Python CET Course Day 1

By Seow Khee Wei / Republic Polytechnic

About This Workshop

- Learn about Python 3, a very versatile and useful language
- Discuss its advantages and disadvantages (also what to look out for)
- Improve your problem solving skills:
  How to automate the most boring and repetitive stuff using Python
- The tools and useful modules you can use to build your applications

Prereqs and Preparations

Before you attend this workshop, please make sure:

- Your laptop works
- You have installed the latest version of Python 3
- You have installed a suitable editor:
  We are using Wing IDE Community Edition in this course
- Usage of Chrome web browser, Microsoft Excel or equivalent to read xlsx files, Acrobat PDF reader.

Programme Day One

Morning

- Install Python and using Wing 101 IDE
- Data Types
- If-else
- For loops
- Functions

Afternoon

- Try/except
- String functions, formatting
- Find files by name, by extension, by size, by content and calculate the total size
- Graphical User Interface

Programme Day Two

Morning

- Read and writing files
- Copying, moving and deleting files and folders
- Working with Excel
- Processing CSV files
- Generating PDF

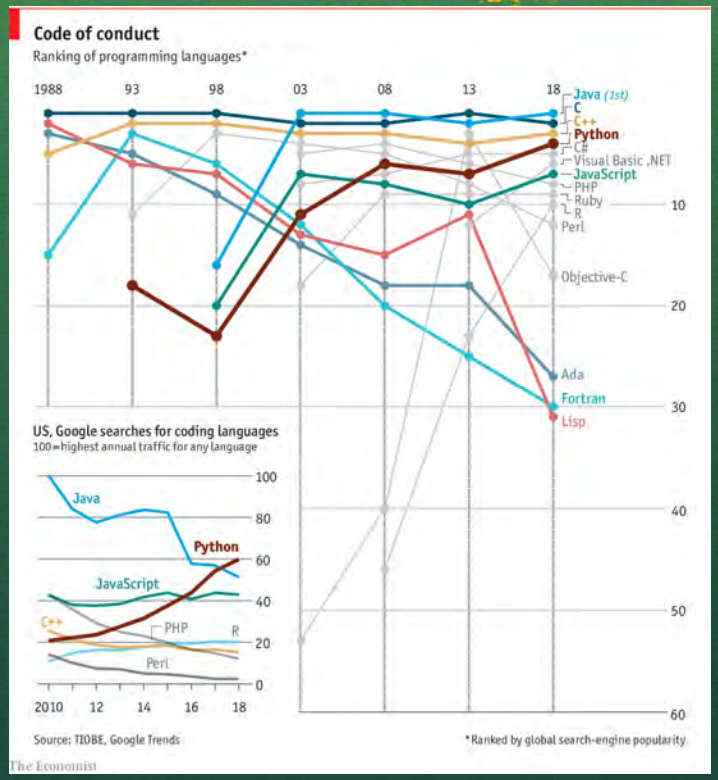
Afternoon

- Image processing
- Data Visualization - Charting
- Connecting to the Web
- Sending emails
- Telegram bot

Introduction to Python

What is Python?

- Interpreted
- Interactive
- Functional
- Object-oriented
- Programming language,
  not just a scripting language



Introduction to Python

- Allows modular programming
- Great emphasis on readability:
  - Codes are forced to be indented
- Easy to embed in and extend with other languages
- Easy to learn for beginners
- Completely FREE!
- Copyrighted but use is not restricted

Introduction to Python

Interfaces to:

- COM, DCOM, ODBC
- Most databases (MySQL, MSQL, Sybase, Oracle,…)
- Java (Jpython)
- Many GUI / GFX libraries
  - Platform-independent: Tk, wxWindows, GTK, Pyglet
  - Platform-specific: MFC, MacOS, X11

Introduction to Python

Who uses Python?

Web Development
- Google (in search spiders)
- Yahoo (in maps application)

Games
- Civilization 4 (game logic & AI)
- Battlefield 2 (score keeping and team balancing)

Graphics
- Industrial Light & Magic (rendering)
- Blender 3D (extension language)

Financial
- ABN AMRO Bank (communicate trade information between systems)

Science
- National Weather Center, US (make maps, create forecasts, etc.)
- NASA (Integrated Planning System)

Education
- University of California, Irvine
- University of New South Wales (Australia)
- Republic Polytechnic, Singapore
- National University of Singapore (NUS)
- Singapore University of Technology and Design (SUTD)
- Singapore Management University (SMU)

http://wiki.python.org/moin/OrganizationsUsingPython

Introduction to Python

What is Python used for?

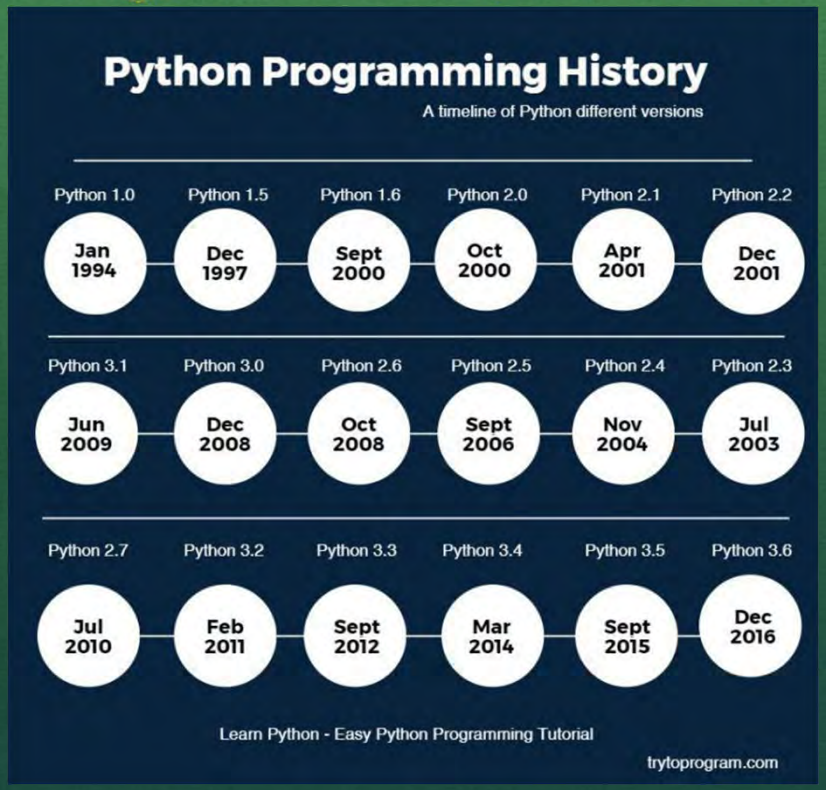
- Web development (Django, Flask, Google App Engine)
- File storage (Dropbox)
- Embedded in software packages (Maya, Blender)
- Scripting to perform simple (but mundane & repetitive) tasks

Introduction to Python

Why the name, Python?

