SWINBURNE UNIVERSITY OF TECHNOLOGY

Python in a nutshell

Hung Luu
hluu@swin.edu.au

Melb 3 Aug 2023

Python Benefits

- Python is processed at runtime by the interpreter.
  - Other language like C, PHP, etc. require compilation into a complete program to be executed.
- Python responses to your programs in an interactive way
  - You can write Python line by line (same as MATLAB, IDL or R) instead of writing the full program for running.
- Python supports OOP
  - The mixture of code, types and objects
- Python good for data processing and machine learning applications
  - It is easy to use, with lots of libraries.

Python Variable

- Variable in Python is dynamic: its type and value can be changed throughout the program
  - It is contrasted with other language (e.g., C, Java) where data type are fixed.
  - No declaration is required before assign any value to variable

- Assignment is denoted by equal sign (=)
  - It is used to "store" value of type or instance of class/object

```
score = 1

score = [2,3,4]

score = ["one", "two", "three"]
```

Python Some rules for combination

- Use sum symbol （+） to concat two strings in Python
  - Examples: `"Donald " + "Trump"` → `"Donald Trump"`
- Can not concat directly a string and a number
  - Error: ~~`2 + "people"`~~
- We must convert number to string for concatenation
  - Examples: `str(2) + " people"` or `str(area)`
- One may convert a string to float or integer number
  - Examples: `int("3.14159")` or `int("365")`

Python Input/output to Terminal

- `print()` is a built-in function used to display content in Terminal/Command Prompt

```
print(1)
print("Student score",10)
```

- `input()` is a built-in function used get user's string input in Terminal/Command Prompt
  - In Python 2 or earlier, it is **`input_raw`**. From Python 3 onward, it is simply input.
  - We can assign its return to get value to be used later.
  - Its default return value is string. You can convert it to integer or float/real number.

```
name = input("What is your name?")
age = int(input("How old are you?"))
cost = float(input("How much does it cost?"))
```

Python Array with `list`

- Array in Python can be used to store numbers, strings, objects or mixture.
- Array in Python is defined by `list` where values by square brackets [ ]
- Example arrays of numbers and strings

```
array_start_empty = [ ]
my_number = [1, 2, 3, 4, 10]
shop_list = ["chicken", "pork", "beef"]
```

- Example of array with objects or mixture

```
my_mixed_list = [1, [2, 3], ["chicken"]]
```

Python Array with list

- To append value to a list

```
my_number = [1, 2, 3, 4, 10]
my_number.append(10)
```

- To access a value from a list (Note: Index in Python start from 0)

```
# my_number[0] = 1
# my_number[3] = 4
```

- To count total element in array, use `len()`

```
print(len(my_number))
```

Python Dictionary with dict

- Dictionary in Python can be used to store pairs of key and value associated with object.
- Dictionary in Python is created by curly brackets `{ }` or define `dict()`
- Dictionary in Python adds and extracts value by square brackets `[ ]`

```
english_japanese = { }
english_japanese["hashi"] = "bridge"
english_japanese["ichi"] = 1

print(english_japanese["ichi"])
```

Python Tuple

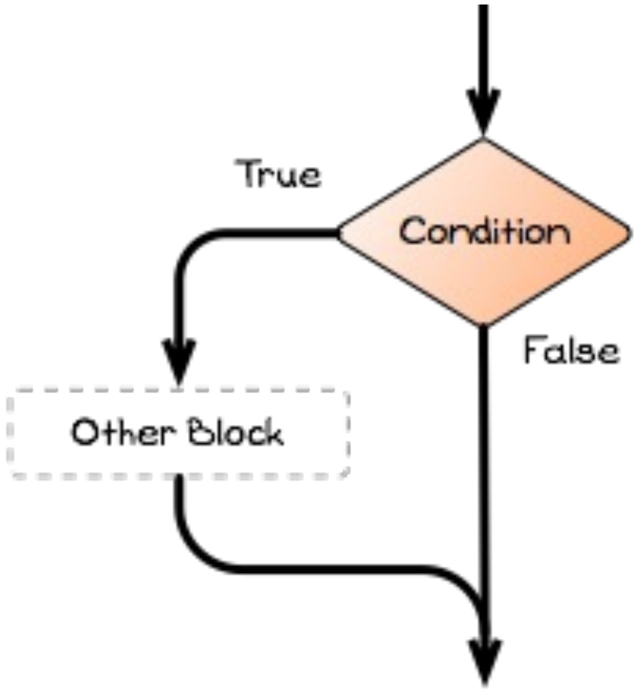
- A tuple is a collection of objects which ordered and immutable.
- Dictionary in Python is created by brackets ( )

```
info = ("Hung", "Luu", 2021, "Swinburne")

given_name, sur_name, year, school = info

print(given_ name)
```

