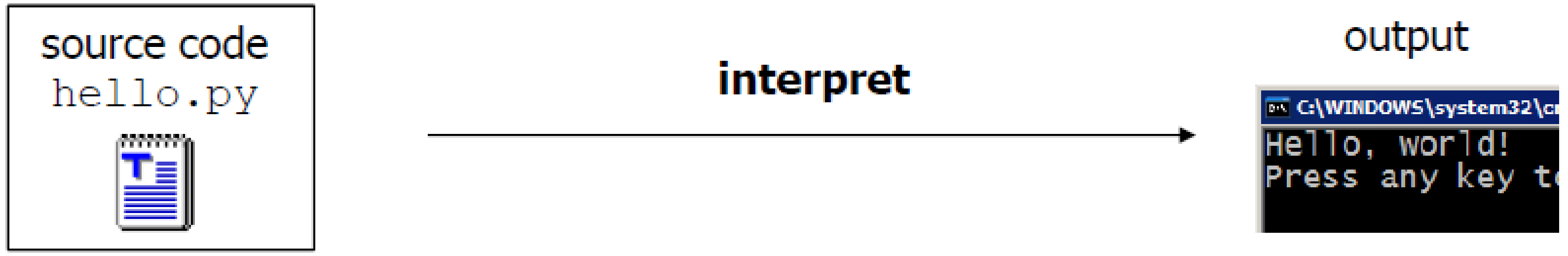
# Compiled Vs Interpreted Languages

- Compiled languages have to be translate source code to machine code for computer to run (e.g. C, C++, Fortran)



```
Hello, world!
Press any key t
```