- Originally not a snake, but from the British comedy
  "**Monty Python's Flying Circus**". The snake logo came later.
- Invented in 1990 by Guido Van Rossum
- First public release was in 1991

Introduction to Python



Python has two versions currently: **2.7.15 and 3.7.2**

https://www.python.org/downloads/

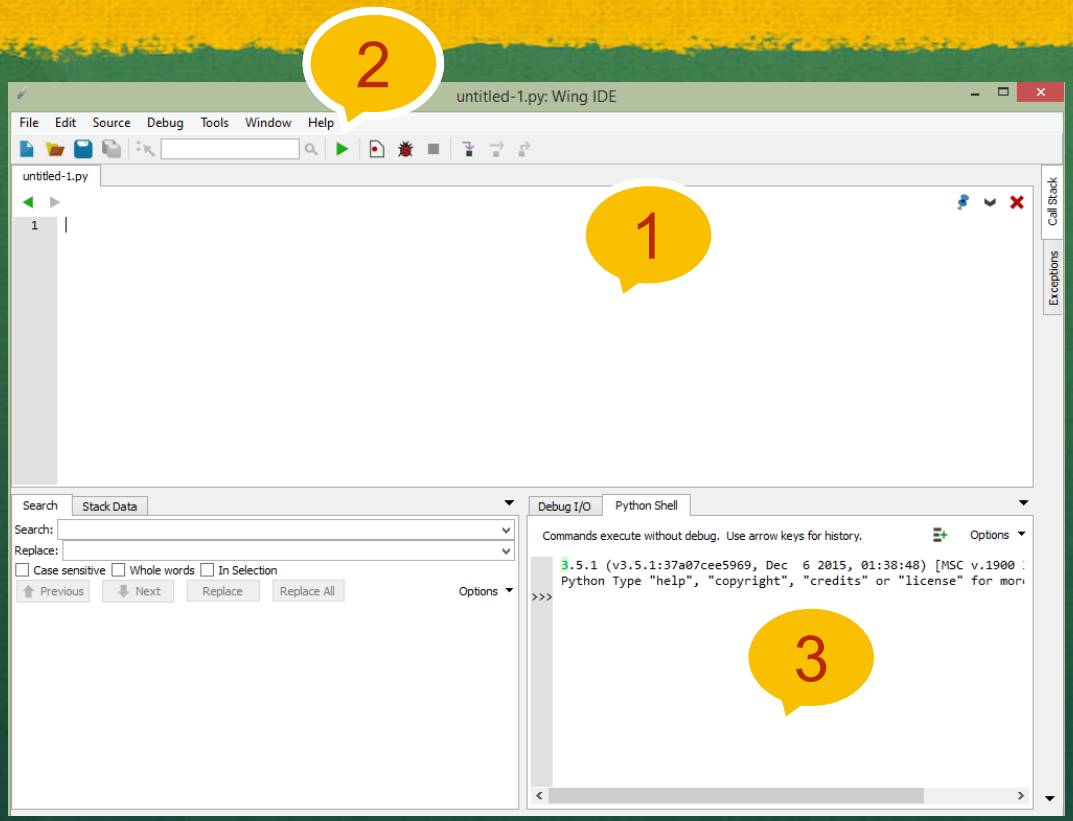
Where is Python on the Map

Compiled			**Interpreted**
Explicitly compiled to machine code	Explicitly compiled to byte code	Implicitly compiled to byte code	Purely interpreted
C, C++, Fortran	Java, C#	**Python**	Shell, Perl

Python 2 vs. Python 3

- **Different syntax**: e.g. print statement, division
  - Python 2
    - ✓ print “Hello World!”
    - ✓ x = 5 / 2 # x’s value will be 2
  - Python 3
    - ✓ print(“Hello World!”) # brackets are compulsory now
    - ✓ x = 5 / 2 # x’s value will be 2.5
- **Which to learn?**
  - Many major frameworks and third-party modules have already migrated or are in the process of moving to Python 3
  - Python 2’s EOL is in 2020, no Python 2.8
  - **The obvious pick: Python 3**

Run Wing101 IDE



1. Editor
2. Run button
3. Output window / Console

Using the Console

- Also known as the interpreter
- See the output straightaway
- Usually used to test very small chunks of code
- Type code after >>>
- Let’s try!

Using the Console

Python Console

```
/usr/local/bin/python3.6 "/Applications/PyCharm CE.app/Contents/helpers/pydev/pydevconsole.py" 58143 58144

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/michelleteo/PycharmProjects/PythonCourse'])

Python 3.6.4 (v3.6.4:d48ecebad5, Dec 18 2017, 21:07:28)
>>> print("Hello World!")
Hello World!
>>> i = 5
>>> j = 10
>>> print(i * j)
50
>>> print(i, j)
5 10
>>> question = "What is your name?"
>>> name = input(question)
What is your name?
>?
```

Start here!

Data Types

We shall focus on these basic data types in our workshop:

Numbers

int	for whole numbers
float	for numbers with decimal point, e.g. 5.2, 2.0

Text

str	for a sequence of characters

Containers

list	a sequence of objects, use an index to access each object

Data Types – Numbers

```
>>> i=3.0
>>> type(i)
<class 'float'>
>>>
>>> i=3
>>> type(i)
<class 'int'>
>>>
>>> print(i/2)
1.5
>>> print(i//2)
1
>>> |
```

- Python is a ‘friendly’ language
- The data type of a variable will change automatically depending on the values assigned to it.
- `i` is a variable
- In the 1st case, a float number is assigned to a variable `i`, so its data type is float.
- In the 2nd case, if an integer value is assigned to the same variable, its data type will change to integer.

Data Types – Strings

```
>>> s = "hello"
>>> t = "world"
>>> print(s+" "+t)
hello world
>>> s = "123"
>>> print(s+4)
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
>>> print(s*4)
123123123123
>>> |
```

Notes:

- For string assignment, you can use “hello” or ‘hello’.
- But not “hello’

- Strings contain characters
- Strings can be added together with the + operator
- Strings can contain number characters, but they are not numbers (int or float)
- You cannot add a number to a string
- The * operator will replicate the string with an integer value

Data Types – Lists

```
>>> l = [1,2,3]
>>> print(l)
[1, 2, 3]
>>> print(l[0])
1
>>> print(l[2])
3
>>> print(l[3])
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
IndexError: list index out of range
>>> |
```

- Lists can contain anything
- Items in a list must be accessed by an index
- First index position starts from 0
- Python doesn't like it if you ask for something that is not in the list
- Try using a negative index, e.g. -1: What happens?

Data Types – Lists

```
>>> l = [1,2,3,4,5,6]
>>> print(len(l))
6
>>> print(l[2:5])
[3, 4, 5]
>>> print(l[:3])
[1, 2, 3]
>>> print(l[3:])
[4, 5, 6]
>>> print(l[-1])
6
>>>
```

- len gives you the length or size of list
- Get a range of items using : colon (This is called slicing)

  [:3] first 3 items

  [3:] last 3 items
- Negative index gives you items from the back

  [-x] x^{th} last item

Range

```
>>> print(list(range(10)))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
>>> print(list(range(1,10)))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
>>> print(list(range(1,10,2)))
[1, 3, 5, 7, 9]
>>>
>>> print(list(range(10,1,-1)))
[10, 9, 8, 7, 6, 5, 4, 3, 2]
>>>
```

Note: if s is negative, then step down by its absolute value

Three versions:

- `range(y)`
  starts at 0
  ends before y
  step up by 1
- `range(x, y)`
  starts at x
  ends before y
  step up by 1
- `range(x, y, s)`
  starts at x
  ends before y
  step up by s

Data Types – Conversion

```
>>> si="5"
>>> sf="5.5"
>>> print(int(si), float(sf))
5 5.5
>>> i=5
>>> print("int to str:"+str(i))
int to str:5
>>> 
```

- You can convert anything into its string version:
  str(…)
- A string containing all digit characters a decimal point can be converted into a number type:
  int(…) or float(...)

