Lesson 2 – Python basics

- S.P. Chong

Objectives

- In this lesson, you will be introduced to the Python programming language and learn the **basics** of the language.

- The topics include
  - Comments, strings, operators
  - Data types: lists, tuples, dictionaries
  - Flow control: for loop, while loop, if-else
  - Functions, importing modules
  - Optional: OOP (Object Oriented Programming)

- At the end of this lesson, you will be able to write Python programs to solve some real problems!

Python programming language

Taught in A Level Computing

Highly portable – Windows, Mac, Linux.

Simple to use, high level programming language.

Extensive libraries.

Interpreted language – may not be as fast as a compiled language.

One of the top 10 programming languages, along with C, Java etc…

Named after the BBC show “Monty Python’s Flying Circus” – nothing to do with reptile.

Everything is an object. We will not cover encapsulation, inheritance, polymorphism in details in this elective.

Python programming language (cont.)

What is meant by…

- High level programming language?
- Interpreted language, vs compiled language?

If an **interpreted** language is used, an "Interpreter" will take one line of the high-level language program, convert it to machine code, execute it, and then repeat.

if button is pressed
switch on LED…

10110010101100010
…

The **compilation** process

Python programming language (cont.)

What other languages are in the top 10 lists?

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

In practice, the choice of a programming language is often dictated by other real-world constraints such as cost, availability, training, and prior investment, or even emotional attachment.

https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

Python programming language (cont.)

Who use Python?

