

PRG1



**NGEE ANN**  
SCHOOL OF INFOCOMM TECHNOLOGY

W  
E  
E  
K  
  
1

# Introduction to Python

## **Programming I (PRG1)**

Diploma in Information Technology

Diploma in Financial Informatics

Diploma in Information Security & Forensics

Year 1 (2018/19), Semester 1

# Objectives

At the end of this lecture, you will be able to

- Use IDLE for programming Python
- Recognize and use the components in Python
- Understand the usage of variables in Python

# PYTHON

## The Zen of Python

Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than \*right\* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!

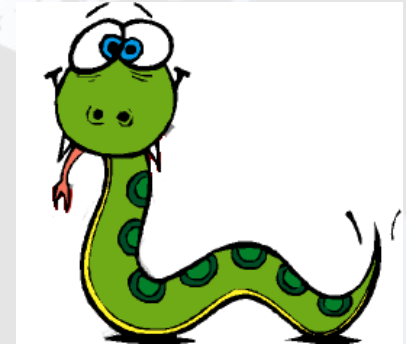
# What is Python?

- A programming language with a set of libraries
- Developed in early 1990s by Guido van Rossum
- Characteristics:
  - Easy to read and learn
  - Clean look with few unnecessary symbols
  - Easy to keep up to date



# What is Python useful for?

- Scripting
  - Short programs to perform administrative tasks
- Website development
  - E.g.: Django, Zope
- Text processing
  - Handling text and files
- Education
  - Fast becoming the first language to learn around the world (EG: NUS, Cambridge, CMU, MIT, etc...)



# IDLE

- Simple editor for beginners
  - Packaged with Python installation
  - Others exist: Notepad++, Sublime, etc.



**Let's install Python!**

# Using IDLE

- Let's do this together
  1. Launch IDLE
  2. Choose 'File' -> 'New Window' (or CTL-N)
  3. Type the following **`print('Welcome to ICT!')`**
  4. Save your program by choosing 'File' -> 'Save' (or CTL-S).
    - Save on your desktop as 'welcome.py'
  5. Run your program by choosing 'Run' -> 'Run Module' (or F5)
    - A Python shell will appear with your statement
- Note:
  - Python is case-sensitive. 'A' is different from 'a'
  - Check your codes carefully



# IDLE

- Useful shortcuts

Command	What it does
CTL-N	Opens new editor window
CTL-O	Opens file for editing
CTL-S	Save current program
F5	Run current program
CTL-Z	Undo last action



# Syntax & Programming Structure

- Syntax (Rules)

- Languages have **rules**, e.g. English: full-stop to end sentence, commas for breaks in sentence, etc.
- Programming languages also have rules or **syntax**.

- Program Structure (Format)

- Languages have formats, e.g. writing letters, writing memos, reports, etc.
- Programming languages also have formats or **program structure** to follow when writing a program

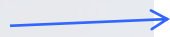
The background of the slide features a stylized, semi-transparent image of a globe showing the continents of Africa and Europe. To the left of the globe, there is a faint, perspective-view image of a computer keyboard. The title 'COMPONENTS IN PYTHON' is centered in the lower half of the slide in a bold, blue, sans-serif font.

# COMPONENTS IN PYTHON

# Identifiers

- Identifiers
  - are names defined by the programmer
  - are case-sensitive
  - begin with lowercase letter
    - cannot begin with a digit
    - contain only letters, digit and/or the underscore
  - should have meaningful names

identifier



```
pi = 22/7  
print (pi)
```

# Reserved Words

- Reserved Words
  - Special words not allowed for use as identifiers
  - Reserved by Python for special use

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	

# Statements

- Statements in Python are separated by a new-line character (return key)
- If the statement is too long, you can use a backslash ( \ ) to continue the statement to the next line

```
print('welcome to ICT!')  
print('this statement is way too looooooooooo \'  
      'nnnnnnng')
```




```
welcome to ICT!  
this statement is way too loooooooooonnnnnnnng  
>>> |
```

# Comments

- Explains meaning or logic of code
  - Comments are ignored by computer
- Good practice to include following in comments
  - Description of program with date and version number
  - Programmer's name and relevant details
- For single-line comment, type after the # sign
- For multi-line comment, enclose in triple-quote

```
''' this program prints a welcome message.  
    program created by ICT staff '''  
print('welcome to ICT!')    #this is the print statement
```

comments





# Understanding Variables

## Consider this problem

Write a computer program that calculates the BMI of a person. Display result on the screen.