Print Formatted Numbers

```
>>> import math
>>> print("Pi is " + str(math.pi))
Pi is 3.141592653589793
>>> print("Pi is approx %.2f"%(math.pi))
Pi is approx 3.14
>>> print("Pos or Neg: %+d %+d"%(-5,3))
Pos or Neg: -5 +3
>>>
```

```
print("Art: %5d,  Price per Unit: %8.2f" % (453, 59.058))
```

output

String Modulo Operator

```
Art:   453,  Price per Unit:    59.06
```

Formatting numbers

%d int
%f float
%s string

Special formatting

%.2f

- float
- two digits behind the point

%+d (or f)

- force print the sign

The time library

One of the functions in the time library is strftime, a flexible function to display the time based on certain format:

```
import time

print(time.strftime("Today is %d-%m-%Y %H:%M",time.localtime()))
```

https://docs.python.org/3/library/time.html?highlight=strftime#time.strftime

```
>>>
Today is 04-06-2017 17:30
>>>
```

The datetime library

```python
from datetime import datetime, timedelta
d1 = datetime(1991, 4, 30)
print(d1)
# -> 1991-04-30 00:00:00

d2 = d1 + timedelta(10)
print(d2)
# -> 1991-05-10 00:00:00

print(d2 - d1)
# -> 10 days, 0:00:00

d3 = d1 - timedelta(100)
print(d3)
# -> 1991-01-20 00:00:00

d4 = d1 - 2 * timedelta(50)
print(d4)

#adding seconds
d1 = datetime(1991, 4, 30)
print(d1)
# -> 1991-04-30 00:00:00

d2 = d1 + timedelta(10,100)
print(d2)
# -> 1991-05-10 00:01:40

print(d2 - d1)
# -> 10 days, 0:01:40

#get current Date
d1 = datetime.now()
print(d1)
# -> 2019-03-07 12:51:51.196327
print(d1.day, d1.hour, d1.second)
# -> 2019-03-07 12:51:51.196327
```

The datetime module allows manipulating dates and times in both simple and complex ways. date and time arithmetic is supported

Basic Arithmetic

Operator Name	Code	Example When x = 2 and y = 1
Plus	x + y	x + y will give 3
Minus	x – y	x – y will give 1
Divide	x / y	x / y will give 2.0
Multiply	x * y	x * y will give 2 You must use * instead of x for multiplication.
x to the power of y	x ** y	x ** y means 2 to power of 1 and will give 2
Modulus	x % y	x % y will give 0 0 is the remainder from 2 divides by 1

Exercise – Homework Calculator

- Mick took 3.5 hours to finish his homework. Alice took 2.5 hours to finish her homework. Write a program to calculate the total amount of time in <u>seconds</u> that they took to finish their homework

Exercise – Time Conversion

- Write a program (in 1 script file) to convert 1000 seconds to minutes and seconds.

```
Debug I/O (stdin, stdout, stderr) appears below

Minutes: 16
Remaining Seconds: 40
Time in mins and secs: 16min and 40sec
```

Getting User Input

- You can use **input()** function to ask for user input.
- The value entered by the user is stored into **a variable** as a **string**.
- If the value is to be used as a number, you can use the **int()** or **float()** function to **convert** the value to the appropriate number data type.

```
>>> word = input("Enter a word: ")
    Enter a word: hello
>>> print(word)
    hello
>>> type(word)
    <class 'str'>
>>>
```

```
>>> num = input("Enter a Whole number : ")
    Enter a Whole number : 8
>>> print(num)
    8
>>> type(num)
    <class 'str'>
>>> num = int(num)
>>> print(num)
    8
>>> type(num)
    <class 'int'>
>>> |
```

Exercise – Temperature Calculator

The normal human body temperature is 36.9 Degree Celsius. Write a program to ask the user for name and temperature and print a message on the screen that indicate the temperature difference from the normal body temperature.

```
Enter patient's name:-John
Enter patient's temperature:-37.5
John's temperature is 0.6 degree celsius from 36.9 degree celsius.
```

If-Else Statement

```
1  correct_password = "secret"
2  password = input("Enter password:-")
3
4  if password == correct_password:
5      print("You have entered correct password")
6  else:
7      print("You have entered wrong password")
```

```
Enter password:- secret
You have entered correct password
```

All lines, except line 5 are executed as **if password == correct_password** returns **True**.

```
Enter password:- letsguess
You have entered wrong password
```

All lines, except lines 6-7 are executed as **if password == correct_password** returns **False**.

Conditions in Decision Making

The condition(s) in a test can be expressed through the use of the following comparison operators.

Expression	What it does
a == b	Evaluates to True when a is equal to b
a != b	Evaluates to True when a is not equal to b
a < b	Evaluates to True when a is lesser than b
a > b	Evaluates to True when a is bigger than b
a <= b	Evaluates to True when a is lesser than or equal to b
a >= b	Evaluates to True when a is greater than or equal to b

if…elif…else

1. Type out the code below:

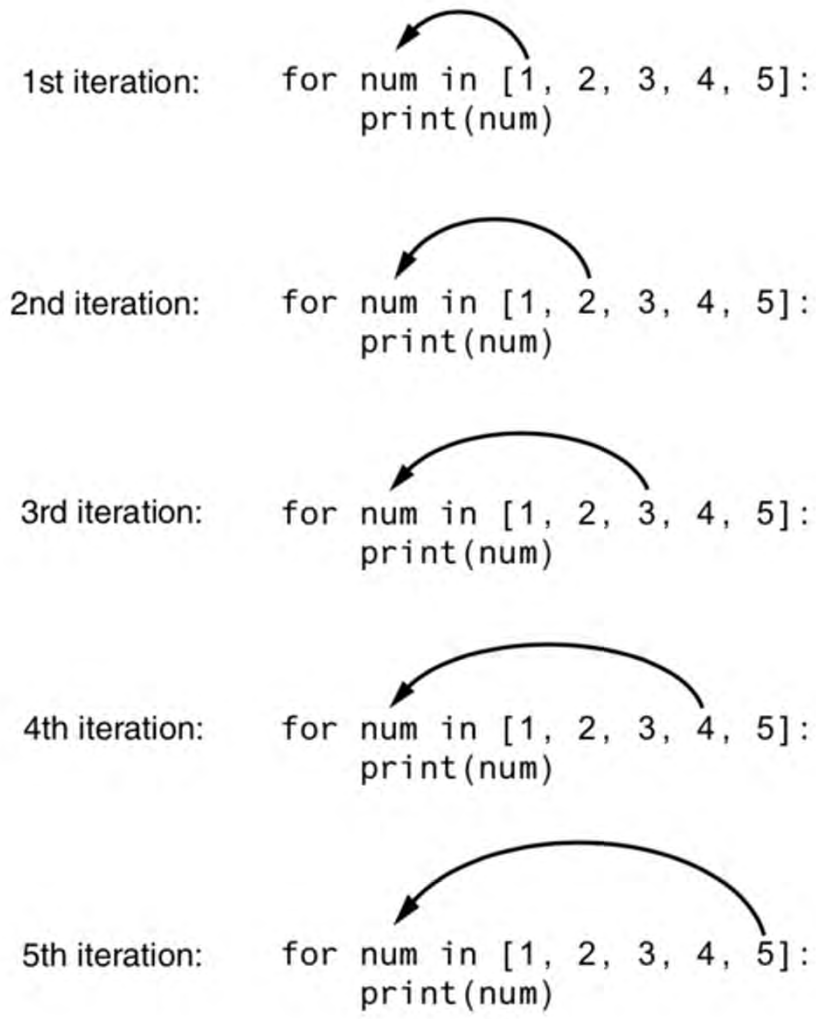
```
number1 = input("Enter first number:- ")
number2 = input("Enter second number:- ")
number1 = float(number1)
number2 = float(number2)

if number1 < number2:
    print("First number is smaller than the second number.")
elif number1 > number2:
    print("First number is greater than the second number.")
else:
    print("Two numbers are equal.")
```

Exercise - BMI Calculator

Develop a BMI Calculator to calculate the BMI of a patient given the weight and height.

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height (m)} \times \text{Height (m)}}$$

Category	Underweight	Ideal	Overweight	Obese
$\text{BMI} = \frac{weight(kg)}{height(m)^2}$	< 18	≥ 18, but < 25	≥ 25, but < 30	≥ 30



For Loops

- For loops often go hand-in-hand with lists
- Every object in the list will be processed by what is inside the for loop
- What is the data type of `i`?

Image source : Starting out with Python, 4th Ed, Tony Gaddis

For Loops

```
>>> numbers = range(10)
>>> for i in numbers:
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>> |
```

Notice how each call of print at each loop will print at a different line.