- Interpreted languages (e.g. Python, Matlab) are directly run by computer

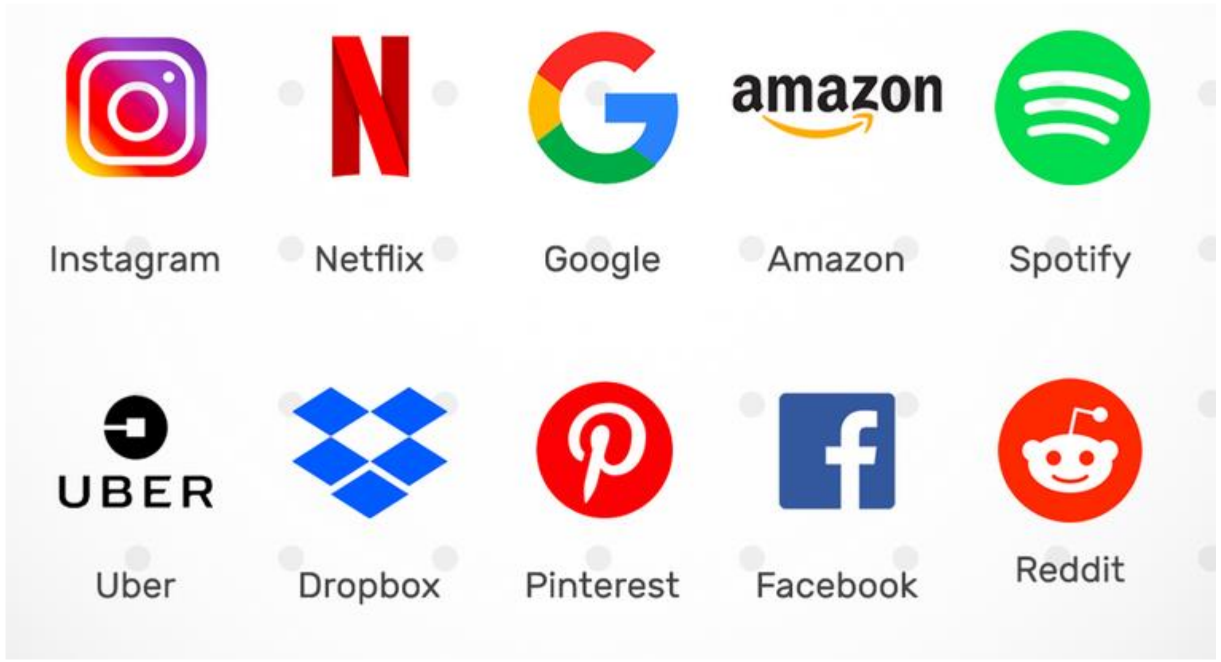


```
Hello, world!
Press any key t
```

## Python Applications

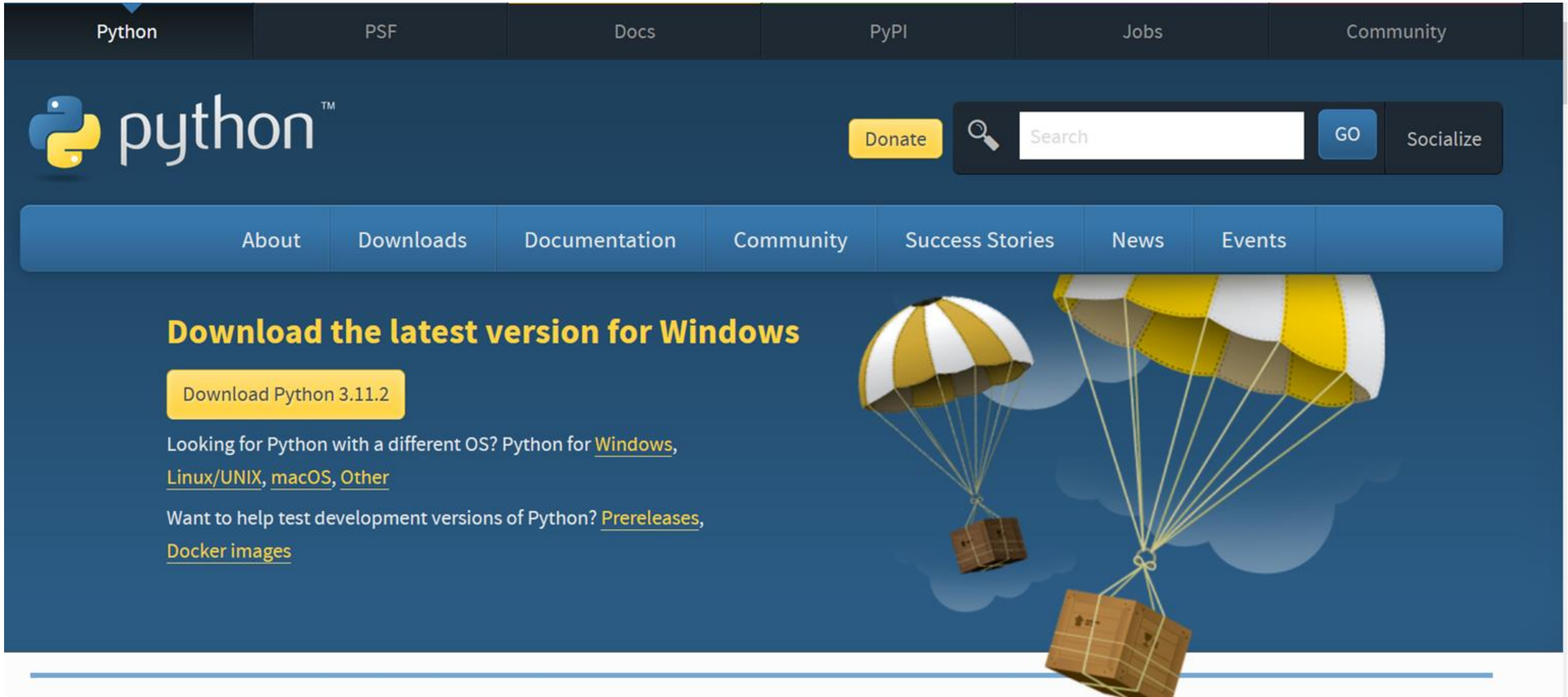
Popular applications: AI, Data Science, Scientific and Mathematical Computing, Web Development, Computer Graphics, Games Development, Embedded System.

