ENGG1810/9810

Introduction to Engineering Computing

Lecture 1: Introduction to Programming (Python)

THE UNIVERSITY OF SYDNEY

Dr. Hooman Alavizadeh
Faculty of Engineering

1

2

What will you learn in this course?

Week 1: Introduction to Python	*Programming Basics*
Week 2: Storing Data and Making Decisions	
Week 3: Repeating Actions I	
Week 4: Repeating Actions II	
Week 5: Functions I	*Functions and Packages*
Week 6: Functions II	
Week 7: Libraries and Modules I	
Week 8: Libraries and Modules II	
Week 9: Application I	*Advanced Topics*
Week 10: Application II	
Week 11: Case Study I	
Week 12: Case Study II	
Week 13: Revision and Exam Guide	

3

Today's Lecture

- ***Why Engineers Must Learn Programming?***
- ***Why Python?***
- *Python Basics*
  - *Mathematical Operations*
  - *Python Variables and Types*
- *Built-in Functions – print(), int(), str(), input()*

4

Why Should Engineers LEARN to </CODE>?

Work Faster and More Efficient

You can work 10 times faster and more efficient by writing computer programs to automate tedious tasks (e.g. data cleaning and integration) that you would otherwise need to do by hand.

Discover Creative Solutions

Programming allows you to discover more creative solutions than your colleagues who don't know how to program. It lets you go beyond simply using the tools and data sets that everyone else around you uses; implement far more sophisticated analysis

Why Should Engineers LEARN to </CODE>?

Why Python?

Python is an interpreted high-level general-purpose programming language, designed by Guido Van Rossum

1989 *In December, Van Rossum had been looking for a '**hobby**' **programming project** during Christmas as his office was closed when he decided to **write an interpreter for a "new scripting language"**.*

He named "Python" to "being in an irreverent mood (and a big fan of Monty Python's Flying Circus)"

1999 *Van Rossum submitted a funding proposal to DARPA called **"Computer Programming for Everybody"**, in which he further defined his goals for Python:*

1. *Code that is as **understandable as plain English***
2. *An **easy and intuitive language** just as powerful as major competitors*
3. ***Open source**, so anyone can contribute to its development*
4. ***Suitability for everyday tasks**, allowing for short development times*

Why Learn Python?

1. ***Code that is as understandable as plain English (human-readable)***

   Human: If there is 4 in 1,2,3,4, please print "There is 4"

   Python: `if 4 in [1,2,3,4]: print("There is 4")`

2. ***An easy and intuitive language just as powerful as major competitors***

   Python was built with the goal of getting rid of the complex and keeping only the necessary. It is easier to read, write, and learn than most other major programming languages.

Why Learn Python?

3. ***Open source, so anyone can contribute to its development***

Python is free, open and multiplatform*; not all languages can boast that. Python offers a rich ecosystem of packages held within The* ***Python Package Index (PyPI)****. Users can build modules for the ever-growing PyPi library.*

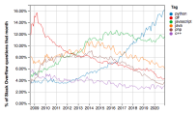
4. ***Suitability for everyday tasks, allowing for short development times***

Python serves as a general-purpose language, which can process virtually any task. Python is used in Data Mining, Data Science, AI, Machine Learning, Web Development, Web Frameworks, Embedded Systems, Graphic Design applications, Gaming, Network development, Product development, Rapid Application Development, Testing, Automation Scripting, etc.

Why Learn Python?

5. ***Python is used in the world's largest and most sophisticated companies***

6. ***Python is the fastest growing programming language***



Today's Lecture

- *Why Engineers Must Learn Programming?*
- *Why Python?*
- ***Python Basics***
  - *Mathematical Operations*
  - *Python Variables and Types*
- *Built-in Functions – print(), int(), str(), input()*

Let's Start with Python Basic!

Python Basics

- Python is an interpreter language, which means it executes the code line by line.
  - can be typed 1) directly in a **shell** or 2) stored in a **.py file** that is read into the shell and evaluated
- Python provides an Python Interpreter: **Shell/REPL**, for executing a single Python command and display the result.