How do we print numbers 0 to 9 all on the same line (0123456789)?

For Loops

```
>>> s = "freedom"
>>> for c in s:
...     print(c,end=" ")
...
f r e e d o m
>>> |
```

- A string is a sequence, like a list
- The for loop works similarly with strings

For Loops

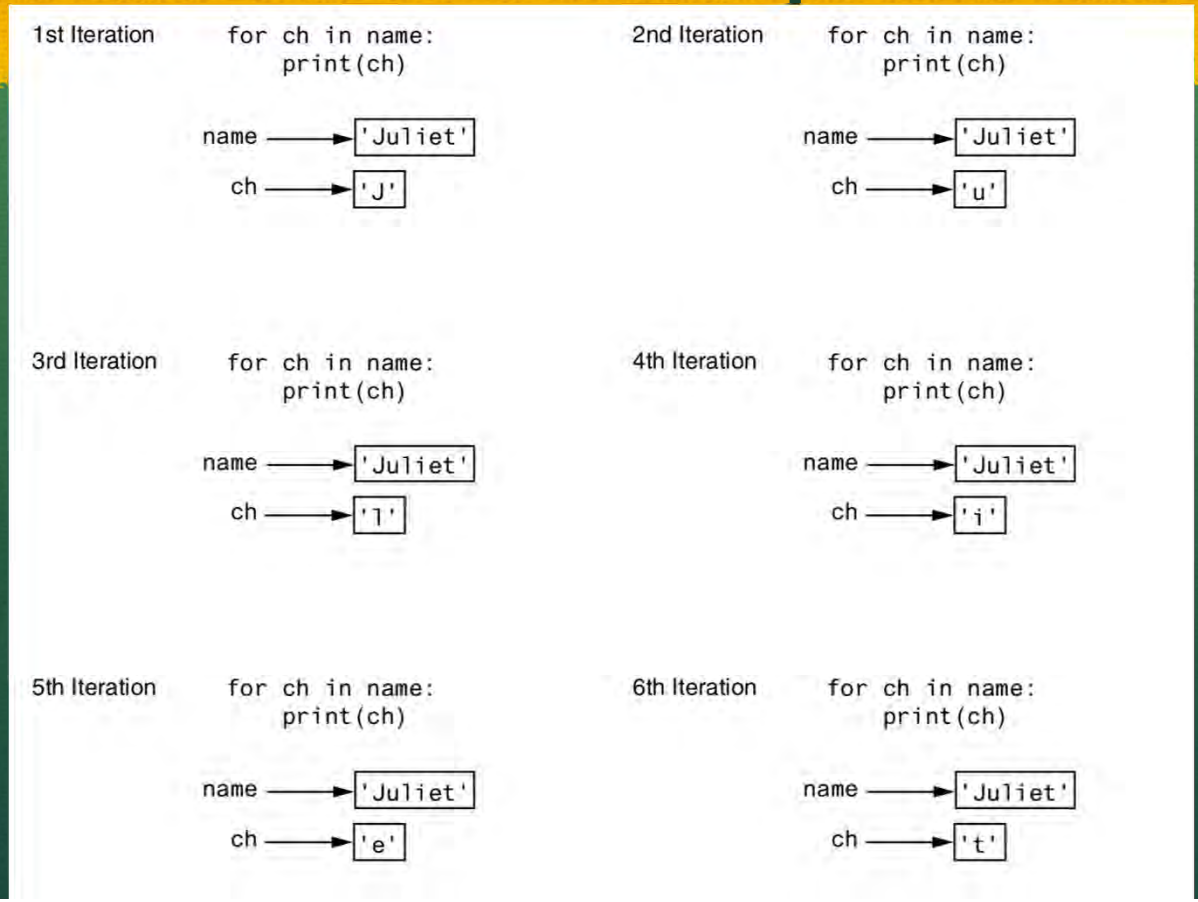


Image source : Starting out with Python, 4th Ed, Tony Gaddis

For Loops

```
>>> s = "freedom"
>>> print(s[:4])
free
>>> print(s[-3:])
dom
>>> |
```

Slicing works for any sequence, so it works for strings too.

[:4] gets from the start till the fourth character

[-3:] gets the last third till the last character.

Data Types – Dictionary

```
{'year': '1995', 'type_of_public_transport': 'MRT', 'average_ridership': '740000'}
{'year': '1995', 'type_of_public_transport': 'LRT', 'average_ridership': '0'}
{'year': '1995', 'type_of_public_transport': 'Bus', 'average_ridership': '3009000'}
{'year': '1995', 'type_of_public_transport': 'Taxi', 'average_ridership': '0'}
{'year': '1996', 'type_of_public_transport': 'MRT', 'average_ridership': '850000'}
{'year': '1996', 'type_of_public_transport': 'LRT', 'average_ridership': '0'}
```

- A dictionary stores multiple key-value pairs
- E.g. In the first row of output, the dictionary contains 3 key-value pairs (which are the keys?)
- Every key is unique; no duplicate key within a dictionary
- A dictionary uses a set of curly brackets to store its key-value pairs {…}
  => Contrast with a list that uses square brackets to store its objects [...]
- To access a value in the dictionary, we use the key as an index

Data Types – Dictionary

```
>>> scores = {'Mary': 90, 'Ben': 67, 'Jenny': 21}
>>> for s in scores:
...     print(s)
...
Mary
Ben
Jenny
```

- How does a `for` loop work on dictionaries?
- Doing '`for s in scores`' in the above code will assign the value of each key to `s`
- Change '`print(s)`' to '`print(s, scores[s])`', what do you get?

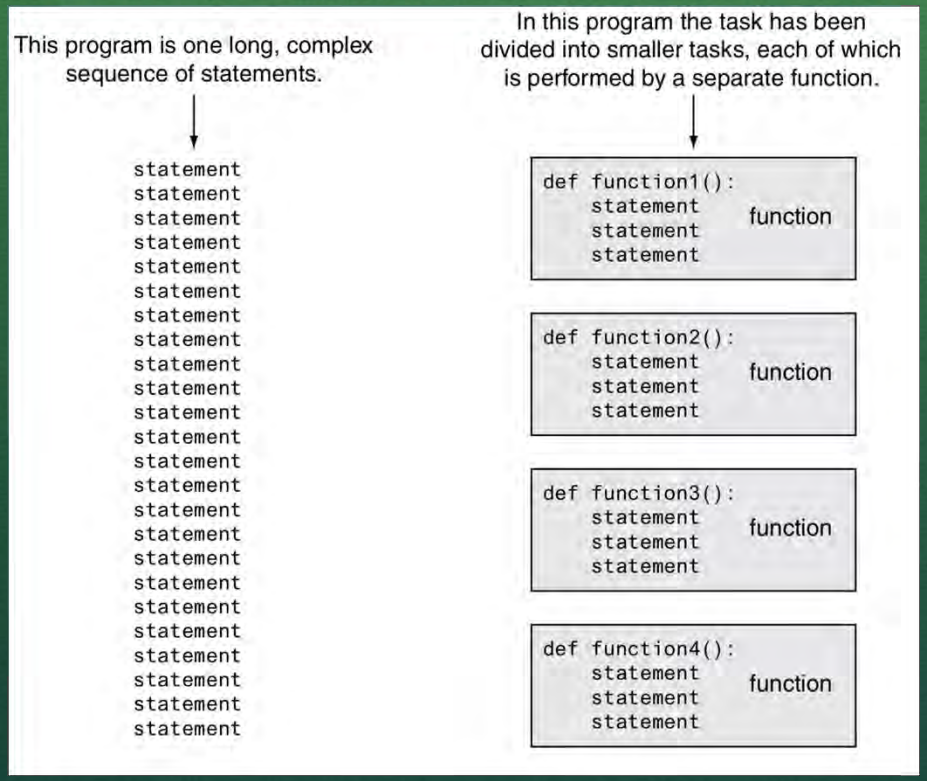
Exercise – Even Odd Counter

Write and test a program that will read 10 positive integer numbers, determine if it is even or odd, keep count of the number of even and odd numbers and display the final outcome as follows:

```
Enter number 1: 12
Enter number 2: 7
. . .
Enter number 10 : 67
Even #: 4
Odd #: 6
```

- Q: What if a user does not enter a positive integer?