Companies that use Python:



## Downloading Python

- Download Python from [https://www.python.org/downloads/](https://www.python.org/downloads/)



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python? Prereleases, Docker images

## Tools for Coding Python

- Python source code is saved as a regular text file that has the **.py** extension

  Can be written in any **plain text** editor (eg: Notepad, Text Editor)

- Integrated Development Environment (IDE)
  - Popular choices for Python IDE
    - IDLE – Built-in with Python installer
    - Thonny – Suitable for beginners
    - Spyder – Data science
    - Jupyter Notebook – Word processing, interactive codes

## Google Colab

- A web-based version of Jupyter Notebook that enables you to write and execute Python code.
- Sign-in at https://colab.research.google.com
- Open a new notebook
- Try type: `print("Hello World!")`
- Click the Run button

- Congrats, you have written your first line of code!

# Chapter 1

Python Basic

1

## Variables

What are variables?

- Variables hold a piece of information that can change over time.
- They can hold numbers, texts, and Booleans

- A variable name can contain letters, number, underscore but cannot start with a number.

## Variables: Example

- Create a variable that represents how much money in our wallet.
- Try this:

```
wallet = 41
print(wallet)
```

`wallet` is the variable name.

`print(wallet)` is to display the information inside variable (wallet)

## Basic Datatypes in Python

- Variables can hold few data types:

| Datatypes | Description | Example | Conversion Function |
|---|---|---|---|
| Integers | Whole numbers | 1, 2, 256 | `int()` |
| Floats | Fractional numbers | 0.001, 0.5, 3.142 | `float()` |
| Booleans | Truth values | `True` or `False` | `bool()` |
| Strings | Groups/string of characters | "Cyberjaya" 'basketball' | `str()` |

## Numbers: ints and floats

- There are two number types: integers and floats
- Examples of integers:

```
day = 21
temp = -15
```

- Examples of floats:

```
weight = 65.5
height = 155.5
```

## Numbers: Arithmetic Operations

- Try this:

```
day = 21
temp = -15
weight = 65.5

print(3 + 6)
print(day + 3)
print(weight * 2)
print(temp - 5)
print(weight / 2)
```

## Arithmetic Operators (Example)

- Try this:

```
print("5 + 2 =", 5+2)
print("5 - 2 =", 5-2)
print("5 * 2 =", 5*2)
print("5 / 2 =", 5/2)
print("5 % 2 =", 5%2)
print("5 ** 2 =", 5**2)
print("5 // 2 =", 5//2)
```

## Strings

- Strings – a way to represent text inside of Python

- Try this:

```
plant = 'mango'
plant = "mango"
print(plant)
```

- We can use a single quote (`'`) or a double quote (`"`) to represent a string.

## Strings: Using variables in strings

- How to print variables in strings? Use f symbol
- Try this:

```
day = 31
month = "March"
temp = 33
print(f"Today is {month} {day} and it’s {temp} degrees
outside")
```

- Try to change the variable values.

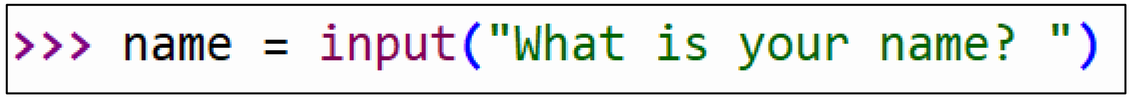
## Console Input and Output

- Console is the interface we interact with
    - Command prompt (Windows)
    - Terminal (Linux)
    - IDLE Shell (Python interactive mode)

- To interact, there must be input and output:
    - Output: Use the print () function
    - Input: Use the input () function