Python if branching

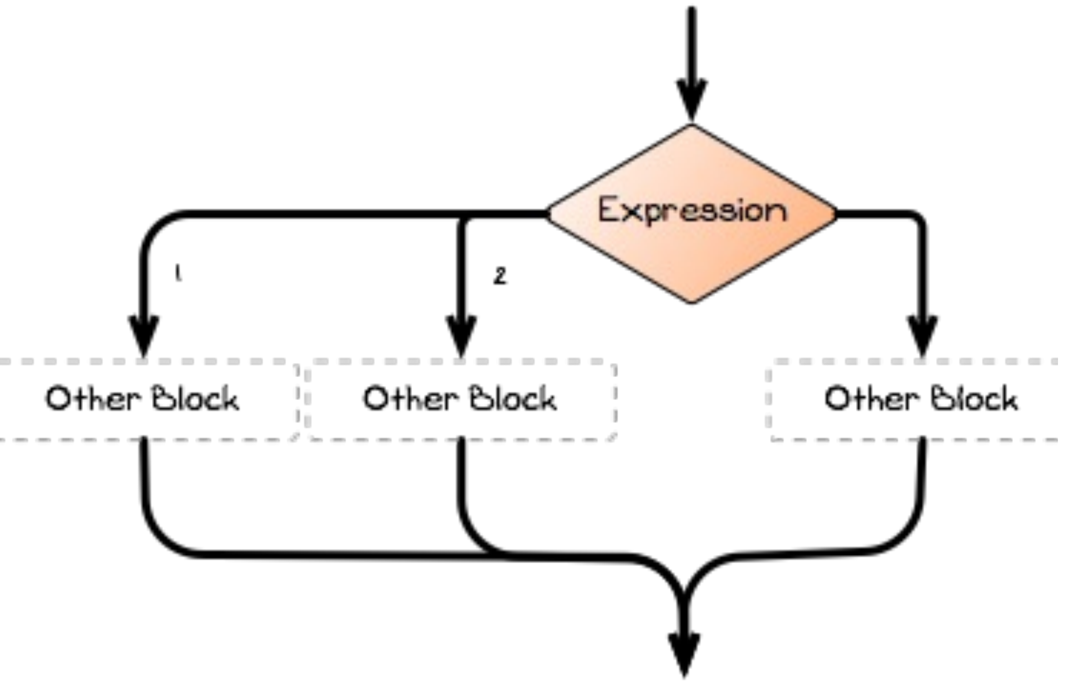


```
if <true/false condition> :
      <what's to do>
else:
      <what's to do otherwise>
```

Example

```
print("Me enjoying the party...")
caller = input("Some one calling…")
if ( caller == "girlfriend"):
      print("Sorry guys! I must return home!")
      print("My house is on fire." )
else:
      print("Hey, come and join us!")
```