Functions



A function is a group of statements that exist within a program for the purpose of performing a specific task.

Image source : Starting out with Python, 4th Ed, Tony Gaddis

Functions

```
>>> def myFunction():
...     print("hello")
...
>>> type(myFunction)
<class 'function'>
>>> myFunction
<function myFunction at 0x03C74738>
>>> myFunction()
hello
>>>
```

```
>>> import math
>>> def calcPHI():
...     return (math.sqrt(5)-1)/2
...
>>> calcPHI()
0.6180339887498949
```

Functions

identCar

```
1  def multiply(a,b):
2    return a*b
3
4  print(multiply(2,3))
5
6  def identCar(car,colour):
7    return "Car %s has color %s" % (car, colour)
8
9  print(identCar(colour="red", car="honda"))
10
```

Be sure to have indentation

Search | Stack Data | Debug I/O | Python Shell

Search:

Replace:

Case s Whole In Sele

Commands execute without debug. Use arrow keys for hist

Options

```
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Python Type "help", "copyright", "credits" or "license" fo
>>> [evaluate function_01.py]
6
Car honda has color red
>>>
```

Functions

Passing parameters

- Python uses neither “pass-by-reference” nor “pass-by-value”
- It uses “pass-by-object”
- Arguments must be passed in order

```
>>> def multiply(a,b):
        return a*b

>>> multiply(2,3)
6
>>>
```

Alternatively, use parameter names to identify the arguments

```
>>> def identCar(car,colour):
...     print("Car %s has colour %s"%(car,colour))
...
>>> identCar(colour='red', car='honda')
Car honda has colour red
>>>
```

Default parameters

Default parameters values and checking if parameter has been passed

identCar

```
1  def identCar(car=None, colour="red"):
2    if car == None:
3      print("You have to give me a car name")
4      return
5    print("Car %s has colour %s" % (car, colour))
6
7  identCar(colour="blue")
8  # You have to give me a car name
9
10 identCar(car="toyota")
11 # Car toyota has colour red
```

Search | Debug I/O | Python Shell

Search:

Replace:

Commands execute without debug. Use Options

```
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELE
Python Type "help", "copyright", "credits" o
>>> [evaluate function_02.py]
You have to give me a car name
Car toyota has colour red
```

Arbitrary argument list

If you don't know how many parameters the function will receive, you can use *args, which will be a list.

```
>>> def addAll(*args):
...     sum=0
...     for num in args:
...         sum+=num
...     return sum
...
>>> addAll(1,2)
3
>>> addAll(1,2,3,4,5,6,7,8,9)
45
>>>
```

Create a function that takes in an unknown amount of parameters and returns the sum.

try .. except

Error handling is done through the use of exceptions that are caught in try blocks and handled in except blocks

```
>>> try:
...     5/0
... except:
...     print("error")
...
error
>>>
```

```
>>> try:
...     5/0
... except Exception as e:
...     print("Exception ",type(e),": ",e.args)
...
Exception  <class 'ZeroDivisionError'> :  ('division by zero',)
>>>
```

try .. except

You can also use the finally block. The code in the finally block will be executed regardless of whether an exception occurs.

```
>>> try:
...     5/0
... finally:
...     print("oeps, just before we run into an exception.")
...
oeps, just before we run into an exception.
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 2, in <module>
ZeroDivisionError: division by zero
>>>
```

try .. except

A good use for try expect is to check if the user has the specific library installed and if now, explains to the user what to do:

```
>>> try:
...     import special_module
... except ImportError:
...     print("Sorry, you don't have the special_module module installed,")
...     print("and this program relies on it.")
...     print("Please install or reconfigure special_module and try again.")
...
Sorry, you don't have the special_module module installed,
and this program relies on it.
Please install or reconfigure special_module and try again.
>>>
```

try .. except

Another example is to check if a website is available:

```
1 from urllib.request import urlopen
2 def isOnline(reliableserver='http://www.google.com'):
3     try:
4         urlopen(reliableserver)
5         return True
6     except IOError:
7         return False
```

```
>>> isOnline()
True
>>>
```

String functions

Split

```
>>> a='python or java'
>>> b=a.split(' ')
>>> type(b)
<type 'list'>
>>> b
['python', 'or', 'java']
>>>
```

```
>>> a='python or java'
>>> b=a.split('on')
>>> b
['pyth', ' or java']
>>>
```

Join

```
>>> a=['python','and','java']
>>> b=' '.join(a)
>>> b
'python and java'
>>> c=','.join(a)
>>> c
'python,and,java'
>>>
```

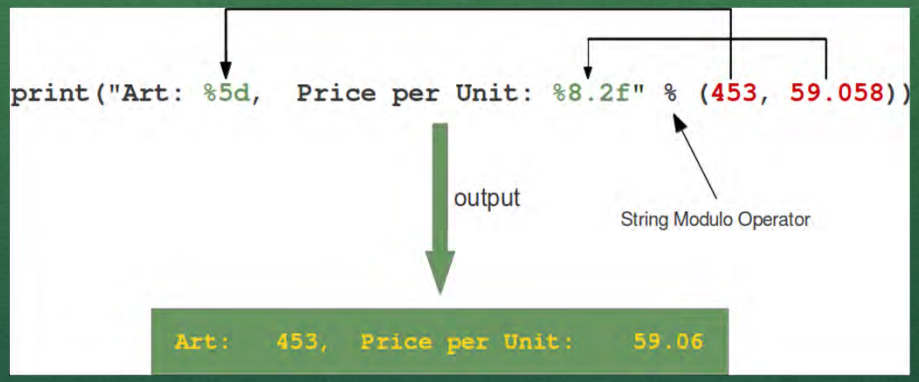
Exercise – Find Longest Word

Create the function findLongestWord that takes in a sentence and returns the longest word. Hint: Use split()

String formatting

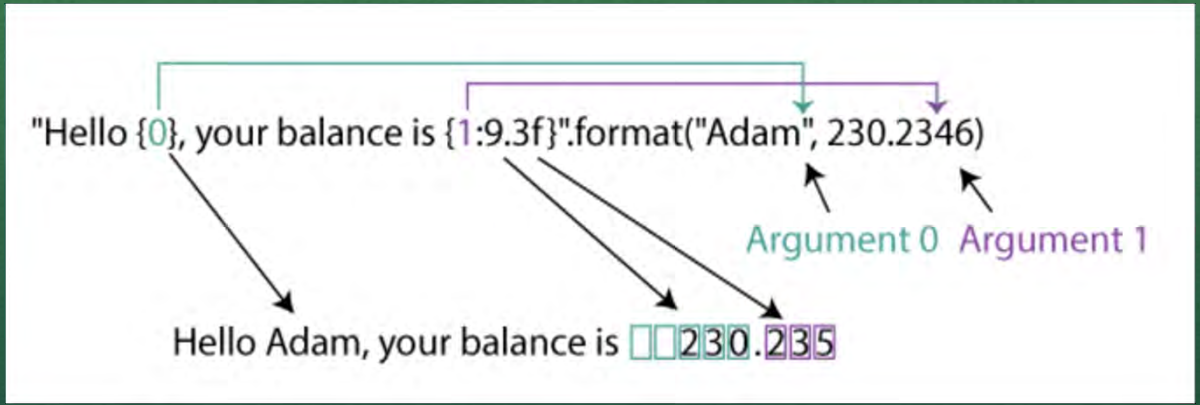
Try this out yourself !

```
>>> import math
>>> a = math.pi
>>> a
3.141592653589793
>>> b=5
>>> c="python"
>>> line="%s %f %d"%(c,a,b)
>>> line
'python 3.141593 5'
>>>
```



```
>>> line="%03d"%(b)
>>> line
'005'
```

