TAPIT
BUILD THE FUTURE WITH US!

Computer Vision

Lecture 1: Python

TAPIT.RGB

CuBoBKDN
bo.phamconganhhuy@gmail.com
tapit.vn

TAPIT.RGB

What is Python?

- A general-purpose programming language which can be used for a wide variety of applications
- Fourth Generation Language
- An interpreted language
- Python has an enormous user community

TAPIT.RGB

Hello, World!

Example 1:

```
print('Hello, World!')
```

```
Hello, World!
```

Example 2:

```
x = 1
if x == 1:
    print('Hello, World!')
```

```
Hello, World!
```

TAPIT.RGB

Variables and Types
Numbers: Integer - Float

To define an integer:

```
myInt = 7
print(myInt)
print(type(myInt))
```

```
7
<class 'int'>
```

To define a floating point number:

```
myfloat = float(myInt)
print(myfloat)
print(type(myfloat))
```

```
7.0
<class 'float'>
```

Variables and Types

Strings

```
mystring = 'hello'
print(mystring)
mystring = "Hello"
print(mystring)
```

```
hello
Hello
```

```
one = 1
two = 2
three = one + two
print(three)
hello = 'hello'
world = "world"
helloworld = hello + " " + world
print(helloworld)
```

```
3
hello world
```

```
# This will not work!!!
one = 1
two = 2
hello = 'hello'
print(one + two + hello)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-a44a01d8f212> in <module>
      3 two = 2
      4 hello = 'hello'
----> 5 print(one + two + hello)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

TAPIT.RGB

Exercises 1.1

```
# change this code
mystring = None
myfloat = None
myint = None

# testing code
if mystring == "hello":
    print("String: ", mystring)
if isinstance(myfloat, float) and myfloat == 10.0:
    print("Float: ", myfloat)
if isinstance(myint, int) and myint == 20:
    print("Integer: ", myint)
```

Lists

```
list1 = ['vatly', 'hoahoc', 1997, 2000, 69.96]
print(list1)
print(list1[2])
```

```
['vatly', 'hoahoc', 1997, 2000, 69.96]
1997
```

```
del list1[3]
print(list1)
```

```
['vatly', 'hoahoc', 1997, 69.96]
```

```
mylist = []
mylist.append(1)
mylist.append(3.0)
print(mylist)
mylist.append('ahihi')
print(mylist)
```

```
[1, 3.0]
[1, 3.0, 'ahihi']
```

TAPIT.RGB

Exercises 1.2

```
numbers = []
strings = []
names = ["John", "Eric", "Jessica"]
# write your code here
second_name = None

# this code should write out the filled arrays
# and the second name in the names list (Eric).
print(numbers)
print(strings)
print("The second name on the names list is", second_name)
```

Basic Operators

Arithmetic Operators:

```
number = 1 + 2 * 3 / 4.0
print(number)
```

```
2.5
```

```
remainder = 11 % 3
print(remainder)
```

```
2
```

```
squared = 7 ** 2
cubed = 2 ** 3
print(squared, cubed)
```

```
49 8
```

Operators with Strings and Lists:

```
helloworld = 'hello' + ' ' + 'world'
print(helloworld)
```

```
hello world
```

```
lotsofhellos = 'hello' * 10
print(lotsofhellos)
```

```
hellohellohellohellohellohellohellohellohellohello
```

```
even = [2, 4, 6, 8]
odd = [1, 3, 5, 7]
print(odd + even)
```

```
[1, 3, 5, 7, 2, 4, 6, 8]
```

```
print([1, 2, 3] * 3)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

TAPIT.RGB

Exercises 1.3

The target of this exercise is to create two lists called x_list and y_list, which contain 10 instances of the variables x and y, respectively. You are also required to create a list called big_list, which contains the variables x and y, 10 times each, by concatenating the two lists you have created

```
x = object()
y = object()

# TODO: change this code
x_list = [x]
y_list = [y]
big_list = []

print("x_list contains ", len(x_list), " objects")
print("y_list contains ", len(y_list), " objects")
print("big_list contains ", len(big_list), " objects")

# testing code
if x_list.count(x) == 10 and y_list.count(y) == 10:
    print("Almost there...")
if big_list.count(x) == 10 and big_list.count(y) == 10:
    print("Great!")
```

String Formatting

```
name = 'Anh Huy'
print('Hello, %s!' % name)
```

```
Hello, Anh Huy!
```

```
name = 'Anh Huy'
age = 23
print('%s is %d years old!' % (name,age))
```

```
Anh Huy is 23 years old!
```

```
mylist = [1,2,3]
print('A list is %s' % mylist)
```

```
A list is [1, 2, 3]
```

%s – String (or any obj with a string representation, like numbers)

%d – Intergers

%f – Floating point numbers

%.<number of digits>f – Float point numbers with a fixed amount of digits to the right of the dot.