Python if multiple branching

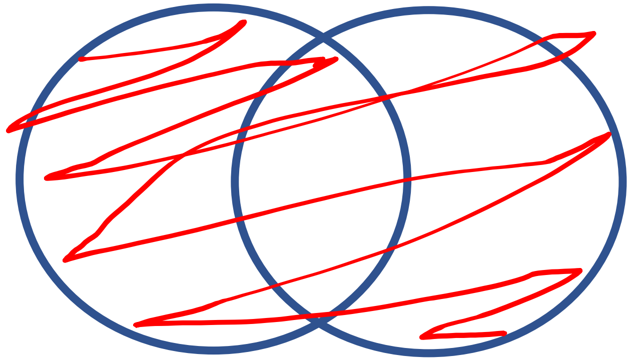


Syntax

```
if <true/false condition A> :
    <what's to do in case A>
elif <true/false condition B> :
    <what's to do in case B>
elif <true/false condition C> :
    <what's to do in case C>
elif <true/false condition D> :
    <what's to do in case D>
else:
    <what's to do otherwise>
```

Python or logic operator

a or b



handsome? or rich?

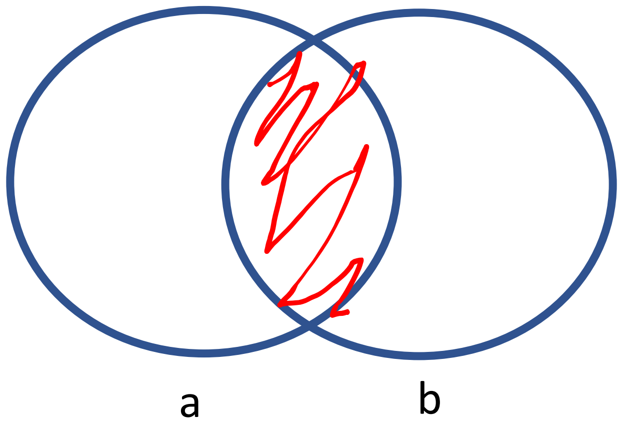
```
if <true/false condition A> or <true/false condition B> :
     <what's to do in case of A or B>
else:
     <what's to do otherwise>
```

Example

```
print("Me enjoying the party...")
caller = input("Some one calling…")
if ( caller == "girlfriend" ) or ( caller == "mum" ):
     print("No signal found!")
else:
     print("Hey, come and join us!")
```

Python and logic operator

a and b



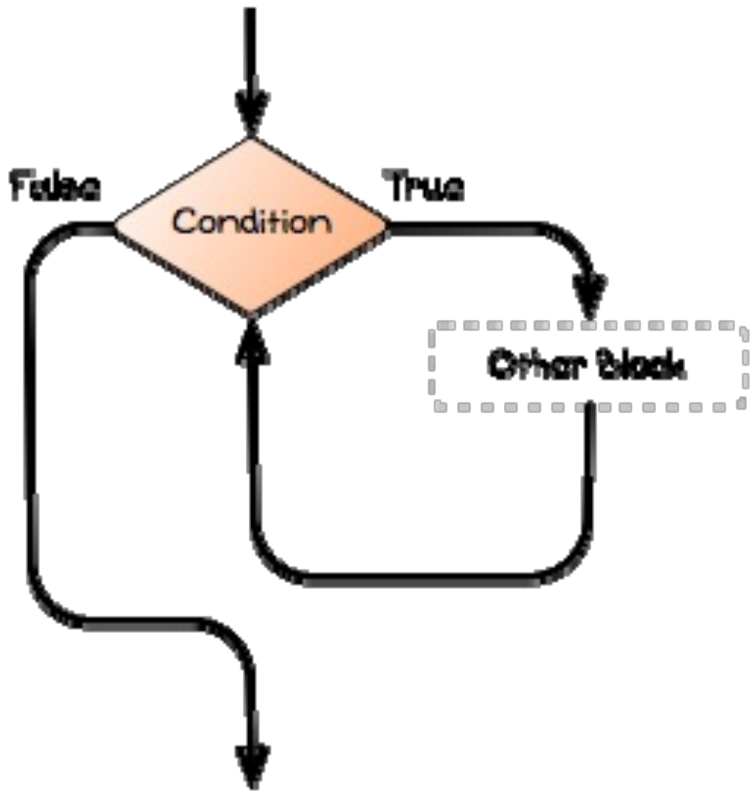
handsome? and rich?

```
if <true/false condition A> and <true/false condition B> :
     <what's to do in case of A and B>
else:
     <what's to do otherwise>
```