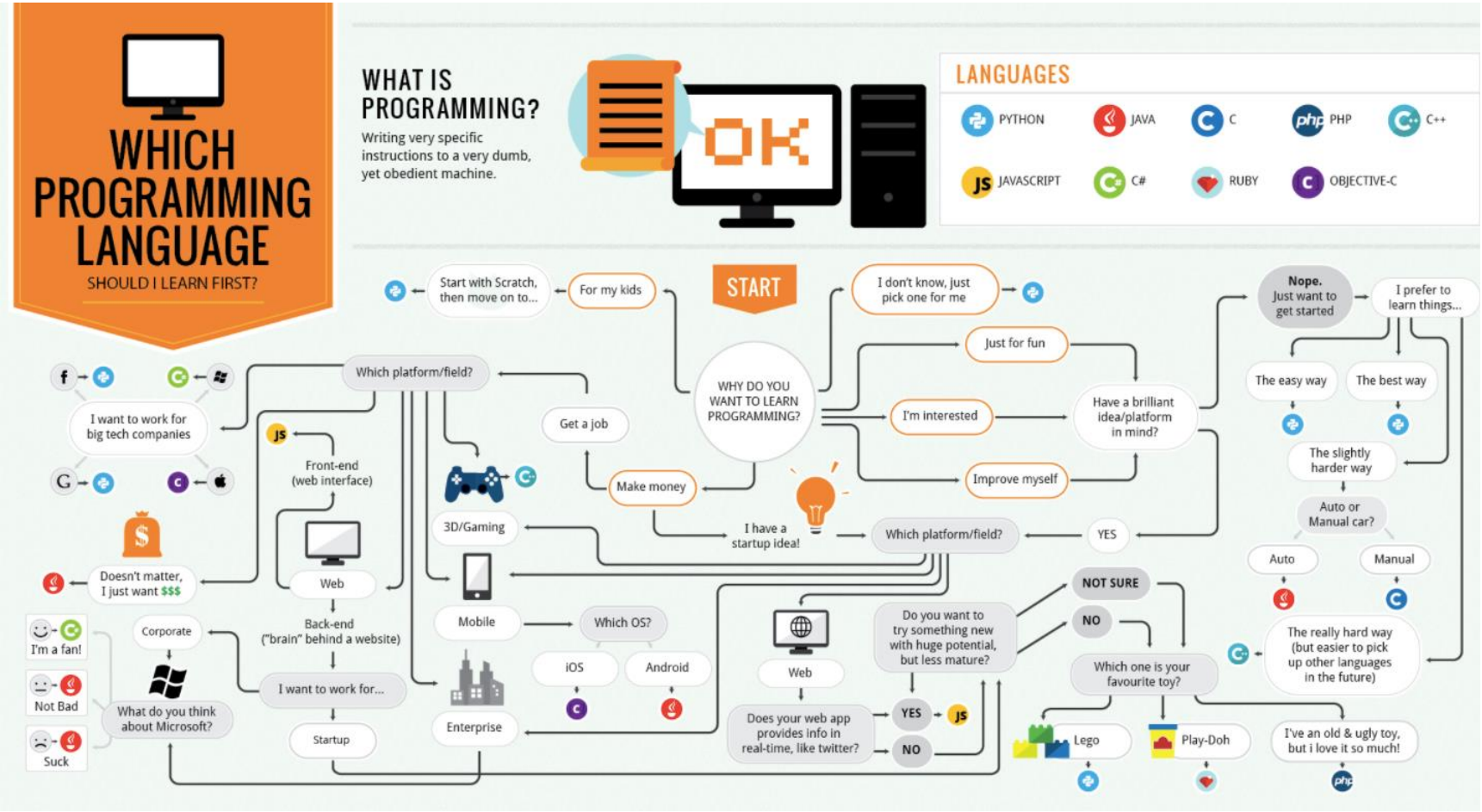
Google

Verity Ultraseek

http://www.c-sharpcorner.com/article/how-python-is-different-from-other-languages/

Python programming language (cont.)

Python is good for…



https://medium.freecodecamp.org/what-programming-language-should-i-learn-first-ɟdɪɹɔsɐʌɐɟ-ɹəʍsuɐ-19a33b0a467d

Python programming language (cont.)

Python vs other languages…

https://www.python.org/doc/essays/comparisons/

Java

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing. For example, a Python programmer wastes no time declaring the types of arguments or variables, and Python's powerful polymorphic list and dictionary types, for which rich syntactic support is built straight into the language, find a use in almost every Python program. Because of the run-time typing, Python's run time must work harder than Java's. For example, when evaluating the expression a+b, it must first inspect the objects a and b to find out their type, which is not known at compile time. It then invokes the appropriate addition operation, which may be an overloaded user-defined method. Java, on the other hand, can perform an efficient integer or floating point addition, but requires variable declarations for a and b, and does not allow overloading of the + operator for instances of user-defined classes.

For these reasons, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language. In fact, the two together make an excellent combination. Components can be developed in Java and combined to form applications in Python; Python can also be used to prototype components until their design can be "hardened" in a Java implementation. To support this type of development, a Python implementation written in Java is under development, which allows calling Python code from Java and vice versa. In this implementation, Python source code is translated to Java bytecode (with help from a run-time library to support Python's dynamic semantics).

Comments, strings, operators

- **Comments** are used to make a program easier to understand.
- In Python, a single line comment is preceded by a # sign:
  - e.g. #This is a single line comment
- Comments will not be interpreted, so won't affect how a program runs.

- **String** is like a sentence.
- In Python, a string is enclosed by a pair of single or double quotation marks:
  - e.g. 'Hello world!'

Comments, strings, operators (cont.)

Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> #This is a single line comment
>>> 'Hello World!'
'Hello World!'
>>> print("Hello World again!")
Hello World again!
>>> ans='I don\'t know'
>>> ans
"I don't know"
>>> ans2="I also don't know"
>>> ans2
"I also don't know"
>>>
```

- This screen capture shows a comment & some strings.
- Note:
  - the use of forward slash \ as an escape character.
  - what happens when a string is printed.
  - Interchangeability of ‘ and “.

Comments, strings, operators (cont.)

- These are the **arithmetic operators**:

Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 5+2
7
>>> 5-2
3
>>> 5*2
10
>>> 5/2
2.5
>>> 5%2
1
>>> 5**2
25
>>> |
```

Comments, strings, operators (cont.)

- These are the **relational operators**:

```
Python 3.5.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 5==2
False
>>> 5!=2
True
>>> 5>5
False
>>> 5>=5
True
>>> |
```

Comments, strings, operators (cont.)