Let's Start with Python Interpreter: Shell/REPL

- **Q:** Why called **Shell**?
  A: The "Shell" covers/wraps a kernel and allows interacting a user's input.
  Kernel: A computer program at the core of a computer's operating system and has complete control over everything in the system
  HW: hardware (CPU, RAM, Disks, Network Ports)

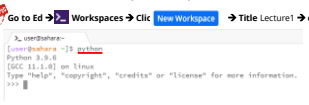
- **Q:** Why REPL (**R**ead, **E**valuate, **P**rint, **L**oop)**?**
  A: It **reads** the command, **evaluates** the command, **prints** the result, and **loops** it back to read the command again.

13

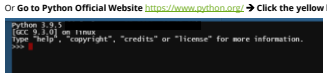
Python Basics

Where can we find the Python Interpreter: Shell/REPL?

Go to Ed → Workspaces → Clic New Workspace **→ Title** Lecture1 **→ cli** **→ Type** python **and Press↵**



Or **Go to Python Official Website** https://www.python.org/ **→ Click the yellow but**



14

Python Basics

Let's try some Basic Math (addition, subtraction, multiplication, division)

1. Addition

```
>>> 1+2
```
and press ↵

2. Subtraction

```
>>> 30 - 4
26
```

3. Multiplication

```
>>> 1212121212 * 3
3636363636
```

4. Division

```
>>> 4000 / 3
1333.3333333333333
>>> 4000 // 3
1333
```

*We are currently using **python 3** but **python 2** would produce the following outcome:*

```
>>> 4000 / 3
1333
```

15

Python Basics: Operators

Arithmetic Operators

Operator	Operation	Example	Result
+	Addition	2+1	3
-	Subtraction	6.3 – 2.1	4.2
*	Multiplication	3*4	12
/	True Division	6/3	2
%	Remainder (modulo)	5/3	2
**	Exponentiation	2**3	8
//	Floor division	20//3	6

16

Python Basics: Programming

Human Language VS Programming Languages

- **Human Language** (English): Words
- **Programming Language**: Numbers, Strings, Simple Operators

English: Can you write your phone number backwards?

Python:

```
1 phone_number = "0498498473"
2 print(phone_number[::-1])
```

How can we manipulate Numbers, Strings, and Simple Operators?

17

Python Basics: Data Types

Objects are Python's abstraction for data. In Python, data are represented by objects or by relations between objects.

Data Types

Data types are the classification or categorization of data items.

Numbers

- **Int**egers – whole numbers `>>> type(11) <class 'int'>`
- **Float** – numbers with decimal points
- **Complex number** – real and imaginary numbers `>>> type(5+2j) <class 'complex'>`

Texts

- **String** - combination of any characters that appear on keyboard e.g. hello world, Nut3lla# `>>> type("ENGG1810") <class 'str'>`

Boolean – True/False options

*Use the `type()` function to get the type or class name of an object.

18

Python Basics: Variables and Data Types

Thus, all values are actually an object of a class depending upon the value.

A literal value is **assigned to a variable** using the **= operator;**

- the left side should be the **name of a variable**,
  - Use meaningful names
  - Should not have space in them
  - Must not use any reserved words (e.g. if, for, True, etc.)
- the right side should be a **value**.
  - Can be numbers, text, Boolean

The following assigns a name to an integer value:

```
my_age = 20
```

Variable name *Value*

Variables in Python are names given to objects, so that it becomes easy to refer a value.
In other words, **a variable points to an object.**

19

Python Basics: Variables and Data Types

A literal value is assigned to a variable using the = operator;

- *the left side should be the name of a variable,*
- *the right side should be a value.*

20

ENGG1810/9810

Python Basics: Variables and Data Types

A literal value is assigned to a variable using the = operator;

- *the left side should be the name of a variable,*
- *the right side should be a value.*


```
ps5 = 1500      bose_sbar = 700
```

21

ENGG1810/9810

Python Basics: Variables and Data Types
(Numbers)


```
ps5 = 1500      bose_sbar = 700
```

Q: Caren wants to buy both PS5 and Bose Soundbar. How much does she need to pay?

```
>>> ps5=1500
>>> bose_sbar=700
>>> ps5+bose_sbar
2200
```

22

ENGG1810/9810

Python Basics: Variables and Data Types
(Numbers)


```
ps5 = 1500      bose_sbar = 700
```

Q: Caren wants to buy only PS5 as she can receive 10% off on PS5. How much does she need to pay?

```
>>> ps5=1500
>>> ps5 * 0.90
1350.0
```

* You can also try, `ps5 * (1 - 0.10)` or others.

