PRG1

WEEK 1

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

Introduction to Python

Programming I (PRG1)
Diploma in Information Technology
Diploma in Financial Informatics
Diploma in Cybersecurity & Digital Forensics
Common ICT Programme
Year 1 (2019/20), 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

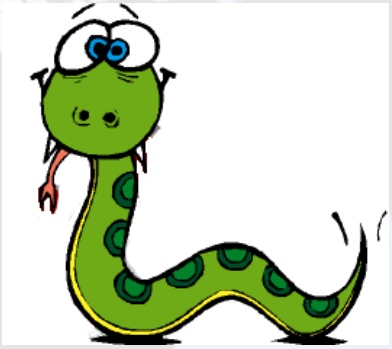
```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
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

Components in Python

Identifiers

- **Identifiers**
  - ✓ names defined by the programmer
  - ✓ 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

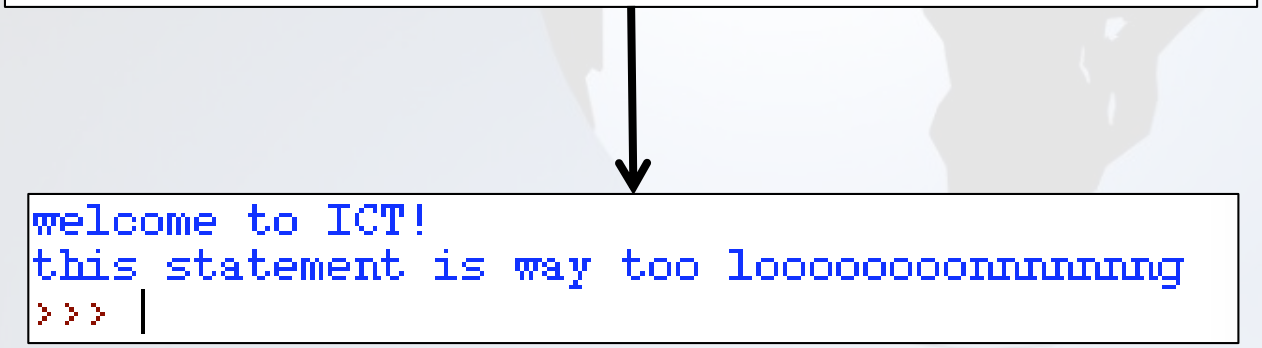
- ✓ Special words not allowed for use as identifers
- ✓ Reserved by Python for special use

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

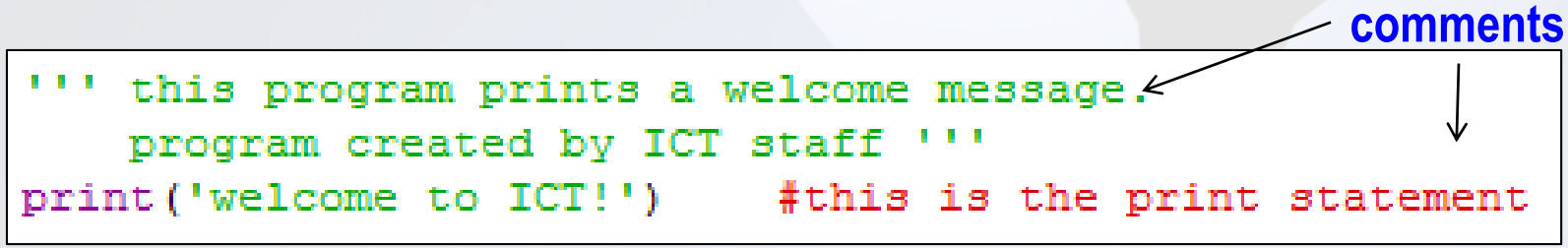
- **Statements in Python are separated by a new-line character (return key)**
  - ✓ If the statement is too long, you can use a backslash (\ ) to continue the statement to the next line

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



Comments

- **Explains meaning or logic of code**
  - ✓ Comments are ignored by computer
- **Good practice to include following 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 ' ' '**



Values & Data Types

- ✓ A value is one of the fundamental things — like a letter or a number — that a program manipulates.
- ✓ These values are classified into different data types
- ✓ Python has a function called *type* which can tell you the type of a value.

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

Values and data types