- These are the **logical operators**:

```
Python 3.5.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> True and False
False
>>> True and True
True
>>> True or False
True
>>> False or False
False
>>> not True
False
>>> not False
True
>>> |
```

Data types (integers, floating point numbers)

- You are probably familiar with **integers** & **floating point numbers**.
- A number can be “type-cast” to be integer or floating point.

Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> ans=5/2
>>> ans
2.5
>>> type(ans)
<class 'float'>
>>> int(ans)
2
>>> ans2=int(5/2)
>>> ans2
2
>>> type(ans2)
<class 'int'>
>>>
```

One of the powerful features of Python is “implicit typing”. This means no prior declaration is needed before a variable is used.

Data types (lists) (cont.)

- A **list** is enclosed within a pair of **square brackets** [ ].
- The **items** in a list are separated by commas.
- The items in a list can be of **mixed type**. The example used (myfruits) however, is a list of strings.
- The **index** starts with 0.
- The last item has index value of -1.

```
Type "copyright", "credits" or "license()" for more information.
>>> myfruits=['apple','banana','coconut','durian']
>>> myfruits
['apple', 'banana', 'coconut', 'durian']
>>> for fruit in myfruits:
        print(fruit)


apple
banana
coconut
durian
>>> myfruits[0]
'apple'
>>> myfruits[2]
'coconut'
>>> myfruits[-1]
'durian'
>>> myfruits[-3]
'banana'
>>> 'banana' in myfruits
True
>>> 'orange' in myfruits
False
>>> len(myfruits)
4
>>> myfruits.append('Entawak')
>>> myfruits
['apple', 'banana', 'coconut', 'durian', 'Entawak']
>>> myfruits.remove('durian')
>>> myfruits
['apple', 'banana', 'coconut', 'Entawak']
>>> myfruits.pop(3)
'Entawak'
>>> myfruits
['apple', 'banana', 'coconut']
>>> myfruits.insert(2,'Zucchini')
>>> myfruits
['apple', 'banana', 'Zucchini', 'coconut']
>>> myfruits.count('banana')
1
>>> |
```

Data types (lists) (cont.)

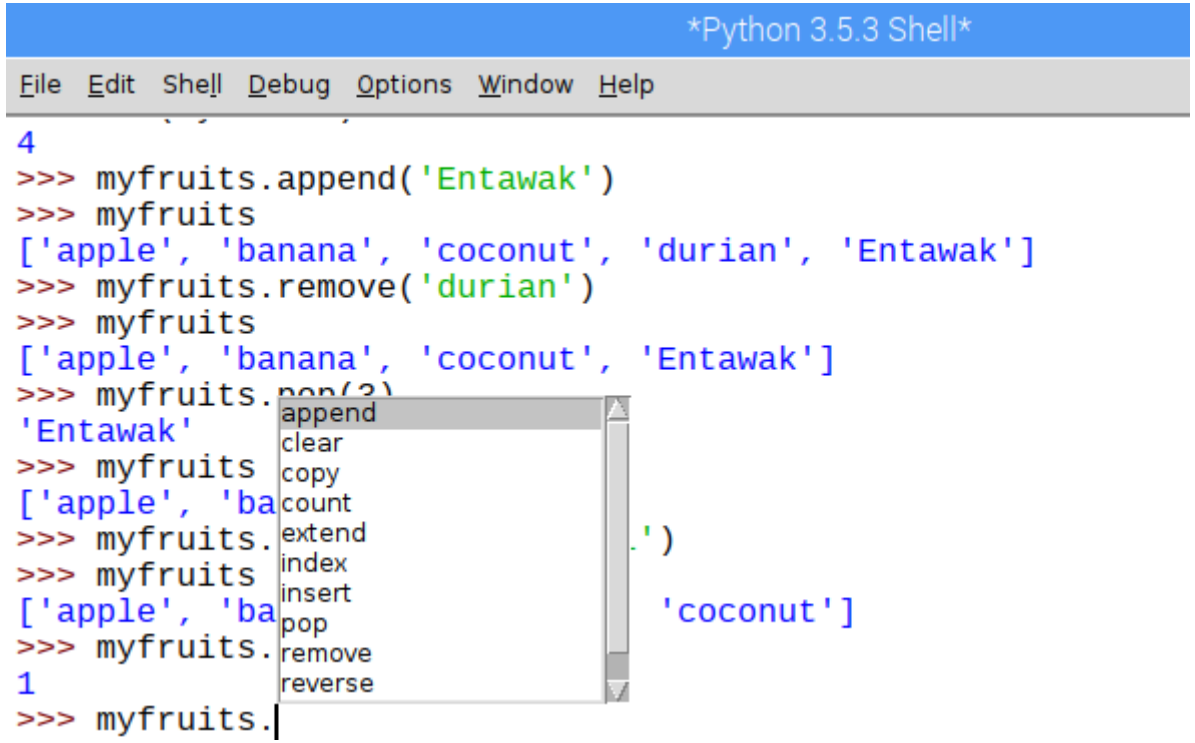
- Note how these are used:
  - **in**
  - **len**
  - **append**
  - **remove**
  - **pop**
  - **insert**
  - **count**