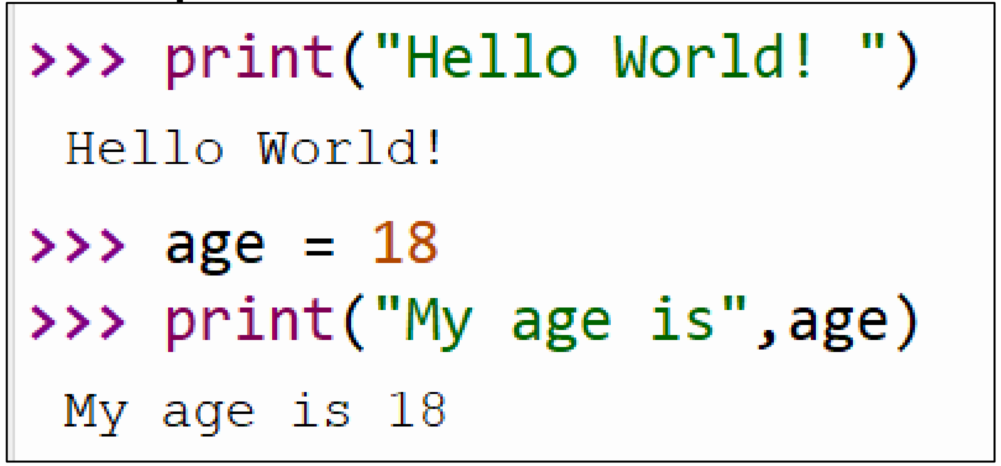
# input() and print()

- Input



```
>>> name = input("What is your name? ")
```

- Output



```
>>> print("Hello World! ")
 Hello World!
>>> age = 18
>>> print("My age is",age)
 My age is 18
```

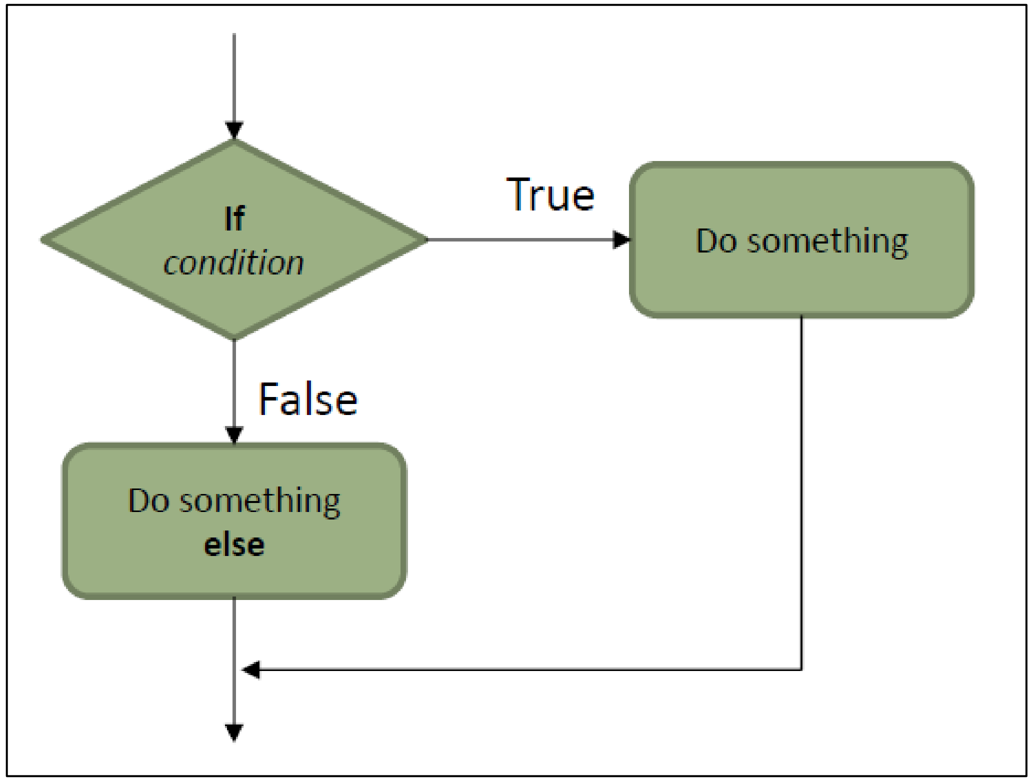
# Chapter 2

If, elif and else

2

## if and else

- Simulates **cause and effect**;
  - **If** a condition is true, do something
  - Otherwise do something **else**