%x/%X – Intergers in hex representation (lowercase / uppdercase)

TAPIT.RGB

Exercises 1.4

You will need to write a format string which prints out the data using the following syntax: ***Hello John Doe. Your current balance is $53.44.***

```
# TODO: change this code
data = ("John", "Doe", 53.44)
format_string = "Hello"

# testing code
print(format_string % data)
```

Basic String Operations

```
astring = "Hello world!"
print("single quotes are ' '")
print(len(astring))
```

```
single quotes are ' '
12
```

```
astring = "Hello world!"
print(astring.index("o"))
```

```
4
```

```
astring = "Hello world!"
print(astring.count("l"))
```

```
3
```

```
astring = "Hello world!"
print(astring[3:7])
```

```
lo w
```

```
astring = "Hello world!"
print(astring[3::2])
```

```
l ol!
```

```
astring = "Hello world!"
print(astring[::-1])
```

```
!dlrow olleH
```

```
astring = "Hello world!"
print(astring.upper())
print(astring.lower())
```

```
HELLO WORLD!
hello world!
```

```
astring = "Hello world!"
print(astring.startswith("He"))
print(astring.endswith("asd"))
```

```
True
False
```

```
astring = "Hello world!"
print(astring.split(" "))
```

```
['Hello', 'world!']
```

```
astring = ['Hello', 'world!']
print(','.join(astring))
```

```
Hello,world!
```

TAPIT.RGB

Exercises 1.5

Try to fix the code to print out the correct information by changing the string.

```
# TODO: change this code
s = "Hey there! what should this string be?"

# testing code
# Length should be 20
print("Length of s = %d" % len(s))

# First occurrence of "a" should be at index 8
print("The first occurrence of the letter a = %d" % s.index("a"))

# Number of a's should be 2
print("a occurs %d times" % s.count("a"))

# Slicing the string into bits
print("The first five characters are '%s'" % s[:5]) # Start to 5
print("The next five characters are '%s'" % s[5:10]) # 5 to 10
print("The thirteenth character is '%s'" % s[12]) # Just number 12
print("The characters with odd index are '%s'" %s[1::2]) #(0-based indexing)
print("The last five characters are '%s'" % s[-5:]) # 5th-from-last to end
```

TAPIT.RGB

Exercises 1.5

Try to fix the code to print out the correct information by changing the string.

```
# Convert everything to uppercase
print("String in uppercase: %s" % s.upper())

# Convert everything to lowercase
print("String in lowercase: %s" % s.lower())

# Check how a string starts
if s.startswith("Str"):
    print("String starts with 'Str'. Good!")

# Check how a string ends
if s.endswith("ome!"):
    print("String ends with 'ome!'. Good!")

# Split the string into three separate strings, each containing only a word
print("Split the words of the string: %s" % s.split(" "))
```

Conditions

Boolean Operators:

```
x = 2
print(x == 2)
print(x >= 3)
print(x < 3)
```

```
True
False
True
```

```
print(True and True)
print(True and False)
print(True or False)
print(not True)
```

```
True
False
True
False
```

```
x = 2
if x == 2:
    print("x equals two!")
elif x == 3:
    print("x equals three!")
else:
    print("x does not equal to two.")
```

```
x equals two!
```

The "in" Operator:

```
name = "John"
if name in ["John", "Rick"]:
    print("Your name is either John or Rick.")
```

```
Your name is either John or Rick.
```

The "is" Operator:

```
x = [1,2,3]
y = [1,2,3]

print(x == y)
print('---------')
print(x is y)
print(id(x), id(y))
print('---------')
x = y
print(x is y)
print(id(x), id(y))
```

```
True
---------
False
2837811493704 2837811493000
---------
True
2837811493000 2837811493000
```