```
Type "copyright", "credits" or "license()" for more information.
>>> myfruits=['apple','banana','coconut','durian']
>>> myfruits
['apple', 'banana', 'coconut', 'durian']
>>> for fruit in myfruits:
        print(fruit)


apple
banana
coconut
durian
>>> myfruits[0]
'apple'
>>> myfruits[2]
'coconut'
>>> myfruits[-1]
'durian'
>>> myfruits[-3]
'banana'
>>> 'banana' in myfruits
True
>>> 'orange' in myfruits
False
>>> len(myfruits)
4
>>> myfruits.append('Entawak')
>>> myfruits
['apple', 'banana', 'coconut', 'durian', 'Entawak']
>>> myfruits.remove('durian')
>>> myfruits
['apple', 'banana', 'coconut', 'Entawak']
>>> myfruits.pop(3)
'Entawak'
>>> myfruits
['apple', 'banana', 'coconut']
>>> myfruits.insert(2,'Zucchini')
>>> myfruits
['apple', 'banana', 'Zucchini', 'coconut']
>>> myfruits.count('banana')
1
>>> |
```

Data types (lists) (cont.)

- The Python IDLE provides "**context sensitive**" help – all the methods associated with an object (a list in this case) will be shown for you to choose.
- You may want to find out what **extend / reverse / sort** will do to the list?



Data types (tuples) (cont.)

- A **tuple** is a list with **immutable** contents.
- A tuple is enclosed within a pair of **round brackets** ( ).

Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> weekdays=('Mon','Tue','Wed','Thu','Fri')
>>> weekdays
('Mon', 'Tue', 'Wed', 'Thu', 'Fri')
>>> weekdays[3]
'Thu'
>>> print(weekdays[-1])
Fri
>>> |
```

Data types (dictionaries) (cont.)

- A **dictionary** is a list of **key-value** pairs.
- A dictionary is enclosed within a pair of **curly brackets** { }.
- Note how a key-value pair can be added or deleted.

Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> tel={'Alice':98761234,'Bernard':88884444,'Collin':99887700}
>>> tel
{'Alice': 98761234, 'Collin': 99887700, 'Bernard': 88884444}
>>> tel['Collin']
99887700
>>> tel['Diana']=89868689
>>> tel
{'Alice': 98761234, 'Diana': 89868689, 'Collin': 99887700, 'Bernard': 88884444}
>>> tel.keys()
dict_keys(['Alice', 'Diana', 'Collin', 'Bernard'])
>>> 'Alice' in tel
True
>>> del tel['Bernard']
>>> tel
{'Alice': 98761234, 'Diana': 89868689, 'Collin': 99887700}
>>>
```

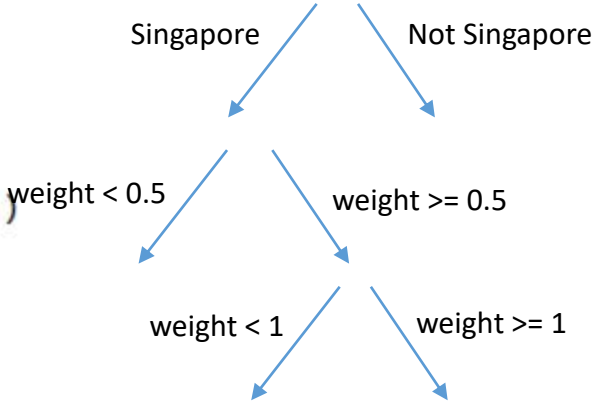
Flow control (if-else)

- **If** is used to control which "**branch**" of the code will be executed.
- In this example, how many branches are there?

if_else.py - /home/pi/if_else.py (3.5.3)

File Edit Format Run Options Window Help