# if and else (Example)

- Using conditional operators for checking conditions

```
>>> if num > 0:
        print(“Positive number”)
    else:
        print(“Negative number”)
```

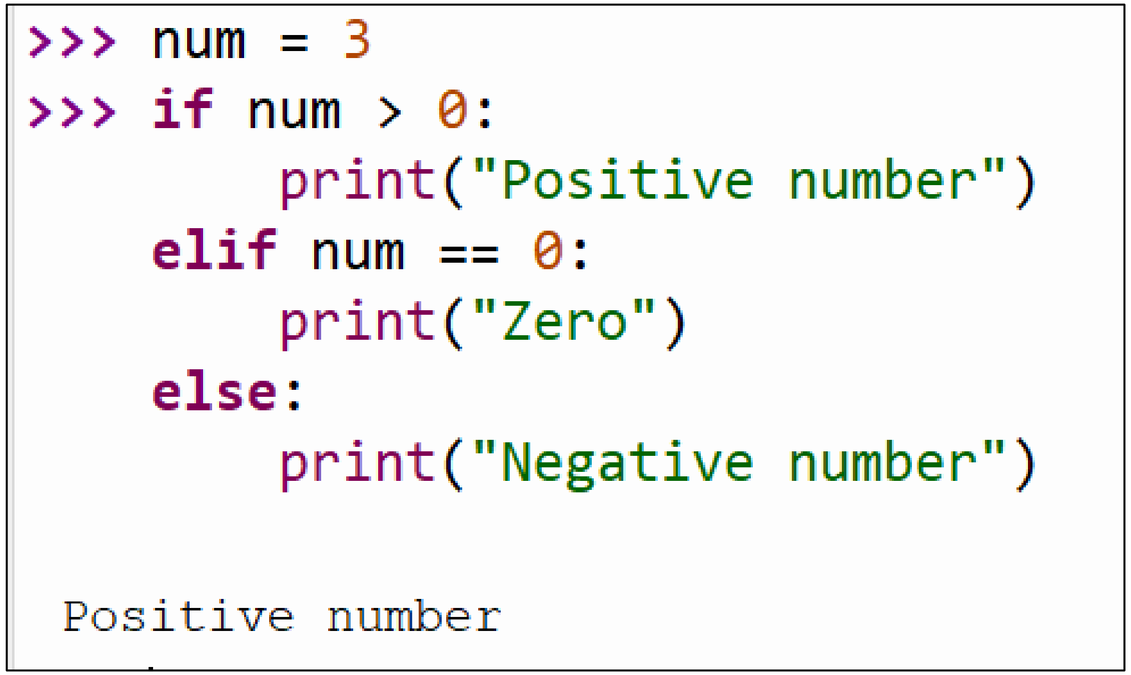
Colon (:) to indicate body of statements

- Using in or is

```
>>> fruits = [“apple”, “banana”, “pear”]
>>> if “banana” in fruits:
        print(“Yes")
    else:
        print("No")
```

## if, elif and else

- If there are more than two conditions, use elif in addition to if and else.



```
>>> num = 3
>>> if num > 0:
        print("Positive number")
    elif num == 0:
        print("Zero")
    else:
        print("Negative number")


 Positive number
```

## Indentation

- Blocks are one or more consecutive lines that form a single unit.

- Other languages (C, C++, C#, Java) require curly braces {} to indicate the beginning and the end of a block

- Python uses **indentation** to create blocks

```
>>> if num > 0:
        print(“Positive number”)
        print(“Not negative”)
```

# Chapter 2

Project: Magic 8 ball

2

## Picking a random numbers

- How to generate a random number?
- Try this:

```
import random
r = random.randint(1,10)
print(r)
```

- Note:
- Need to import random modules in Python
- `randint(a,b)` – generate random numbers between 1-10.