- Does this computer program need to remember information?
- What information does this program need to remember?



# Understanding Variables

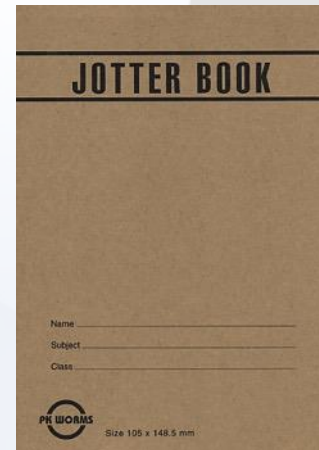
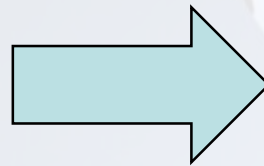
## Consider this problem

Write a computer program that calculates the BMI of a person. Display result on the screen.

- Where does the program store the needed information?

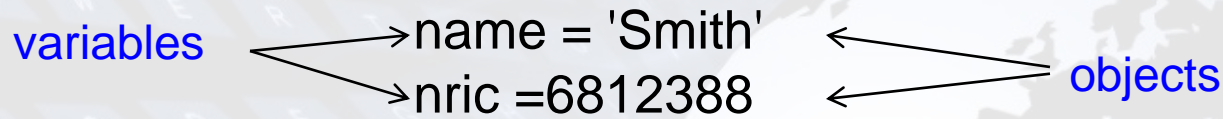
# Understanding Variables

- Programs use of variables to store data
- Variables are memory spaces in the computer's Random Access Memory (RAM)
- Imagine RAM as computer's jotter book to keep track of information



# Understanding Variables

- A variable has a type upon assignment



- The value has a data type
  - 'Smith' → str
  - 6812388 → int
- A variable is allocated a place in computer memory when you "**assign**" a value to it

# Data Types

- Basic data types used in Python

Data Type	Description	Example
<b>int</b>	integer	46
<b>float</b>	floating point or decimal numbers	123.45
<b>bool</b>	boolean	True or False
<b>str</b>	sequence of characters	'A', 'Hello'

# Understanding Variables

- Assign **value** to variable using statement
  - value can be a fixed value or outcome of a calculation, e.g. a multiplication product

*variable = value*

- Example:

*#variables assigned fixed value*

*weight = 55*

*height = 1.7*

*#variable assigned calculated value*

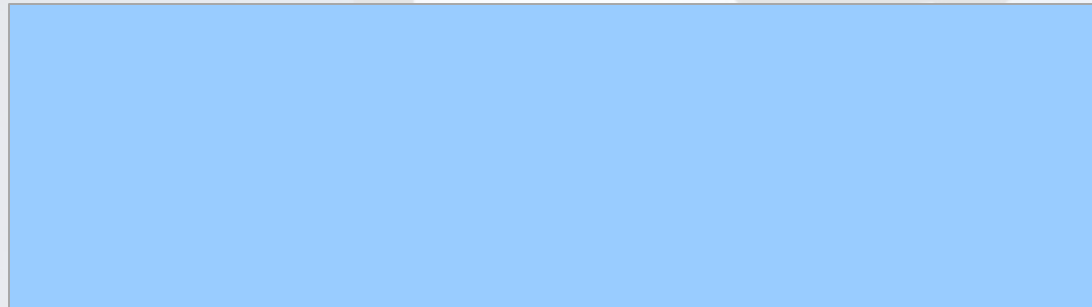
*bmi = weight / (height \* height)*

# Understanding Variables

- Displaying value of a variable
  - The value associated with the variable `bmi` is retrieved and concatenated (joined) to the string 'BMI='.

```
print('BMI =', bmi)
```

- Output:





# Activity 1

- Write a program to display 'Hip Hip Hurray' 2 times followed by 'Welcome to ICT' 3 times, on individual lines.

```
Hip Hip Hurray Hip Hip Hurray  
Welcome to ICT  
Welcome to ICT  
Welcome to ICT
```



# Activity 2

- Write a program to compute and display the square of 25. The output should be as follows:

**Answer of  $25 \times 25 = 625$**

- Requirement: Use variables for 25 and the result.

# Summary

- Introduction to Python
- Syntax & programming structures
- Identifiers and reserved words, statements, comments
- Variables and data types
- Simple output