```
destination=input('Enter destination: ')
weight=float(input('Enter weight:' )) #type cast to float
print('Destination =',destination,',Weight =',weight)
if destination=='Singapore':
    if weight<0.5:
        print('Free shipping')
    elif weight<1:
        print('Shipping = $2')
    else:
        print('Shipping = $5')
else:
    print('Sorry, can\'t deliver to that destination')
```



Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
========================= RESTART: /home/pi/if_else.py ===========
Enter destination: Singapore
Enter weight:0.987
Destination = Singapore , Weight = 0.987
Shipping = $2
>>>
```

Flow control (for loop)

- A **for loop** allows an action to be repeated a certain number of times.
- Python uses what is called (**inclusive, exclusive**) notation.

Python 3.5.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> for x in range(5):
        print(x)


0
1
2
3
4
>>> for y in range(3,6):
        print(y)


3
4
5
>>> for z in range(3,14,4):
        print(z)


3
7
11
>>>
```

Same as range(0,5)

Inclusive-exclusive

Start, end, increment

Flow control (for loop) (cont.)

- Note the effect of **continue**.

Python 3.5.3 Shell

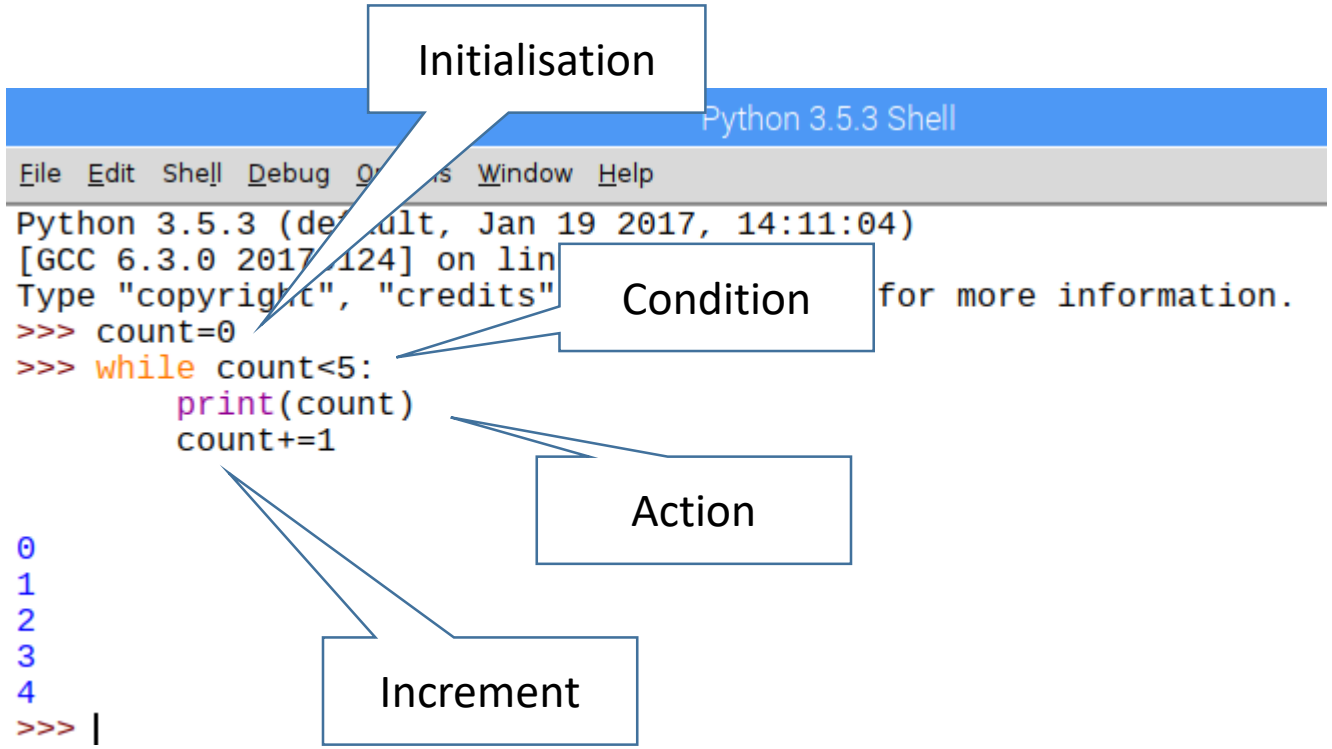
File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> for x in range(10):
        if x%3==0:
                continue
        print(x)


1
2
4
5
7
8
>>> |
```