Example: What's wrong here?

```
print("Me enjoying the party...")
caller = input("Some one calling…")
if ( caller == "girlfriend" ) and ( caller == "mum" ):
     print("Never happen!")
else:
     print("Hey, come and join us!")
```

Python for loop with list

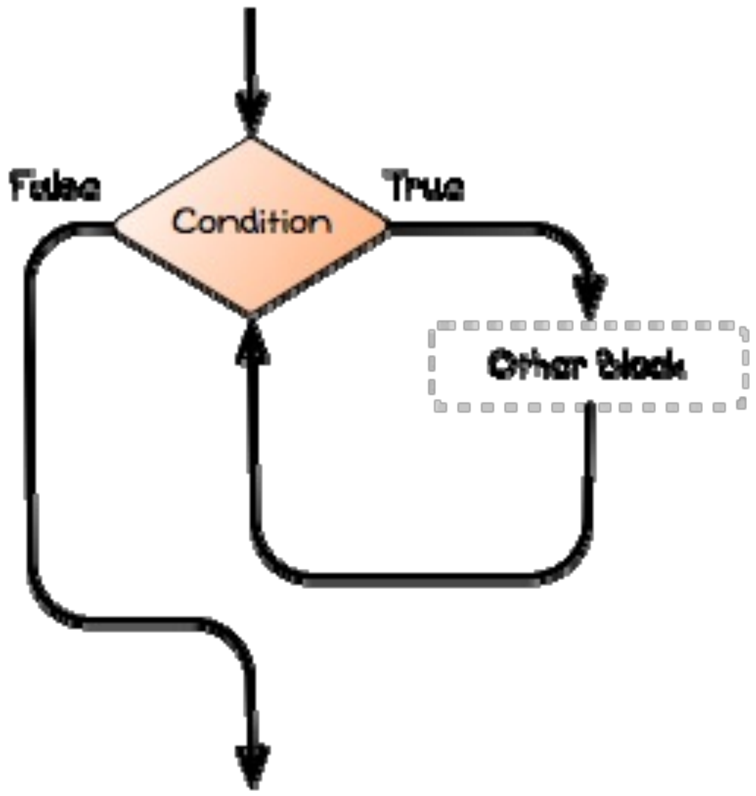


```
for <variable> in [<value1>, <value2>, <value3>]:
    <what's to do>
```

Example

```
mylist = ["chicken", "pork", "beef"]
for thing in mylist:
    print( "I need to buy " + thing)
```

Python for loop with enumerate



```
for <index>,<variable> in enumerate(<list>):
   <what's to do>
```

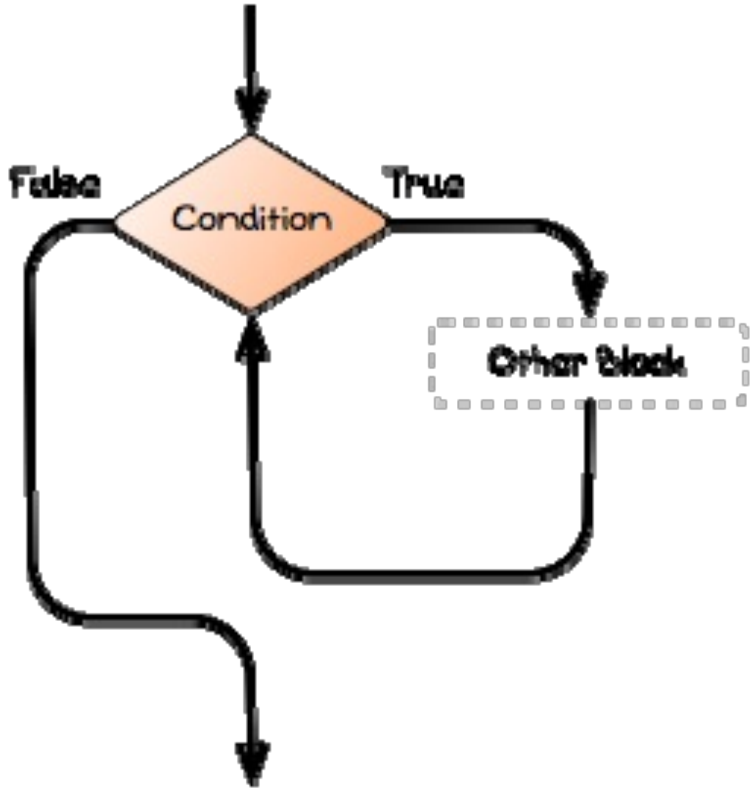
Example

```
things = ["chicken", "pork", "beef"]

for i,thing in enumerate(things):
   print( str(i) + ": I need to buy " + thing)

#0: I need to buy chicken
#1: I need to buy pork
#2: I need to buy beef
```

