Python

Introduction

What is a computer?

- a machine that stores and manipulates information under the control of a changeable program
- input -> process/storage -> output.

What is Computing?

- the study of computers (wrong)

What is a computer program?

- a detailed, step by step set of instructions telling a computer what to do

An important lesson of Computing

- software (program) controls hardware (physical machine)

Programming

- the process of creating software
- a challenging activity
  - requires the ability to see the big picture while paying attention to the minute details

Why learn programming?

- fundamental of becoming a computing professional
- understand strengths and limitations of computers
- true master (creators) and not slaves (users) of computers
- develop problem solving skills (analyzing complex systems)
- create innovative and effective solutions
- computers are ubiquitous, having the ability to understand and program computers will be beneficial, regardless of vocation

Programming languages

Fact: Computers cannot understand China/England.

Fact: Computers can only do one thing, but they do it super fast.

Solution: Need for computer programming language(s).

Note:

- programming languages are used by human beings to write instructions for the computer
- these instructions (in the form of a computer program) are not directly usable by computers
- computer will translate computer program into a form which it can understand and execute

Python

http://python.org

- general purpose
- cross-platform
- easy to learn
- powerful
- object-oriented
- interpreted vs compiled

- one of the 3 official languages used by Google

Software

- Python IDE (Integrated Development Environment)
-

- download and install
- launch IDLE

- program source code extension - .py

Output

print()

eg

```
print("Good morning!")

name = "Lim Ah Seng"
print(name)

print("Good morning " + name)
```

Data Types

- integer: int
- floating point numbers / numbers with fractional parts: float
- text string: str

- to determine type, use type
- eg
  - x = 5, type(x) returns int
  - y = 1.2, type(y) returns float
  - z = "aaa", type(z) returns str

Strings

- can be single, double or triple quoted

eg
'Lim Ah Seng', "Lim Ah Seng", '''Lim Ah Seng''' are all valid

- triple quote for multi-line strings

eg
''' Good morning boys and girls,
Life is hard.
Why not make it harder. '''

Strings: functions

- many useful string functions

eg superhero = "Batman"

len(superhero) returns 6
superhero.upper() returns "BATMAN"

Strings: Slicing

eg superhero = "Batman"

superhero[0] returns "B"
superhero[5] returns "n"
superhero[6] gives error!
superhero[-1] returns "n"

superhero[0:3] returns "Bat" # or [:3]
superhero[3:6] returns "man" # or [3:]

Saving your programs for future (re)use

- whatever is typed at IDLE prompt is not saved
- to save program for future use
- File -> New Window
- save as .py extension

Logical operations

```
==              # equal
!=              # not equal
<               # less than
<=              # less than and equal
>               # greater than
>=              # greater than and equal
is              # equality for objects
is not          # inequality for objects
```

Input

input() eg

```
x = input("Enter a number:")
print(x)

name = input("Enter your name")
print(name)
```

Assignment

- set the evaluated value on the right hand side to the variable on the left hand side

```
x = 1      # right hand side value is 1, so x is assigned 1

x = 1 + 2 # x is assigned 3

x = y + z # x is assigned sum of y and z

x = x + 1 # x is assigned old value of x incremented by 1
```

Arithmetic operations

```
+                   # add
-                   # subtract
*                   # multiply
/                   # divide
//                  # quotient
%                   # remainder
**                  # exponent
abs()               # absolute
int()               # integer
float()             # float
pow()               # power
```

Boolean operations

```
not                 # negate
and                 # both must be true to return true
or                  # at least one true to return true
```

Formatting

<placeholder string>.format(<values>)

eg

format 123 as integer, numbers are right justified

```
"{0}".format(123)
```

format 123.45 as floating point number using 10 columns with 1 decimal place and left justified

```
"{0:<10.1f}".format(123.45)
```

format "123" as string using 8 columns, strings are left justified

```
"{0:8s}".format("123")
```

Formatting (contd)

```
name = "Lim Ah Seng"
```

format name as string using 20 columns

```
"{0:20s}".format(name)
```

format name as string using 20 columns, right justified

```
"{0:>20s}".format(name)
```

Formatting (contd)

```
name = "Lim Ah Seng"
age = 17

# format string and number
"{0:20s} {1}".format(name, age)
```

Q: What does the following do?

```
"{0:20s} {1:5.2f}".format(name, age)
```

Formatting (summary)

- < left justify
- > right justify
- s string
- f floating point number
- 5.2f floating point number with 5 column width (include decimal point and decimal places) and 2 decimal places

Reserved words

- words with a special meaning in the language and cannot be used as program identifiers

```
>>> import keyword
>>> print(keyword.kwlist)

False class finally is return None continue for
lambda try True def from nonlocal while and del
global not with as elif if or yield assert else
import pass break except in raise
```

http://docs.python.org/py3k/reference/lexical_analysis.html#keywords

Comments

begins with # eg

```
# This is a comment
```

- comments are ignored by Python
- meant for human beings to help them understand program code
- use appropriate annotation (do not repeat code)

Conditional Statement

if <condition>:
    <action1>
[else:]
    [<action2>]

eg

```
a = 3
b = 2
if a > b:
    print("greater")
```

Conditional Statements (contd)

- else

```
a = 3
b = 2
if a > b:
    print("greater")
else:
    print("less than")
```

- nested if-else

```
a = 3
b = 2
if a > b:
    print("greater")
elif a < b:
    print("less than")
else:
    print("equal")
```

Importing from modules

- Python provides many modules useful for program development

eg

```
import math

print(math.pi)
```

or

```
from math import pi

print(pi)
```

Exercise

Write a program to get the following input from the user

- weight in kg
- height in m

Compute and output the body mass index (BMI) correct to 2 decimal places.

Display an appropriate message depending on the user's BMI value.

Reference: http://www.hpb.gov.sg/hpb/default.asp?TEMPORARY_DOCUMENT=1769&TEMPORARY_TEMPLATE=2

Practical 1

http://go.dhs.sg/y5cplab1

Due: 28 Jan 2013 (Generally 1 week after posting)

Resources

https://developers.google.com/edu/python

In Python IDE, Help -> F1

http://diveintopython3.org