String formatting



More string formatting

With c="python", a=3 and b=5

```
>>> line="%15s %.0f %d"%(c,a,b)
>>> line
'         python 3 5'
>>>
```

```
>>> "%-15s"%(c)
'python         '
>>>
```

```
>>> "%(language)s has %(#)03d quote types"%{'language':'python',"#":2}
'python has 002 quote types'
>>>
```

More about this string formatting technique can be found here:
http://docs.python.org/library/stdtypes.html#string-formatting-operations
format( )

Exercise – Xmas Tree

Question: Using string formatting and a loop, try to print the following xmas tree:

```
  ##
 ####
######
########
```

The random library

random.randint(a, b)
Return a random integer N such that
a <= N <= b

random.random()
Return the next random floating point number in the range [0.0, 1.0]

Other random functions
random.shuffle(List)
random.choice(List)

More at http://docs.python.org/library/random.html

Exercise - Guessing Game

- Create a random number between 1 and 20 and prompt the user to guess the secret number. He is allowed a maximum of 6 guesses after which the secret number will be displayed and the program exits. For every guess, the program will display a message saying if the number guessed is higher or lower than the secret number. If he guessed the correct number, the program will display the number of tries he had taken and the program exits.

Exercise - Guessing Game

- Sample output

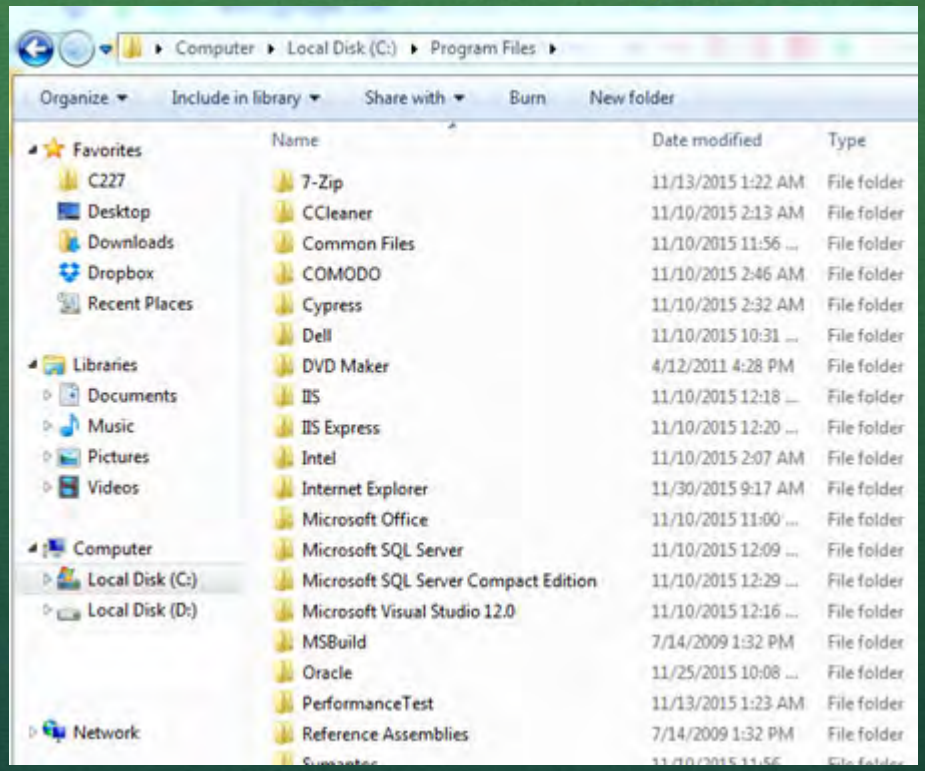
```
What is your name?
John
Well, John, I am thinking of a number between 1 and 20
Take a guess
5
Your guess is too low.
Take a guess
10
Your guess is too low.
Take a guess
15
Your guess is too high.
Take a guess
12
Your guess is too low.
Take a guess
14
Good job, John! You guessed my number in 5 guesses!

Process finished with exit code 0
```

```
What is your name?
John
Well, John, I am thinking of a number between 1 and 20
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
Take a guess
10
Your guess is too high.
nope. The number I was thinking of was 6

Process finished with exit code 0
```

Search for file by filename

Let's say you want to find a file by filename "readme.txt" and it's somewhere in

C:\Program Files\



Search for file by filename

We need to work with the os library, so we import it at the start.

The folder we search in is stored in the variable where.

At line 6 we create a new function called searchByName.
It has one parameter, which is name and contains the name of the file we are looking for.

In this function we walk through all directories and files.
Then for each file we compare if it is the filename we are looking for.

```
1   import os
2
3   where = "C:\\Program Files\\"
4   #where = "C:\\Program Files (x86)\\"
5
6   def searchByName(name):
7       for root, dirs, files in os.walk(where):
8           for file in files:
9               if file == name:
10                  print (os.path.join(root, file))
11
12  searchByName("readme.txt")
13
```

Search for file by filename

Let's say you want to know the combined filesize of all these readme.txt files.

Initialize a new variable totalSize to 0 at the start of our function.

Then for each readme.txt we add the size to this variable with

totalSize += os.path.getsize(os.path.join(root,file))

Then we make this function return this value with return totalSize at the very end of the function with

return totalSize

```
def searchByName(name):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            if file == name:
                print(os.path.join(root,file))
                totalSize += os.path.getsize(os.path.join(root,file))
    return totalSize
```

Make the amendments as indicated above, then run it and observe.

Did it show you the total ?

Search for file by filename

The function returned the value, but we didn't do anything with it.

When we call the function, we can assign the value to a variable.
Then we can print the content.

```
total = searchByName("readme.txt")
print ("Total is : %d" % (total))
print ("All done")
```

```
C:\Program Files\Unity\MonoDevelop\Addins\MonoDevelop.AspNet\Schemas\readme.txt
C:\Program Files\Unity\MonoDevelop\Addins\MonoDevelop.XmlEditor\schemas\readme.txt
Total is 17569
All done.
>>>
```

Exercise – listing files

- Can you update the program to show the individual file size of all the files?

Hint: use os.path.getsize()

```
>>>
  1761 C:\Program Files\7-Zip\readme.txt
    83 C:\Program Files\Unity\Editor\Data\Playba
    62 C:\Program Files\Unity\Editor\Data\Playba
   549 C:\Program Files\Unity\Editor\Data\Playba
   695 C:\Program Files\Unity\Editor\Data\Playba
   126 C:\Program Files\Unity\Editor\Data\Playba
    89 C:\Program Files\Unity\Editor\Data\Playba
    66 C:\Program Files\Unity\Editor\Data\Playba
   250 C:\Program Files\Unity\Editor\Data\Playba
    25 C:\Program Files\Unity\Editor\Data\Playba
 11333 C:\Program Files\Unity\Editor\Data\Playba
   718 C:\Program Files\Unity\Editor\Data\Playba
   906 C:\Program Files\Unity\MonoDevelop\Addins
   906 C:\Program Files\Unity\MonoDevelop\Addins
Total is 17569
All done.
>>>
```

Search for file by filename

Since we use the full filename multiple times, it makes sense to store it in a separate variable.
Same for the filesize.