Python for loop with range



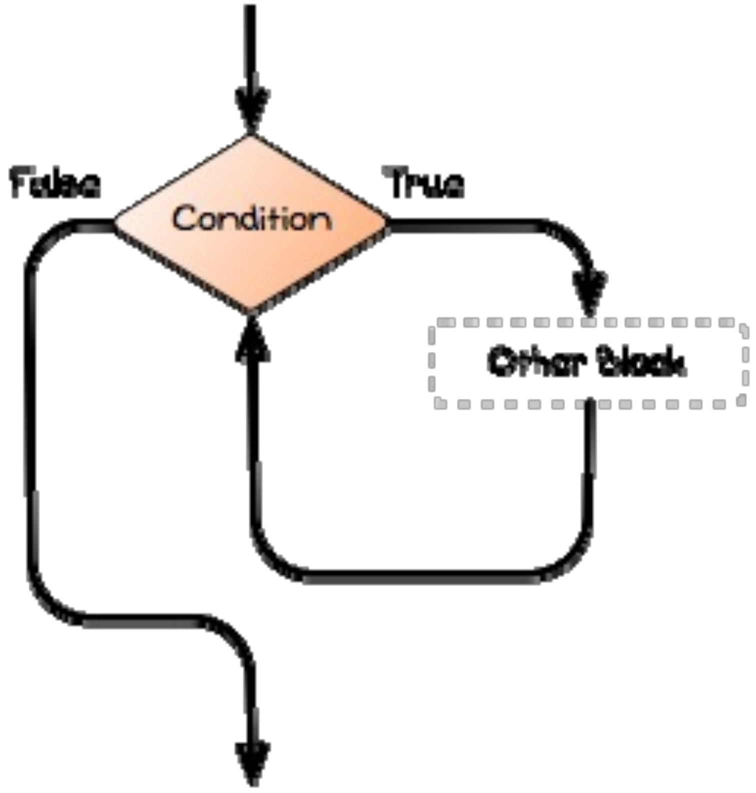
```
for <number> in range(<value_min>,<value_max>):
      <what's to do>
```

Example

```
for counter in range(1,12):
     print("Month {0} of the year.".format(counter))

# Month 1 of the year
# Month 2 of the year
#…
```

Python for loop with combined lists



```
for value1,value2 in zip(list1,list2):
   <what's to do>
```