TAPIT.RGB

Exercises 1.6

Change the variables in the first section, so that each if statement resolves as True.

```
# TODO: change this code
number = 10
second_number = 10
first_array = []
second_array = [1,2,3]
# testing code
if number > 15:
    print("1")

if first_array:
    print("2")
```

```
if len(second_array) == 2:
    print("3")

if len(first_array) + len(second_array) == 5:
    print("4")

if first_array and first_array[0] == 1:
    print("5")

if not second_number:
    print("6")
```

Loops

The “for” loops

```
primes = [2, 3, 5, 7]
for prime in primes:
    print(prime)
```

```
2
3
5
7
```

The “in” Operator

```
for x in range(5):
    print(x)
print('-')
for x in range(3,6):
    print(x)
print('-')
for x in range(3,8,2):
    print(x)
```

```
0
1
2
3
4
-
3
4
5
-
3
5
7
```

The “while” loops

```
count = 0
while count < 5:
    print(count)
    count += 1
```

```
0
1
2
3
4
```

```
count=0
while(count<5):
    print(count)
    count +=1
else:
    print('End')
```

```
0
1
2
3
4
End
```

“break”/“continue” statement

```
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
```

```
0
1
2
3
4
```

```
for x in range(10):
    if x % 2 == 0:
        continue
    print(x)
```

```
1
3
5
7
9
```

TAPIT.RGB

Exercises 1.7

Loop through and print out all even numbers from the numbers list in the same order they are received. Don't print any numbers that come after 237 in the sequence

```
numbers = [951, 402, 984, 651, 360, 69, 408, 319, 601, 485, 980,
           507, 725, 547, 544, 615, 83, 165, 141, 501, 263, 617,
           865, 575, 219, 390, 984, 592, 236, 105, 942, 941, 386,
           462, 47, 418, 907, 344, 236, 375, 823, 566, 597, 978,
           328, 615, 953, 345, 399, 162, 758, 219, 918, 237, 412,
           566, 826, 248, 866, 950, 626, 949, 687, 217,  815, 67,
           104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831,
           445, 742, 717, 958, 609, 842, 451, 688, 753, 854, 685,
           93, 857, 440, 380, 126, 721, 328, 753, 470, 743, 527]

# TODO: change this code
```

Functions

```
def my_function():
    print("Hello From My Function!")

my_function()
```

```
Hello From My Function!
```

```
def sum_two_numbers(a, b):
    return a + b

print(sum_two_numbers(1, 68))
```

```
69
```

```
def my_function_with_args(username, greeting):
    print("Hello, %s , From My Function!, I wish you %s"%(username, greeting))

my_function_with_args('Anh Huy', 'get handsome')
```

```
Hello, Anh Huy , From My Function!, I wish you get handsome
```

TAPIT.RGB

Exercises 1.8

In this exercise you'll use an existing function, and while adding your own to create a fully functional program.

1. Add a function named list_benefits() that returns the following list of strings: "More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect code together"
2. Add a function named build_sentence(info) which receives a single argument containing a string and returns a sentence starting with the given string and ending with the string " is a benefit of functions!"
3. Run and see all the functions work together!

TAPIT.RGB

Exercises 1.8

```
# TODO: change this code
def list_benefits():
    pass

def build_sentence(benefit):
    pass

# testing code
def name_the_benefits_of_functions():
    list_of_benefits = list_benefits()
    for benefit in list_of_benefits:
        print(build_sentence(benefit))

name_the_benefits_of_functions()
```

Classes and Object

```
class MyClass1:
    variable = "blah"

    def function(self):
        print("This is a message inside the class.")

myobject = MyClass1()
```

```
class MyClass2:
    def __init__(self):
        self.variable = "blah"

    def function(self):
        print("This is a message inside the class.")

myobject = MyClass2()
```

Accessing Object Variables

```
print(myobject.variable)

myobject.variable = 'yackity'
print(myobject.variable)
```

```
blah
yackity
```

Accessing Object Functions

```
myobject.function()
```

```
This is a message inside the class.
```

TAPIT.RGB

Exercises 1.9

We have a class defined for vehicles. Create two new vehicles called **car1** and **car2**. Set car1 to be a **red convertible** worth **$60,000.00** with a name of **Fer**, and **car2** to be a **blue van** named **Jump** worth **$10,000.00**.

```
class Vehicle:
    name = ""
    kind = "car"
    color = ""
    value = 100.00
    def description(self):
        desc_str = "%s is a %s %s worth $%.2f." % (self.name, self.color, self.kind, self.value)
        return desc_str

# your code goes here


# test code
print(car1.description())
print(car2.description())
```