The %6d makes sure the output looks nice, in columns format

```
def searchByName(name):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            if file == name:
                fullName = os.path.join(root,file)
                fileSize = os.path.getsize(fullName)
                print("%6d %s"%(fileSize, fullName))
                totalSize += fileSize
    return totalSize
```

Find files larger than xxMB

Create a new function called searchBySize, that takes one parameter and only prints those files larger than that parameter.

for example:
searchBySize(50000000)
will only print the full path and filename of those files that exceed 50Mb

You still need the if statement but it's not based on the name. Do you know the fileSize at the point of the if statement ?

```
def searchByName(name):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            if file == name:
                fullName = os.path.join(root,file)
                fileSize = os.path.getsize(fullName)
                print("%6d %s"%(fileSize, fullName))
                totalSize += fileSize
    return totalSize
```

Find files larger than xxMB

Most of the function can be copied, but you need to move the declaration of fullName and fileSize before the if statement.

The if statement then can use the fileSize.

You can still return the totalSize although that was not required.

```
def searchBySize(size):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if fileSize > size:
                print("%6d %s"%(fileSize, fullName))
                totalSize += fileSize
    return totalSize
```

Find files of certain file type

What if you want to find all the files of specific file type ?

Create another function (copy the last) and call it searchByExtension
It has to take one parameter which is the extension to look for.

```
def searchByExtension(ext):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if file.endswith(ext):
                print("%6d %s"%(fileSize, fullName))
                totalSize += fileSize
    return totalSize
```

We can use the buildin function .endswith(“.doc”) in our if statement to compare:
if file.endswith(“.doc”):

You should be able to call this function like:
searchByExtension(“.doc”)

Find files of certain file type

Copy the last function and call it **searchByContent**.

This new function has to take one more parameter (call it keyword) which is the string.
This string has to be present in the file in order to be counted.

You can combine two conditions in an if statement with and :

if …. and …. :
print("okay")

```
def searchByContent(ext, keyword):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if file.endswith(ext) and keyword in open(fullName, encoding="Latin-1").read():
                    print("%6d %s"%(fileSize, fullName))
                    totalSize += fileSize
    return totalSize
```

Find files of certain file type

And this is how it looks like.
This function is slower because it has to go through all the content in the files.

open(fullName).read() literally represents the whole file!
Large files might be a problem because they can not fit in memory at one time.

A better solution would be to load blocks of content, but we save that for another course.

```
def searchByContent(ext, keyword):
    totalSize = 0
    for root, dirs, files in os.walk(where):
        for file in files:
            fullName = os.path.join(root,file)
            fileSize = os.path.getsize(fullName)
            if file.endswith(ext) and keyword in open(fullName, encoding="Latin-1").read():
                    print("%6d %s"%(fileSize, fullName))
                    totalSize += fileSize
    return totalSize
```

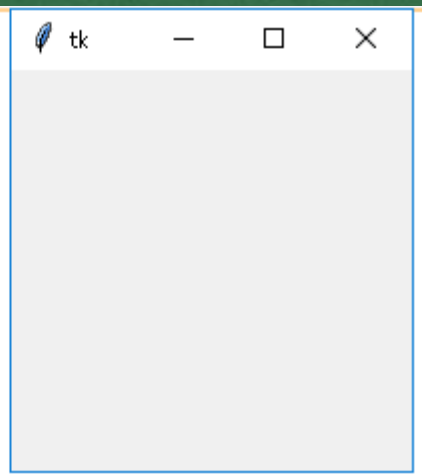
Note:
In your if statement, put the file.endswith(ext) first.
This way it will not execute the second part if the first part is already false and thus save precious time.

Graphical User Interface

https://wiki.python.org/moin/GuiProgramming

Tkinter – Python's standard GUI library
It is a commonly used GuiProgramming toolkit for Python.

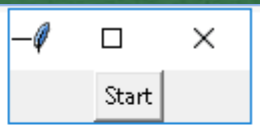
```
import tkinter

window = tkinter.Tk()
window.mainloop()
```



Graphical User Interface

Add a button

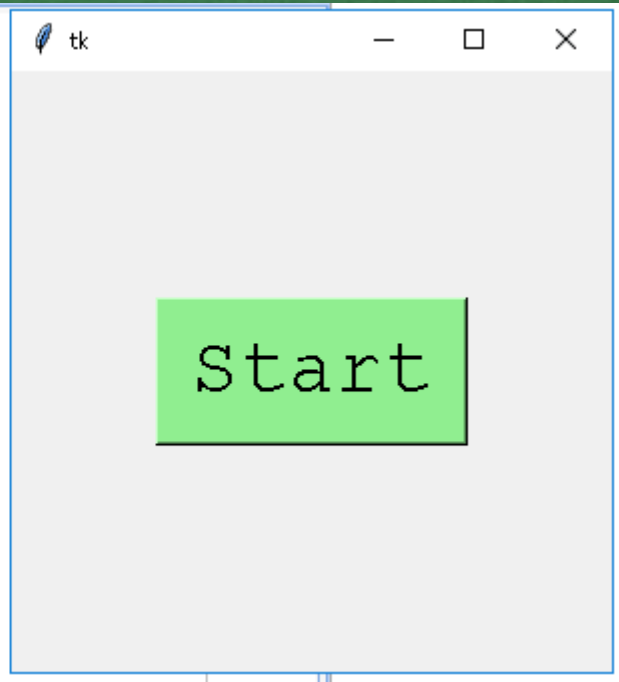
```python
import tkinter

window = tkinter.Tk()

# Add a button
button1 = tkinter.Button(window, text="Start")
button1.pack()

window.mainloop()
```



Graphical User Interface

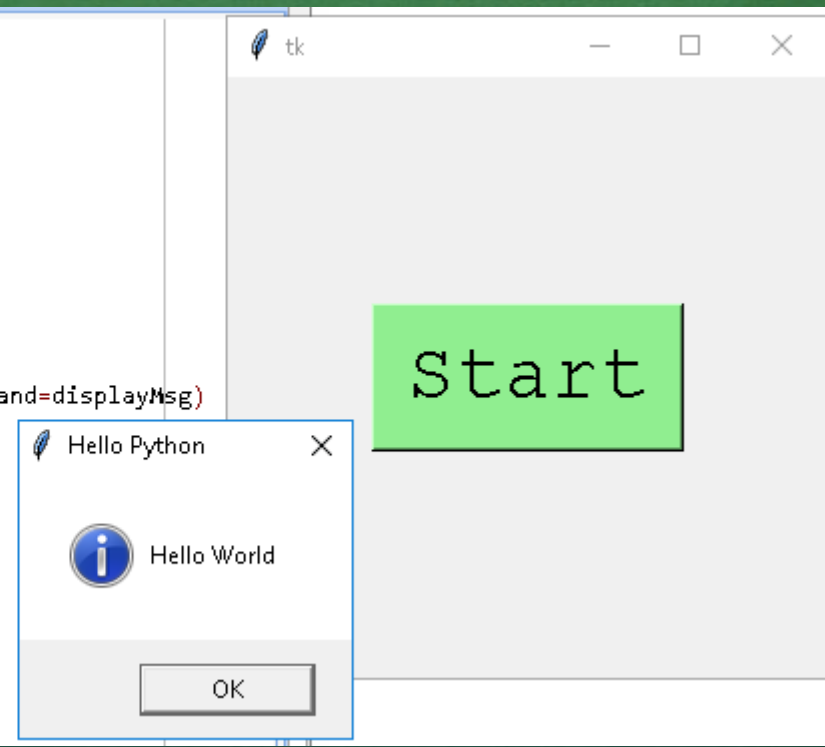
Set the window's size.
Configure the colour and position of the button.

```python
import tkinter
import tkinter.messagebox

window = tkinter.Tk()

# Set the window's size
window.geometry("300x300")

# Add and configure a button
button1 = tkinter.Button(window, text="Start", bg="lightgreen")
button1.config(font=("Courier",30))
button1.pack(side="top", expand=tkinter.YES)

window.mainloop()
```