You can also update the value of the PS5.

```
>>> ps5=1500
>>> ps5= ps5 * 0.9
>>> ps5
1350.0
```

or

```
>>> ps5=1500
>>> ps5 *= 0.90
>>> ps5
1350.0
```

23

ENGG1810/9810

Python Basics: Variables and Data Types
(Text)


```
ps5 = 1500      bose_sbar = 700

ps5_name = 'PLAYSTATION 5 GAMING CONSOLE'

bose_sbar_name = 'BOSE Soundbar 700'
```

Q: Caren wants to send the text message to her friend (Tom) in order to let her know what products are available in the ENGG1810 e-Shopping Mall.
Note: we need to send the exact product name.

```
>>> ps5_name='PLAYSTATION 5 GAMING CONSOLE'
>>> bose_sbar_name='BOSE Soundbar 700'
>>> ps5_name+bose_sbar_name
'PLAYSTATION 5 GAMING CONSOLEBOSE Soundbar 700'
```

The '+' operator concatenates two strings together.

24

ENGG1810/9810

Python Basics: Variables and Data Types
(Text)

ps5 = 1500 bose_sbar = 700

ps5_name = 'PLAYSTATION 5 GAMING CONSOLE'

bose_sbar_name = 'BOSE Soundbar 700'

Q: Caren wants to send the text message to her friend (Tom) in order to let her know what products are available in the ENGG1810 e-Shopping Mall.
Note: we need to send the exact product name.

```
>>> ps5_name='PLAYSTATION 5 GAMING CONSOLE'
>>> bose_sbar_name='BOSE Soundbar 700'
>>> ps5_name+bose_sbar_name
'PLAYSTATION 5 GAMING CONSOLEBOSE Soundbar 700'
```

```
>>> ps5_name + ' ' + bose_sbar_name
'PLAYSTATION 5 GAMING CONSOLE BOSE Soundbar 700'
```

25

ENGG1810/9810

Python Basics: Variables and Data Types
(Text)

ps5 = 1500 bose_sbar = 700

ps5_name = 'PLAYSTATION 5 GAMING CONSOLE'

bose_sbar_name = 'BOSE Soundbar 700'

Q: Caren wants to send the text message to her friend (Tom) in order to let her know what products are available in the ENGG1810 e-Shopping Mall.
Note: we need to send the exact product name.

```
>>> ps5_name='PLAYSTATION 5 GAMING CONSOLE'
>>> bose_sbar_name='BOSE Soundbar 700'
>>> ps5_name+bose_sbar_name
'PLAYSTATION 5 GAMING CONSOLEBOSE Soundbar 700'
```

```
>>> ps5_name + ' ' + bose_sbar_name
'PLAYSTATION 5 GAMING CONSOLE BOSE Soundbar 700'
```

```
>>> 'Hey, Tom! Check this out! ENGG1810 Shopping Mall has ' + ps5_name + '
 and ' + bose_sbar_name + '!'
'Hey, Tom! Check this out! ENGG1810 Shopping Mall has PLAYSTATION 5 GAMING
 CONSOLE and BOSE Soundbar 700!'
```

26

ENGG1810/9810

Python Basics: Variables and Data Types
(Boolean: True/False Option)

ps5 = 1500 bose_sbar = 700

Q: Tom asked whether PS5 would be cheaper than the Soundbar if we receive 60% off on PS5. Caren guessed that it would be cheaper. Is this correct?

```
>>> ps5=1500
>>> bose_sbar=700
>>> ps5 = 1500 * 0.4
>>>
>>> caren_guess = ps5 < bose_sbar
>>> caren_guess
True
```

27

ENGG1810/9810

Python Basics: Operators

Relational Operators

Operator	Operation	Example	Result
<	Less than	3 < 5	**True**
>	More than	3 > 5	**False**
<=	Less than or equal to	3 <= 5	**True**
>=	Greater than or equal to	5 >= 5	**True**
==	Equal/Same as	5 == 3	**False**
!=	Not equal to	5 != 3	**True**

Important Note!
- Symbol == checks whether the **left-side** value is equal to the **right-side** value
- Symbol = assigns value (**right-side**) to variable (**left-side**)

28

Today's Lecture

- *Why Engineers Must Learn Programming?*
- *Why Python?*
- *Python Basics*
  - *Mathematical Operations*
  - *Python Variables and Types*
- ***Built-in Functions – print(), int(), str(), input(), str.format()***

Try Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available

Wait, what is a function?