## Challenge: Magic 8 ball

- Make your own version of a magic 8 ball that prints yes, no or maybe each time you ask it.

```
import random
answer = random.randint(1,3)
if answer == 1:
    print("Yes")
elif answer == 2:
    print("No")
else:
    print("Maybe")
```

## Generate a lucky number

- Generate a luck number and display it.
- Try this:

```
import random
lucky_number = random.randint(1,100)
```

- Challenge: Try to print out the lucky number in this string

You will have a great day! Your Lucky Number is ….

## Choosing what fortune to show

```
import random
fortune_number = random.randint(1,3)

if fortune_number == 1:
     fortune_text = 'You will have a great day!'
if fortune_number == 2:
     fortune_text = 'Today will be tough...but worth
it.'
if fortune_number == 3:
     fortune_text = ‘Believe in yourself, for others
already do'
```

## Choosing what fortune to show (cont..)

- #How to print the fortune text and lucky number?

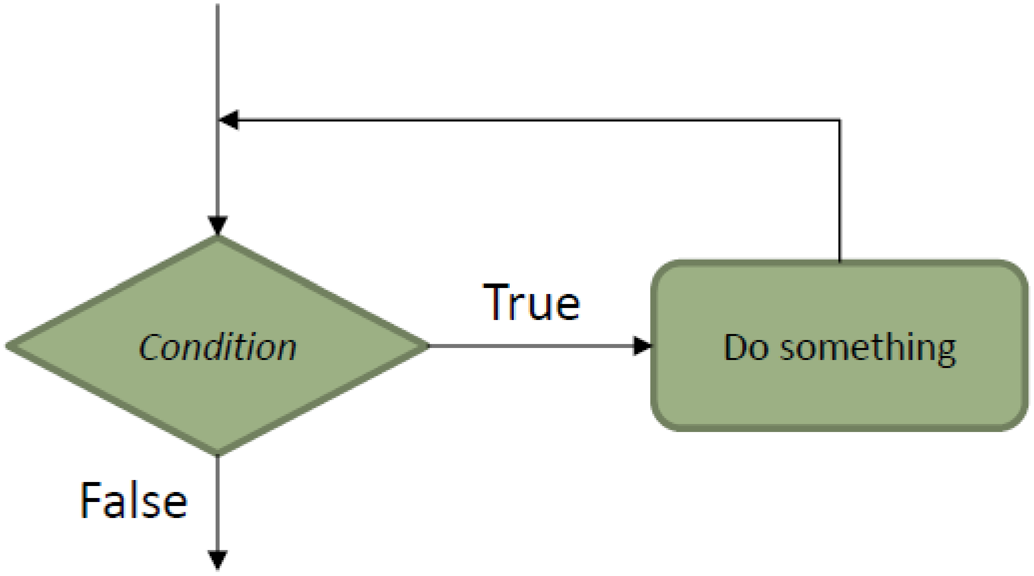
# Chapter 2

Repetition Loops

2

## Repetitions using Loops

- Loops allow some coding to be run repeatedly.



## For loops

- For loop – to repeat the same code several times
- Try this:

```
for number in range (10):
    print("Hello")
for num in range (10):
    print(num)
```

- Observe your output.

## For loops - challenge

- #Loop 20 times and print the number of loop times 2. Eg: 2,4,6,8....

```
for number in range(20):
    #complete your code here
```

## Introducing ‘While’ loop

- Use `for` loop when we know the number of iterations
- Use `if` when we need to make a choice
- `while` loop is combination of `for` and `if`
- Use `while` loop when we’re not sure about the number of iterations

## Guess a number

- Let the user guess a number. Reply whether the guess is correct or not, the correct number is higher or lower than the guess.
- When the guess is correct, inform the number of attempts that have been made.