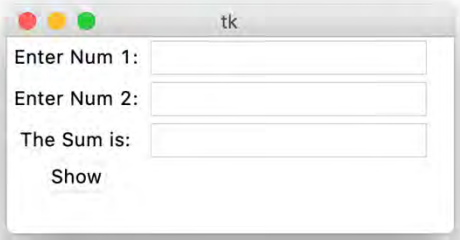
Graphical User Interface

```python
import tkinter
import tkinter.messagebox

window = tkinter.Tk()

# Set the window's size
window.geometry("300x300")

def displayMsg():
    tkinter.messagebox.showinfo("Hello Python", "Hello World")

# Add and configure a button
button1 = tkinter.Button(window, text="Start", bg="lightgreen", command=displayMsg)
button1.config(font=("Courier",30))
button1.pack(side="top", expand=tkinter.YES)

window.mainloop()
```

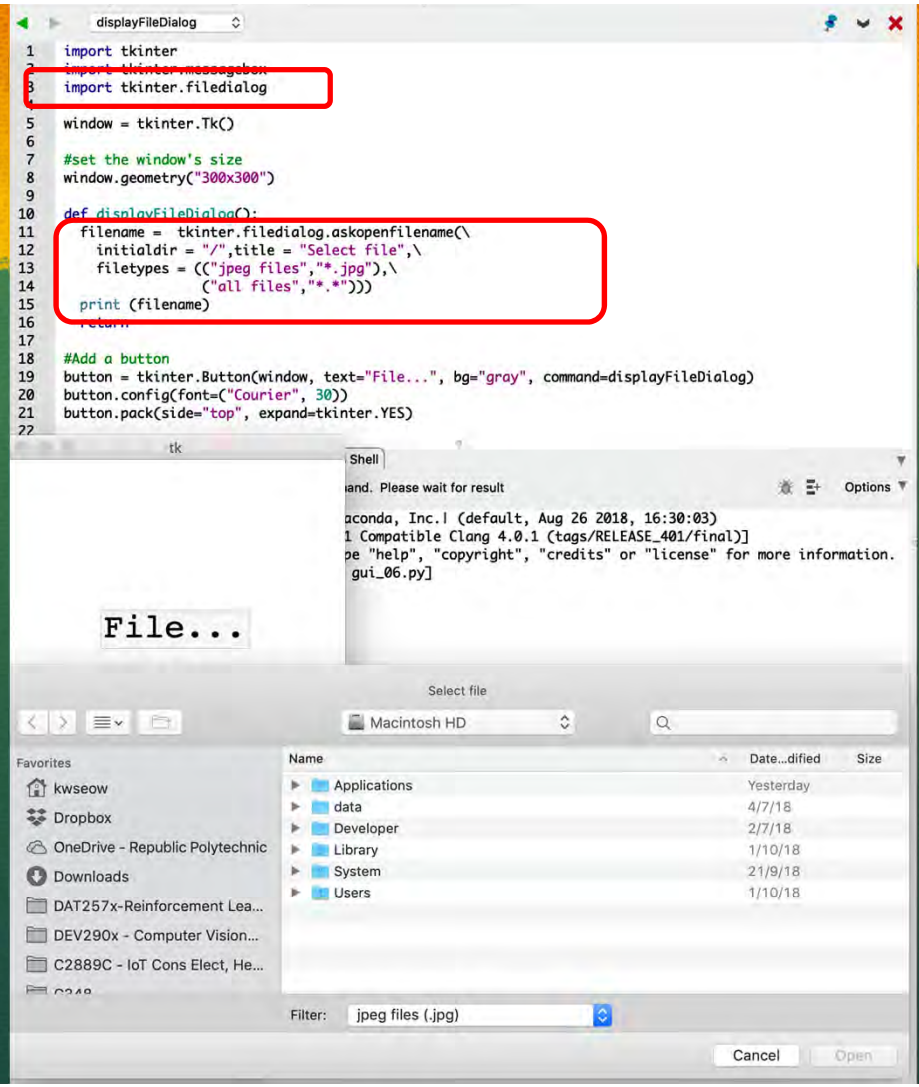


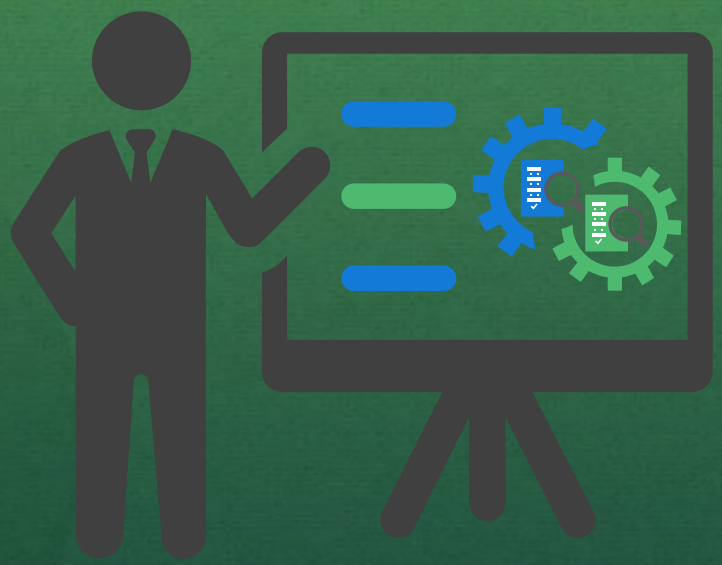
Entry Widget

```python
import tkinter
import tkinter.messagebox

window = tkinter.Tk()

#set the window's size
window.geometry("300x300")

def show_answer():
    Ans = int(num1.get()) + int(num2.get())
    print(Ans)
    ans.insert(0,Ans)

label1 = tkinter.Label(window, text = "Enter Num 1:").grid(row=0)
label2 = tkinter.Label(window, text = "Enter Num 2:").grid(row=1)
label3 = tkinter.Label(window, text = "The Sum is:").grid(row=2)

num1 = tkinter.Entry(window)
num2 = tkinter.Entry(window)
ans = tkinter.Entry(window)

num1.grid(row=0, column=1)
num2.grid(row=1, column=1)
ans.grid(row=2, column=1)

#Add a button
button1 = tkinter.Button(window, text="Show", bg="gray", command=show_answer)
button1.grid(row=4, column=0)

window.mainloop()
```



Filedialog


displayFileDialog
import tkinter
import tkinter.filedialog
window = tkinter.Tk()
#set the window's size
window.geometry("300x300")
def displayFileDialog():
filename = tkinter.filedialog.askopenfilename(\
initialdir = "/",title = "Select file",\
filetypes = (("jpeg files","*.jpg"),\
("all files","*.*")))
print (filename)
#Add a button
button = tkinter.Button(window, text="File...", bg="gray", command=displayFileDialog)
button.config(font=("Courier", 30))
button.pack(side="top", expand=tkinter.YES)
tk
File...
Shell
Options
gui_06.py]
Select file
Macintosh HD
Favorites
kwseow
Dropbox
OneDrive - Republic Polytechnic
Downloads
DAT257x-Reinforcement Lea...
DEV290x - Computer Vision...
C2889C - IoT Cons Elect, He...
Name
Date...dified
Size
Applications
Yesterday
data
4/7/18
Developer
2/7/18
Library
1/10/18
System
21/9/18
Users
1/10/18
Filter:
jpeg files (.jpg)
Cancel
Open



Day 1 Summary

- ✓ *Basics on Python*
- ✓ *Development Environment*
- ✓ *Datatypes (basic, list, dictionary)*
- ✓ *Printing*

- ✓ *Functions*
- ✓ *Time library*
- ✓ *Basic Arithmetic*
- ✓ *Getting user inputs*

- ✓ *If-Else statement*
- ✓ *For Loops*
- ✓ *Exception handling*
- ✓ *File management*
- ✓ *Graphical User Interface*

Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Github
http://bit.ly/2SxPbGk