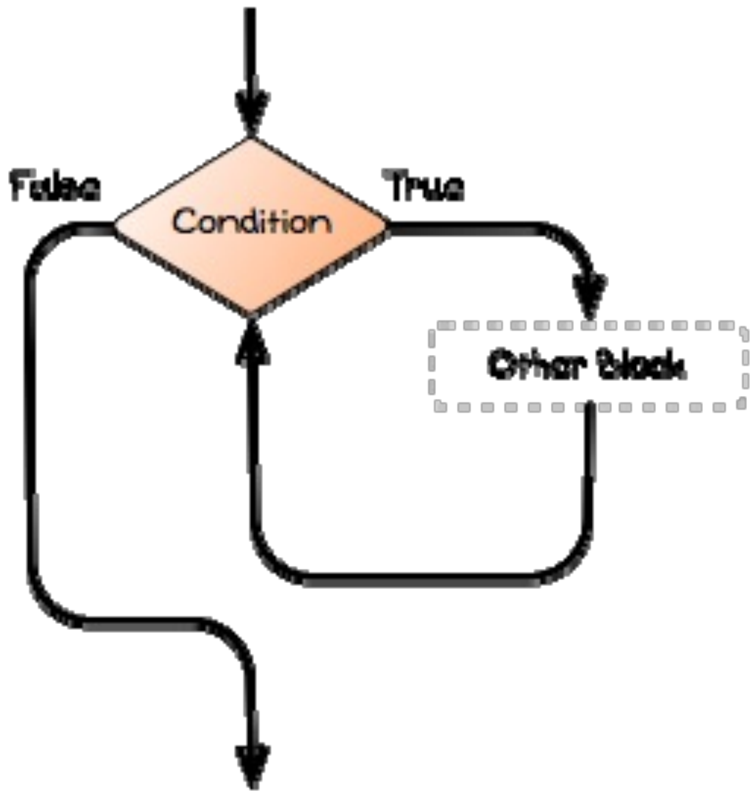
Example

```
things = ["chicken", "pork", "beef]
prices = [10,20,30]

for thing,price in zip(things,prices)
   print("I need to buy " + thing)
   print(price)

# I need to buy chicken \n 10
# I need to buy pork \n 20
# I need to buy beef \n 30
```

Python while loop



```
<1: variable assignment>
while <2: condition of variable>:
      <3: variable changes>
      <what's to do>
```

Example

```
counter = 1
while counter < 10:
      counter = counter + 1
      print("I wont do it again"+ str(counter))
```

Python while vs. for loop

- while loop
  - Require a variable (e.g., *counter*) before loop.
  - Variable is changed inside the loop's body.
  - Before use: check **logic condition**.

```
counter = 1
while counter < 10:
      counter = counter + 1
      print("Loop "+ str(counter))
```

- for loop
  - **Not** require an index (e.g., *counter*) before loop.
  - **No** manual change of variable in the loop's body.
  - Before use: check variable **range**.

```

for counter in 1..10:

      print("Loop "+ str(counter))
```

Python Read and write data file

- Open a file and writing data to it

```
my_file = open("mydata.txt","w")
my_file.write("This is the first line.\n")
my_file.write("This is the second line.\n")
my_file.close()
```

- Open a file and reading data from it

```
my_file = open("mydata.txt","r")
line1 = my_file.readline
line2 = my_file.readline
print(line1)  # this is to display in Terminal window
print(line2)  # this is to display in Terminal window
my_file.close()
```