## Try this

```
guess = int(input("What is your guess? "))
correct_number = 5

while guess != correct_number:
    guess = int(input("What is your guess? "))

print("You've made the right guess!")
```

## Can you count the number of attempts?

```
guess = int(input("What is your guess? "))
correct_number = 5
count = 1

while guess != correct_number:
    count += 1
    guess = int(input("What is your guess? "))

print(f"You've made the right guess! The right 
answer is {correct_number}. It took you {count} 
guesses.")
```

## What if the correct number is randomly selected?

- Problem: The correct number is fixed as 5
- Challenge: Can you make the correct number an unknown, randomly selected number?
- Hint: You need to use a random number generator

## Possible solution

```
import random
correct_number = random.randint(1, 100)
print("Guess between 1 and 100")
guess = int(input("What is your guess? "))
count = 1

while guess != correct_number:
    count += 1
    if guess < correct_number:
        guess = int(input("Nope. Guess higher, please > "))
    elif guess > correct_number:
        guess = int(input("Nope. Guess lower, please > "))

print(f"You've made the right guess! The right answer is 
{correct_number}. It took you {count} guesses.")
```

# Chapter 3

Functions

3

## What is a function?

- A function is a group of code that is referred to by name and solves a specific task. For example:
    - print() : display on console
    - sqrt() : Calculate the square root
    - len(): Determine the length of object

- Syntax:

```
def function_name(parameters):
    statement(s)
```

## Calling Functions

- Instead of printing the result, the function can return a value.

```
def square(x):
        answer = x*x
        return answer

>>> result = square(4)
>>> result
16
```

- Functions can also return a list

```
def duplicate(x):
        answer = x*2
        return answer

>>> double = duplicate([1, 2, 3])
>>> double
[1, 2, 3, 1, 2, 3]
```

## Define a function

```
def hello():
    print("Hello World!")
```

- How to use the function?

## Call the function

```
hello()
```

## Pass Parameters to a Function

```
def hello(name):
    print(f"Hello {name}!")

hello("John")
```

## How many parameters can there be?

- As many as you want

- An example of 2-parameter function:

```
def add_numbers(num1, num2):
    print(num1 + num2)

add_numbers(5, 4)
```

## Functions - Example

Create a function that prints out a cat’s name & age

```
def cat(name, age):
    print(f"I'm a cat. My name is {name} and I'm {age} 
years old.")

cat("Kitty", 9)
```

## Output(s) of a function

- Parameters are the inputs of a function
- There can be outputs from a function
- How to capture those outputs?

## Return

```
def double(number):
    return number * 2

new_number = double(5)
print(new_number)
```

# Chapter 4

Classes and Instances

4

## Classes

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.
- The best way to understand the concept of Classes is through an example. First, can you google for the meaning of apple and write it down here?

## Classes

- The best way to understand the concept of Classes is through an example. First, can you google for the meaning of apple and write it down here?

## Instances

- Now let's check out another concept called Instances.
- Let's first add something to the Apple class.

# Chapter 5

Methods and Inheritances

5

## Methods

- Now we're going to teaching a concept called Methods.
- Actually you've just used two special methods, i.e __init__ and __str__.
- They're special in the sense that you don't call them explicitly. They just work in the background and are called automatically when needed by the program.

## Methods

- Now we teach you how to create Methods of your own (that you can call them explicitly).
- We know that an apple tree can grow from its seed, given the right conditions.
- Let's create a grow Method.

## Inheritances

- Now, we reach the last part of this lecture. We want to teach you a concept that is called Inheritance.

- We all know that apple is a type of fruit, so as banana.
- We can create a Fruit class, and then create the Apple and Banana classes that inherit from the Fruit class.
- Then whatever characteristics defined in the parent (Fruit) class will be passed down to the child (Apple, Banana) class.

CNERGY

THANK YOU

www.mmu-cnergy.com
enquiry.cnergy@mmu.edu.my
MMU Cnergy