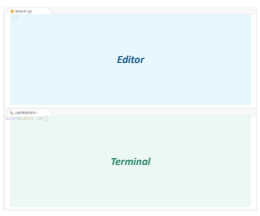
Function is a block of re-usable codes that executes when it is requested.

Built-in Functions				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/3/library/functions.html

Try Built-in Functions with `.py` file

Let's try to run a `.py` file that is read into the shell
In your workspace (opened in the slide 14), click `New File` → Set the file name as `hello.py`
*you can use any name



If you cannot see this terminal, please click the following button:

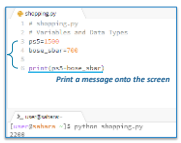
Python Built-in Functions: `print()`

ps5 = 1500 bose_sbar = 700

Same as the slide 22.

Q: Caren wants to buy both PS5 and Bose Soundbar. How much does she need to pay?

```
>>> ps5=1500
>>> bose_sbar=700
>>> ps5+bose_sbar
2200
```



Python Built-in Functions: `print()`

ps5 = 1500 bose_sbar = 700

ps5_name = 'PLAYSTATION 5 GAMING CONSOLE'

bose_sbar_name = 'BOSE Soundbar 700'

Same as the slide 26.

Q: Caren wants to send the text message to her friends (Tom) in order to let her know what products are available in the ENGG1810 e-Shopping Mall.
Note: we need to send the exact product name.

```
>>> ps5_name='PLAYSTATION 5 GAMING CONSOLE'
>>> bose_sbar_name='BOSE Soundbar 700'
>>> 'Hey, Tom! Check this out! ENGG1810 Shopping Mall has ' + ps5_name + '
 and ' + bose_sbar_name + '!'
'Hey, Tom! Check this out! ENGG1810 Shopping Mall has PLAYSTATION 5 GAMING
 CONSOLE and BOSE Soundbar 700!'
```

33

Python Built-in Functions: `int()`

The function int(arg, base)to return an integer.

Use function `int(arg, base)` to return an integer.
- `arg` is the number or string representing an integer literal.
- `arg` default value is 0, base default value is 10.

Default value is returned if no arguments is provided.
If argument is a float, the value is truncated to its whole number.

34

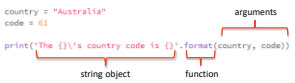
Python Built-in Functions: `str()`

The str() function converts the specified value into a string.

35

Python Built-in Functions: `str.format()`

Another way of displaying a string without casting numeric variables to string (`str()`).



- Curly brackets `{}` are placeholders.
  - First `{}` replaced by value contained in variable `country`
  - Second `{}` replaced by value contained in variable `code`
- Sequence of arguments matters in parenthesis.
- Number of curly brackets must match number of arguments passed into function.

36

Python Built-in Functions: `str.format()`

Another way of displaying a string without casting numeric variables to string (`str()`).

```
country = "Australia"
code = 61

print('The {}\'s country code is {}'.format(country, code))
```

arguments

Escape sequence
- When a backslash (\) appears in a string, it's known as the escape character.
- Backslash and the character immediately following it form an escape sequence.

- Cu
- Se
- Nu

Python Built-in Functions: `str.format()`

We can specify where to put each argument as well.

```
country = "Australia"
code = 61

print('The {1}\'s country code is {0}'.format(code, country))
```

argument 0 argument 1

string object function

- This will return the same output as before but parameter index is stated:
  - `{0}` replaced by value contained in variable `code`
  - `{1}` replaced by value contained in variable `country`
- Parameter index can be used if you want to print the argument list in different order e.g. `{1}{0}` to print value of `country` then `code`.
- Contents outside the curly bracket is printed as it is.
- Parameter index starts with 0.

Python Built-in Functions: `input()`

Python allows for user input. That means we are able to ask the user for input.


```
input.py
1 # input.py
2 # user input
3 name = input("Please Enter Your Name: ")
4 print("HELLO " + name + "! AND WELCOME!")

>_ user@sahara~
[user@sahara ~]$ python input.py
Please Enter Your Name: CAREN
HELLO CAREN! AND WELCOME!
```

Use function `input(prompt)` to get user input from keyboard.
- Argument `prompt` is the text displayed on the terminal
- Value returned by the function is always a string.

THANKS FOR WATCHING

Good luck in studying