Dictionaries

```
phonebook = {}
phonebook["John"] = 938477566
phonebook["Jack"] = 938377264
phonebook["Jill"] = 947662781
print(phonebook)
```

```
{'John': 938477566, 'Jack': 938377264, 'Jill': 947662781}
```

```
phonebook = {
    "John" : 938477566,
    "Jack" : 938377264,
    "Jill" : 947662781}
print(phonebook)
```

```
{'John': 938477566, 'Jack': 938377264, 'Jill': 947662781}
```

Iterating over dictionaries

```
phonebook = {"John" : 938477566,"Jack" : 938377264,"Jill" : 947662781}
for name, number in phonebook.items():
    print("Phone number of %s is %d" % (name, number))
```

```
Phone number of John is 938477566
Phone number of Jack is 938377264
Phone number of Jill is 947662781
```

Remove a value

```
del phonebook["John"]
print(phonebook)
```

```
{'Jack': 938377264, 'Jill': 947662781}
```

or

```
phonebook.pop("John")
print(phonebook)
```

```
{'Jack': 938377264, 'Jill': 947662781}
```

TAPIT.RGB

Exercises 1.10

Add "Jake" to the phonebook with the phone number **938273443**, and **remove Jill** from the phonebook.

```
phonebook = {
    "John" : 938477566,
    "Jack" : 938377264,
    "Jill" : 947662781
}
# write your code here

# testing code
if "Jake" in phonebook:
    print("Jake is listed in the phonebook.")
if "Jill" not in phonebook:
    print("Jill is not listed in the phonebook.")
```

Modules and Packages

Modules

```
import module1[, module2[,... moduleN]
```

md1.py

```
def add(a,b):
    print(a+b)

def circle(r):
    print(3.14*r*r)

def square(l):
    print(l*l)

def rectangle(l,b):
    print(l*b)

def triangle(b,h):
    print(0.5*b*h)
```

```
import md1
md1.add(10,20)
md1.circle(30)
```

```
30
2826.0
```

```
from md1 import *
add(10,20)
circle(30)
```

```
30
2826.0
```

```
from md1 import add,circle
add(10,20)
circle(30)
```

```
30
2826.0
```

Packages

Create Folder “AnhHuy”

msg1.py

```
def msg1():
    print "Day la msg1"
```

msg2.py

```
def msg2():
    print "Day la msg2"
```

msg3.py

```
def msg3():
    print "Day la msg3"
```

__init__.py

```
from AnhHuy.msg1 import msg1
from AnhHuy.msg2 import msg2
from AnhHuy.msg3 import msg3
```

Numpy Arrays

Install library `pip install numpy`

Getting started

```
import numpy as np

# Create 2 new lists height and weight
height = [1.87,  1.87, 1.82, 1.91, 1.90, 1.85]
weight = [81.65, 97.52, 95.25, 92.98, 86.18, 88.45]

# Create 2 numpy arrays from height and weight
np_height = np.array(height)
np_weight = np.array(weight)
print(np_height)
print(np_weight)
```

```
[1.87 1.87 1.82 1.91 1.9  1.85]
[81.65 97.52 95.25 92.98 86.18 88.45]
```

Print out the type of np_height

```
print(type(np_height))
```

```
<class 'numpy.ndarray'>
```

Element-wise calculations

```
bmi = np_weight / np_height ** 2
print(bmi)
```

```
[23.34925219 27.88755755 28.75558507 25.48723993 23.87257618 25.84368152]
```

Subsetting

```
# For a boolean response
bmi > 25
# Print only those observations above 25
bmi[bmi > 25]
```

```
array([27.88755755, 28.75558507, 25.48723993, 25.84368152])
```