Python Write data file with loop

- Write an array of data to file with `for` loop

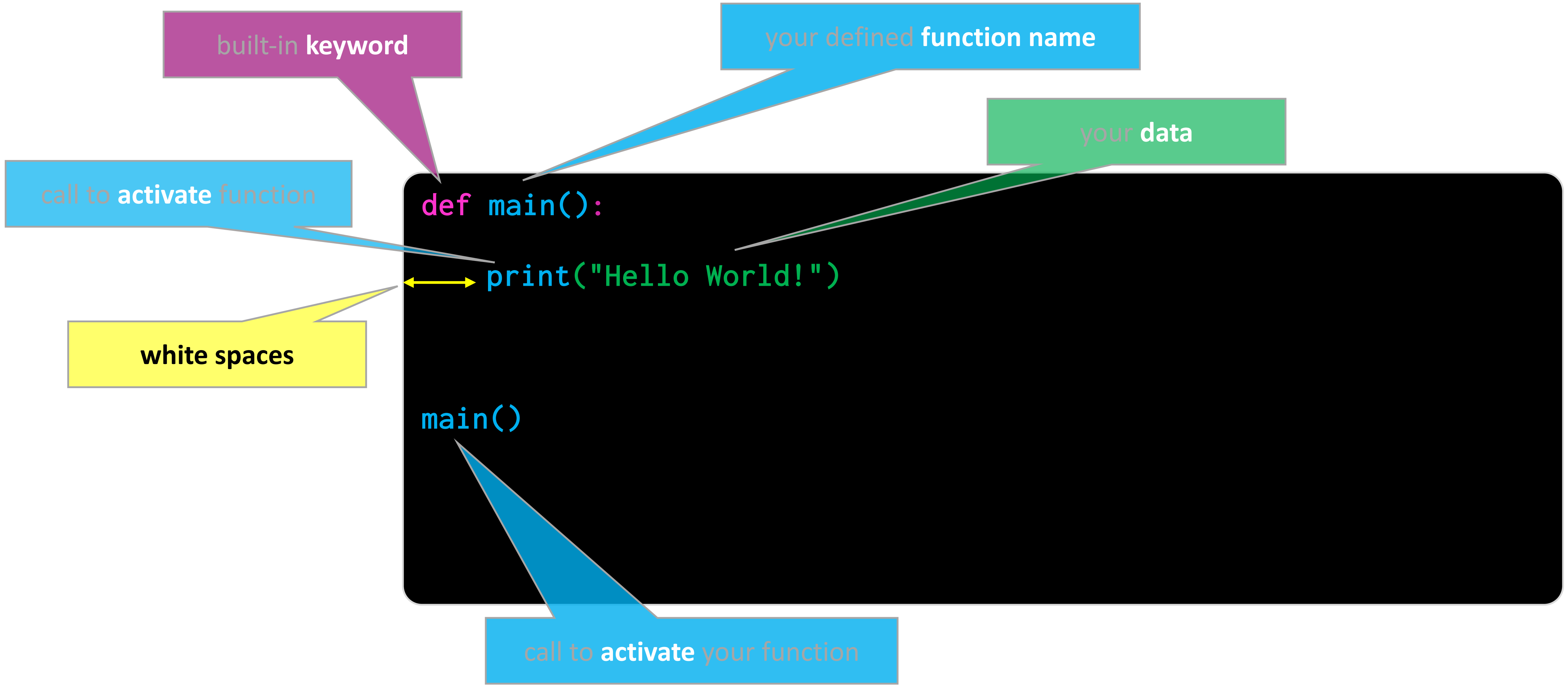
```
foods = ["chicken", "pork", "beef"]

my_file = open("mydata.txt","w+")

for food in foods:
      my_file.write("I need to buy " + food + "\n.")

my_file.close()
```

Python Function


built-in keyword
your defined function name
your data
call to activate function
white spaces
call to activate your function

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


main()
```

Python Function parameters

```
def greeting (name, age):
   print("Hello", name,"who is", str(age))


greeting("Peter",30)
```

- To call a function, we must give it
  - The **same number** of parameters as definition
  - The correct **order** of each parameter
  - The correct **data type** of parameters
- We can set defaults for parameters

```
greeting("Peter")
```

✗

```
greeting(30, "Peter")
```

✗

```
greeting("30", "Peter")
```

✗

```
def greeting (name, age = 28):
```

Python Class initialization

- Example: create a class and print something

```
class Dog:
    def __init__(): # This function will be activated an instance of class is created
        print("Hung")

mydog = Dog()

# Hung
```

Python Class with attributes

- Example: create a class with an attribute

```
class Dog:

    def __init__():

        self.name = "Hung" # Create an attribute so-called name



mydog = Dog()

print(mydog.name)

# Hung
```

Python Class with methods

- Example: create a class with a method

```
class Dog:

    def __init__():

        self.name = "Hung"

    def say_it(): # Create a method to do something

        print(self.name)


mydog = Dog()

mydog.say_it()

# Hung
```

Python Class with input and called method

- Example: create a class with a customized input and called method

```
class Dog:

    def __init__(your_name):

        self.name = your_name

        self.say_it()

    def say_it():

        print(self.name)


mydog = Dog("Hung")

# Hung
```