Flow control (while loop)

- **While** loop can also be used to repeat an action a certain number of times.
- When should you use for loop, and when while loop?



Flow control (while loop) (cont.)

- Note the effect of **break**.

Python 3.5.3 Shell

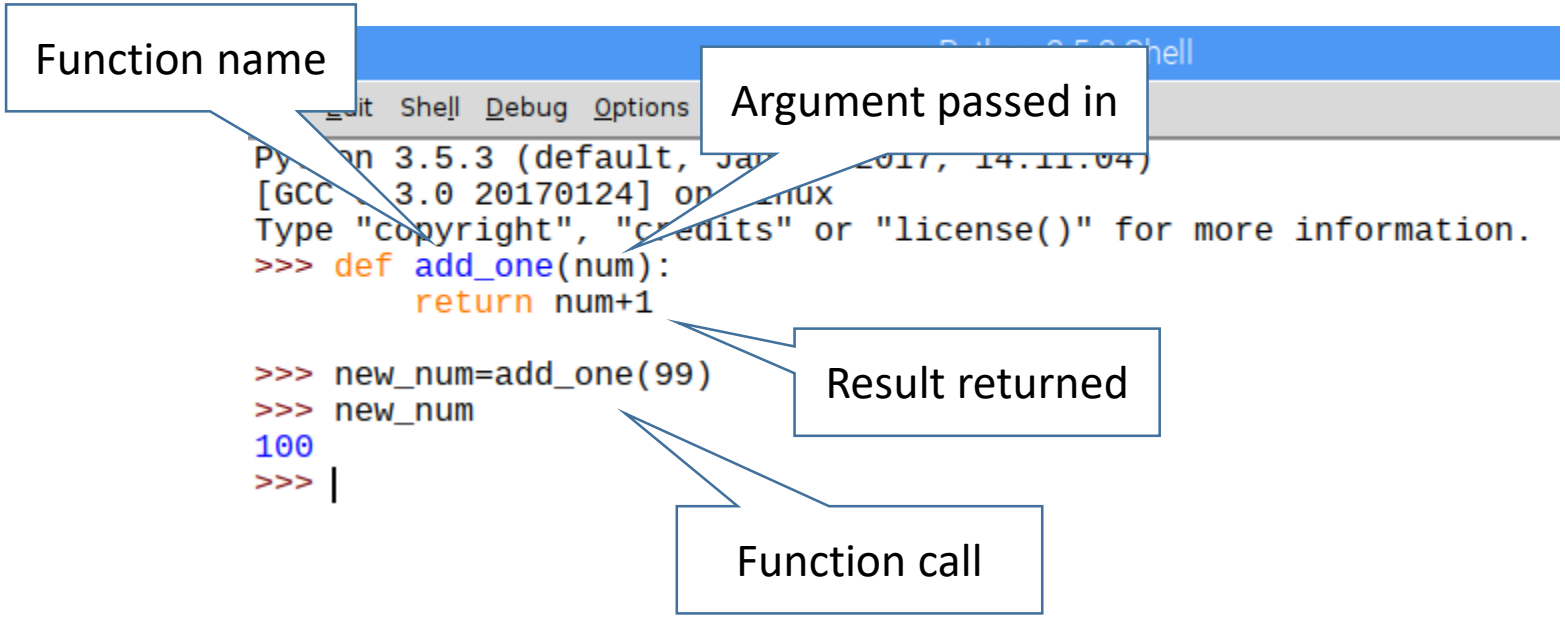
File Edit Shell Debug Options Window Help