```
>>> type("Welcome to ICT")
<class 'str'>
>>> type(8)
<class 'int'>
>>> type(8.5)
<class 'float'>
```

- ***strings*** **belong to the class str and** ***integers*** **belong to the class int and** ***float*** **belongs to float**
  - ✓ At this stage, you can treat the words class and type interchangeably

Exercise 1

- **What is the datatype of the following?**

Program Code	Output
>>> print (type(7))	
>>> print (type("Welcome"))	
>>> print(type(False))	
>>> print(type(7.5))	
>>> print(type(12/17))	
>>> print(type(2.0/1))	

Exercise 1 (cont)

- **What is the datatype of the following?**

Program Code	Output
>>> print(type(11 ** 3))	
>>> print(type(2 == "2"))	
>>> a = str((-4 + 3 / 2 ** 3) + 321 - ((64 / 16) % 4) ** 2) >>> print(type(a))	

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?**

  Yes!

- **What information does this program need to remember?**

  values of :

  weight, height, calculated BMI

- **Where does the program store the needed information?**

  in **VARIABLES**

Program - CalBMI.py

```
# This program calculates the body mass index of a person

# Assign values to variables weight & height
weight = 55
height = 1.7

# Process calculation & assign to variable bmi
bmi = weight / (height * height)

# Display values of the variables
print('Your height is ' + str(height) + 'm')
print('Your weight is ' + str(weight) + 'kg')
print('Your bmi is ' + str(bmi))
```

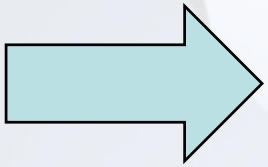
Output

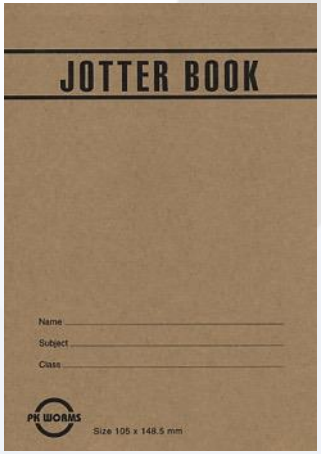
```
>>>
Your height is 1.7m
Your weight is 55kg
Your bmi is 19.031141868512112
>>>
```

What are the variables used in this program?

Understanding Variables

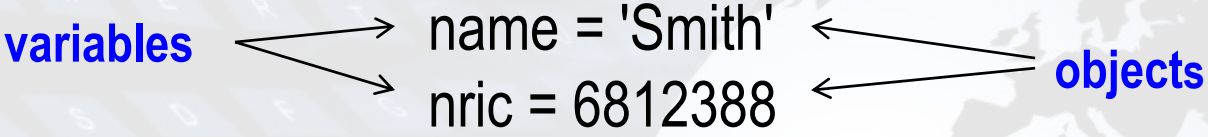
- Programs use variables to store data
- Variables are memory spaces in the computer's Random Access Memory (RAM)
  - ✓ Imagine RAM as computer's jotter book to keep track of information





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

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:

```
BMI = 19.03114
```

Activity 1 – Hip.py

- 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 – TotalCost.py

- Given that the price of an item is $250 and the gst is 7%, calculate and display the total cost of the item.
  - State the input, processing and output needed to solve the problem.
  - Develop pseudocode to calculate and display the total cost of the item.
  - Write a Python program to solve this problem based on the pseudocode developed.

Activity 3 – MarkCalculator.py

- The final mark for PRG1 module is calculated based on 30% of common test, 30% of assignment and 40% of continuous assessment.
  - State the input, processing and output needed to solve the problem.
  - Develop pseudocode to calculate and display the final mark of the module. You may assign continuous assessment 75, assignment 80 and common test 60.
  - Write a Python program to solve this problem based on the pseudocode developed.

Summary

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