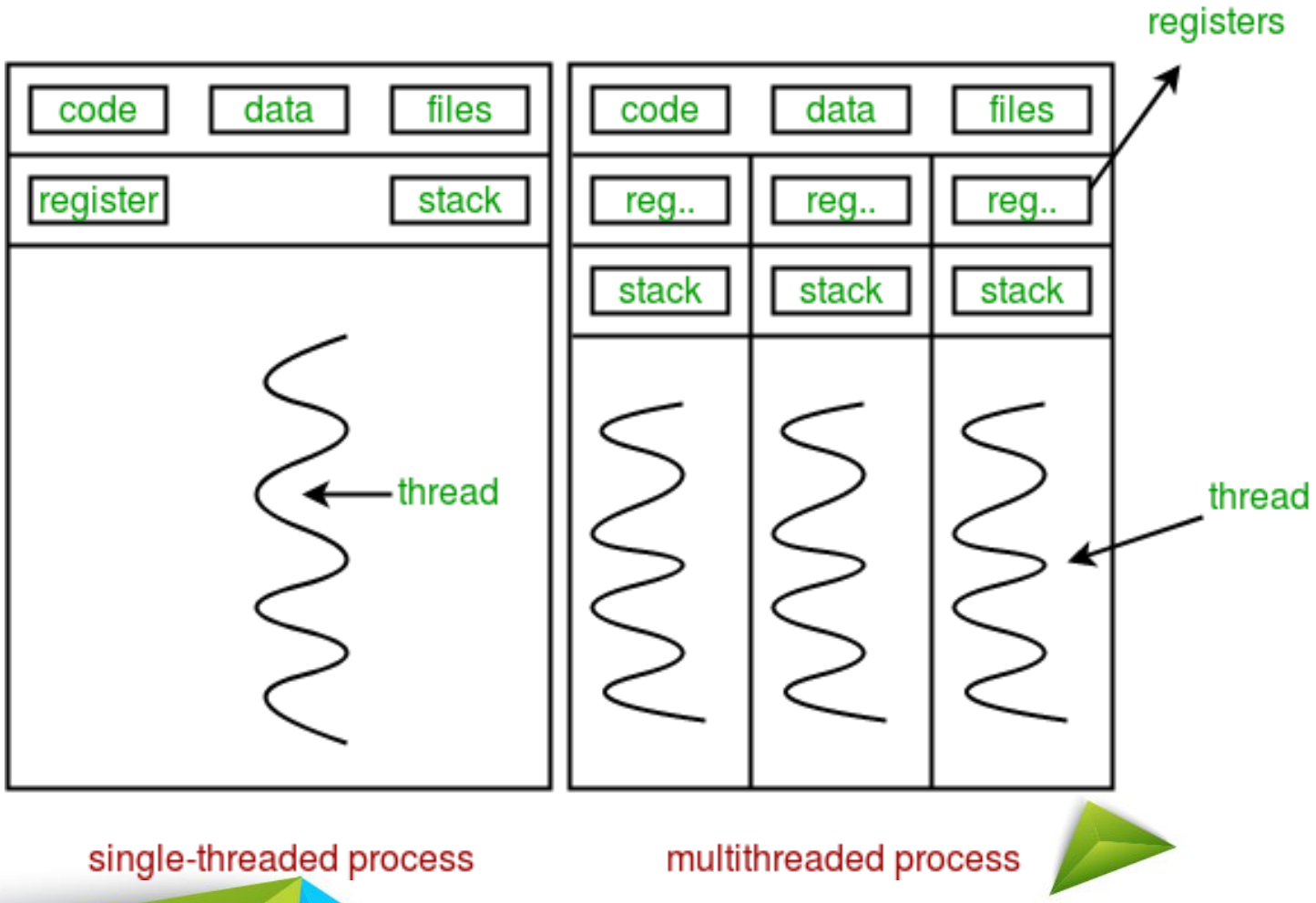
TAPIT.RGB

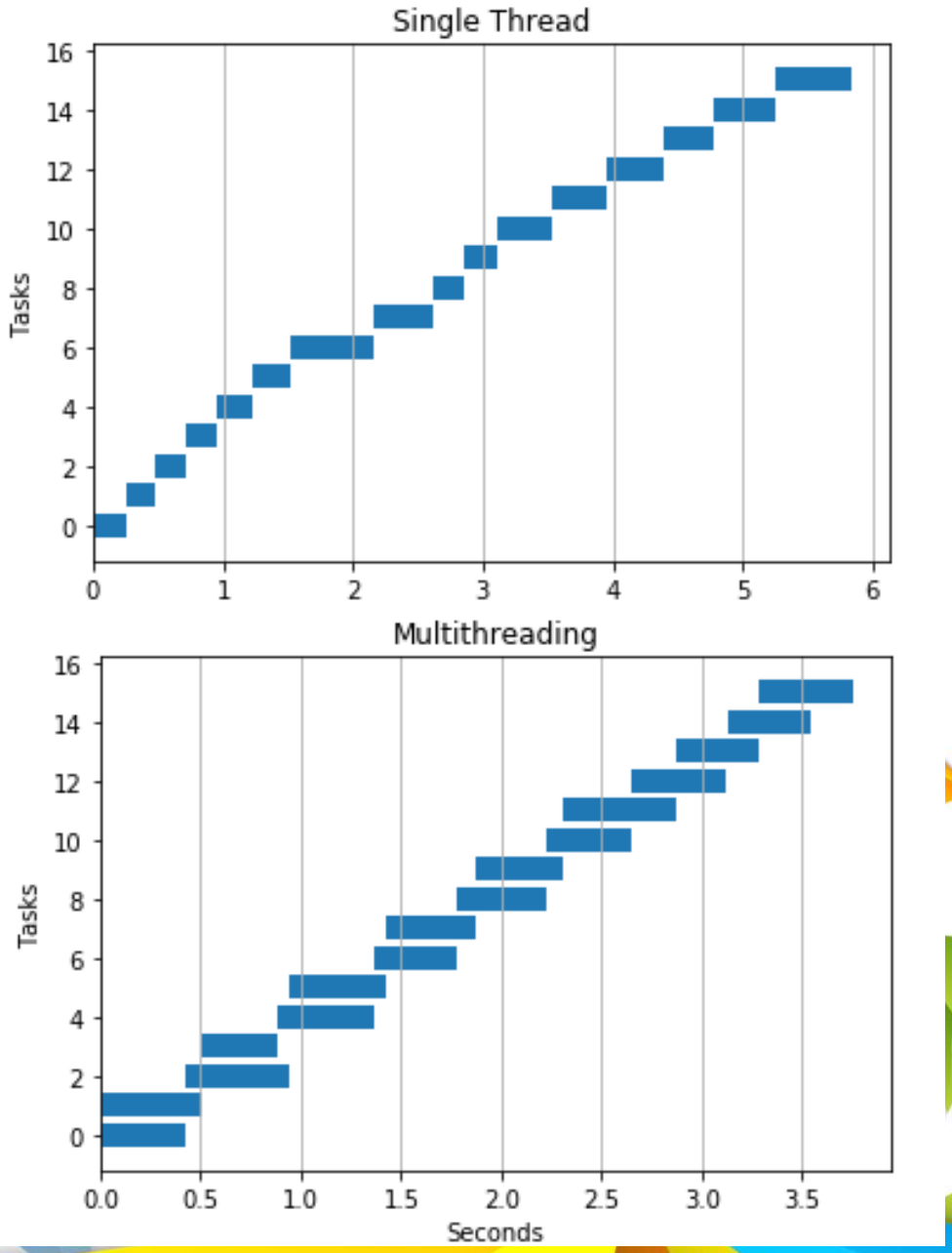
Exercises 1.11

First, convert the list of weights from a **list** to a **Numpy array**. Then, convert all of the **weights** from **kilograms** to **pounds**. Use the scalar conversion of **2.2 lbs per kilogram** to make your conversion. Lastly, **print the resulting array of weights in pounds**.

```
weight_kg = [81.65, 97.52, 95.25, 92.98, 86.18, 88.45]

import numpy as np

# Create a numpy array np_weight_kg from weight_kg


# Create np_weight_lbs from np_weight_kg


# Print out np_weight_lbs
```

Thread


registers
code
data
files
register
stack
thread
code
data
files
reg..
reg..
reg..
stack
stack
stack
thread
single-threaded process
multithreaded process


Single Thread
Tasks
Seconds
Multithreading
Tasks
Seconds

Thread

```python
import threading

def print_cube(num):
    print("Cube: {}".format(num * num * num))

def print_square(num):
    print("Square: {}".format(num * num))

if __name__ == '__main__':
    # creating thread
    t1 = threading.Thread(target=print_cube, args=(10,))
    t2 = threading.Thread(target=print_square, args=(10,))
    # starting thread 1
    t1.start()
    # starting thread 2
    t2.start()

    # wait until thread 1 is completely executed
    t1.join()
    # wait until thread 2 is completely executed
    t2.join()

    print("Done")
```

```
Cube: 1000
Square: 100
Done
```