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> count=0
>>> while True:
        print(count)
        count+=1
        if count>=5:
                break


0
1
2
3
4
>>> |
```

Functions & Importing modules

- **Functions** are written to allow a chunk of code to be reused.
- When defining function, use a meaningful **name**, be sure what the function is supposed to do, what **arguments** to pass in, what result is **returned**.



Functions & Importing modules(cont.)

- Many useful **modules** or **libraries** have been written by others.
- By importing these modules, we allow a chunk of code written by others to be reused i.e. we will not be reinventing the wheels.

random module

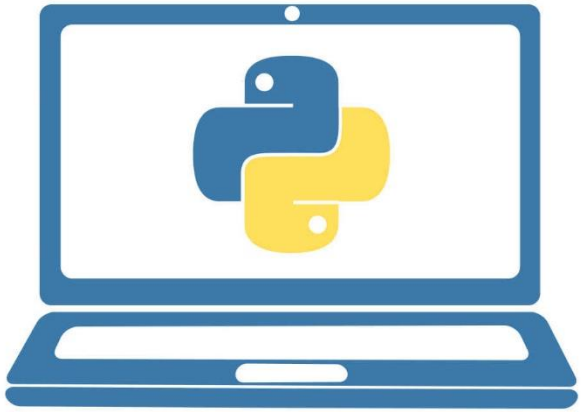
Class, instance, attribute, method are all OOP (Object Oriented Programming) concepts – please read the separate deck of slides for more info.

```
import random
print('Buy these numbers, sure strike!')
for x in range(6):
    print(random.randint(1,50))
```

A method of the random class

```
Python 3.5.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
=================== RESTART: /home/pi/importing_module.py ========
Buy these numbers, sure strike!
34
28
18
9
23
27
>>> |
```

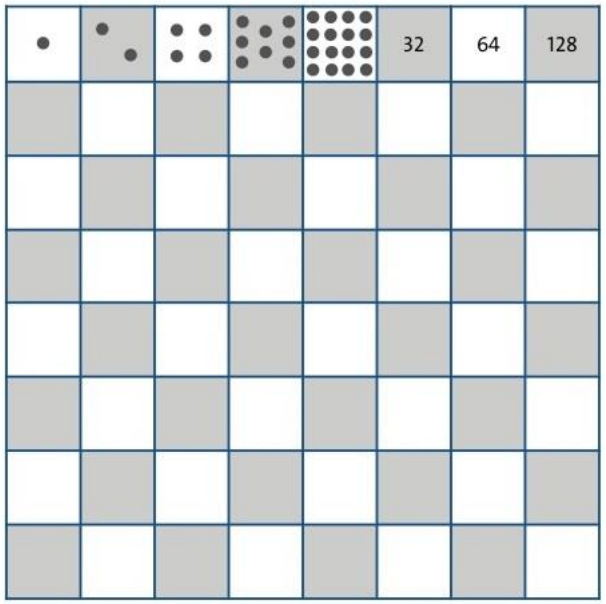
Lab Exercises



- Exercise 2.1 – The king’s rice grains
- Exercise 2.2 – Multiplication table
- Exercise 2.3 – Will I be rich?
- Exercise 2.4 – Useful Python modules

Exercise 2.1 – The king's rice grains

An ancient king was a big chess enthusiast and had the habit of challenging wise visitors to a game of chess. One day a traveling sage was challenged by the king. To motivate his opponent the king offered any reward that the sage could name. The sage modestly asked just for a few grains of rice in the following manner: the king was to put a single grain of rice on the first chess square and double it on every consequent one.
Having lost the game and being a man of his word the king ordered a bag of rice to be brought to the chess board. Then he started placing rice grains according to the arrangement: 1 grain on the first square, 2 on the second, 4 on the third, 8 on the fourth and so on.



Write a Python program to compute how many grains of rice has to be placed in the 64th square.

http://www.singularitysymposium.com/exponential-growth.html

Hints: 1. use exponential
2. answer: 9223372036854775808

Exercise 2.2 – Multiplication table

Write a Python program to prompt the user for an integer (e.g. 5) and print out the multiplication table for that integer, for example:

Enter an integer: 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
.
.
.
5 x 12 = 60

Hints: 1. use for loop
2. Python code for reading user's input is:
num1 = input ("Enter an integer: ")
3. To typecast an input into an integer, use int (num1)

Exercise 2.3 – Will I be rich?

Assuming you start working at 25 years old, and your starting salary is $3000 and get 3 month of bonus every year. Assuming also that you save 30% of what you earn every year and invest it (for a 10% return) at the beginning of the following year. Assuming further that your pay goes up by 10% every year.

Write a Python program to determine when you will have your first one million dollars.

Hints: 1. use while loop & if-else
2. declare variables required (Age, Annual_salary, Annual_saving, Invest_in_Jan, Amount_in_Dec)

Starting pay x (12+3)

x 30%

x 110%

x 110%

Age	Annual salary	Annual saving	Invest in Jan	Amount in Dec
25	$45,000	$13,500		
26	$49,500	$14,850	$13,500	$14,850
27	$54,450	$16,335	$29,700	$32,670
28	$59,895	$17,969	$49,005	$53,906
29	$65,885	$19,765	$71,874	$79,061
30	$72,473	$21,742	$98,827	$108,709
31	$79,720	$23,916	$130,451	$143,496
32	$87,692	$26,308	$167,413	$184,154
33	$96,461	$28,938	$210,461	$231,508
34	$106,108	$31,832	$260,446	$286,491
35	$116,718	$35,016	$318,323	$350,155
36	$128,390	$38,517	$385,171	$423,688
37	$141,229	$42,369	$462,205	$508,425
38	$155,352	$46,606	$550,794	$605,874
39	$170,887	$51,266	$652,479	$717,727
40	$187,976	$56,393	$768,993	$845,893
41	$206,774	$62,032	$902,286	$992,514
42	$227,451	$68,235	$1,054,546	$1,160,001
43	$250,196	$75,059	$1,228,236	$1,351,060

Exercise 2.4 – Useful Python modules

There are many Python modules which a programmer can import and use.

Use internet search to name the modules used for the following:

_____ : for sending / receiving data to / from the internet

_____ : for developing a web server

_____ : for creating a graphical user interface (GUI)

_____ : for computing statistics e.g. mean, variance, standard deviation

https://www.element14.com/community/groups/internet-of-things/blog/2017/02/17/iot-with-python-essential-packages


Thank
You

Appendix - Python “cheat sheet”

```
# This is a single line comment

''' This is a multiple-
line comment '''
```

Arithmetic operators:

```
5 + 2
5 - 2
5 * 2
5 / 2
5 % 2
5 ** 2
```

Relational operators:

```
count == 2
num != 3
duration > 4
total >= 5
```

Logical operators:

```
condition1 and condition2
condition3 or condition4
not condition5
```

Data types:

```
Ans = 5 / 2   # integer division, but result is float!
Num = 5.0 / 2  # float

Myfruits = ['apple', 'orange', 'durian']
 # lists: items can be accessed, added or removed

DaysOfWeek = ('Sun', 'Mon', 'Tue', 'Wed', \
  'Thu', 'Fri', 'Sat')   # tuples – immutable contents

PhoneBook = {'Jack':9876543, 'Jill':8888421}
 # dictionaries: items can be accessed, added or removed
```

Python “cheat sheet” (cont.)

```
for x in range (0, 5):
  print (x)

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

if  num1 > num2:
  print (‘num1 is bigger’)
elif num2 > num1:
  print (‘num2 is bigger’)
else:
  print(‘They are the same’)
```

```
# function definition
def add_one (num):
   return num+1

# function call
new_num = add_one(99)
print(new_num)
# should get 100
```

```
# importing modules / libraries
import random
print (‘Buy these, sure strike:’)
for x in range (0, 6):
  print (random.randint(1, 50))
```