

July 2019

ANL 251 Python Programming

L1: Python Basics (Supplementary readings)

Kevin Kam Fung Yuen, PhD

Senior Lecturer

School of Business

Singapore University of Social Sciences

kfyuen@suss.edu.sg,

kevinkf.yuen@gmail.com

Class Schedule

S/N	Course Code	Group	Delivery Style	Date	Day	From	To	Venue
1	ANL251	T06	SEMINAR	24-Jul-2019	Wed	07:00 PM	10:00 PM	HQ BLK B-LAB B.4.15
2	ANL251	T06	SEMINAR	31-Jul-2019	Wed	07:00 PM	10:00 PM	HQ BLK B-LAB B.4.15
3	ANL251	T06	SEMINAR	07-Aug-2019	Wed	07:00 PM	10:00 PM	HQ BLK B-LAB B.4.15
4	ANL251	T06	SEMINAR	14-Aug-2019	Wed	07:00 PM	10:00 PM	HQ BLK B-LAB B.4.15
5	ANL251	T06	SEMINAR	21-Aug-2019	Wed	07:00 PM	10:00 PM	HQ BLK B-LAB B.4.15
6	ANL251	T06	SEMINAR	28-Aug-2019	Wed	07:00 PM	10:00 PM	HQ BLK B-LAB B.4.15

About coding for beginners

- ◆ I hear so I forget
- ◆ I see so I remember
- ◆ I do so I understand

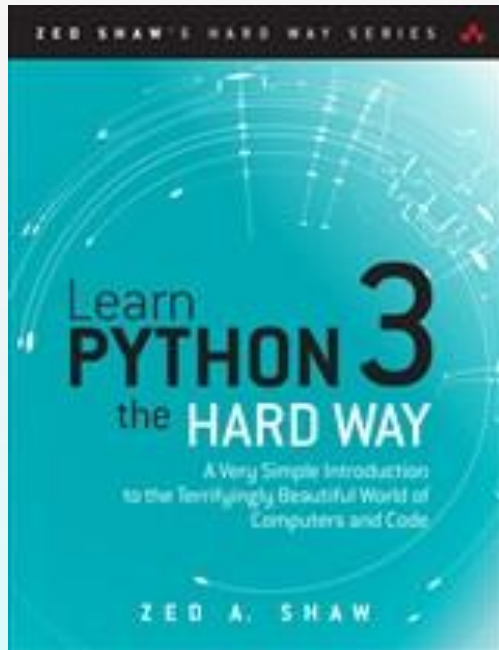
“When we are in the bottom of the mountain, we think the mountain is so high. However, don’t scare.

When we reach the top of mountain, we will find the ground is so close to us!

Next we will challenge the higher mountain. Don’t hesitate! Just go ahead!”

Textbook

- ◆ Don't forget to refer the study guides and slides in Canvas
- ◆ <https://learnpythonthehardway.org/python3/>
- ◆ <http://www.informit.com/promotions/book-registration-learn-python-3-the-hard-way-141409> (videos)



Learn Python 3 **the Hard Way**:
A Very Simple Introduction to the Terrifyingly
Beautiful World of Computers and Code
Zed A. Shaw
print: 9780134692883
eBook: 9780134693651
Addison-Wesley Professional
July 2017

References

- ◆ <http://python.org/> - documentation, tutorials, beginners guide, core distribution, ...
- ◆ <https://docs.python.org/3/contents.html>
- ◆ https://github.com/ehmatthes/pcc/releases/download/v1.0.0/beginners_python_cheat_sheet_pcc_all.pdf
- ◆ Toby Donaldson, Starting out with Python, Third Edition

Ask, Don't Stare

If you write code, you will write bugs. A "bug" means a defect, error, or problem with the code you've written. The legends say that this comes from an actual moth that flew into one of the first computers causing it to malfunction. Fixing it required "de-bugging" the computer. In the world of software, there are a *lot* of bugs. So many.

Like that first moth, your bugs will be hidden somewhere in the code, and you have to go find them. You can't just sit at your computer screen staring at the words you've written hoping that the answer jumps out at you. There is no more additional information you can get doing that, and you need additional information. You need to get up and go find the moth.

To do that you have to interrogate your code and ask it what is going on or look at the problem from a different view. In this book I'll frequently tell you to "stop staring and ask". I'll show you how to make your code tell you everything it can about what's going on and how to turn this into possible solutions. I'll also give you ways to see your code in different ways, so you can get more information and insight.

Do Not Copy-Paste

You must *type* each of these exercises in, manually. If you copy and paste, you might as well not even do them. The point of these exercises is to train your hands, your brain, and your mind in how to read, write, and see code. If you copy-paste, you are cheating yourself out of the effectiveness of the lessons.

Seminar Sessions: 1

Topics to be Covered	Learning Outcomes to be Achieved*	Summary and Discussion of Key Concepts, Theories, Principles	Class Activities to Enhance Learning
Pre-Course Quiz(2%)	SU 1-2		Post announcements on Canvas: <ol style="list-style-type: none"> complete PCQ 1 before the first F2F session. download the textbook videos.
Study Unit 1: Python Basics	<ol style="list-style-type: none"> Execute Python program in the Python interpreter or the PowerShell/Terminal command line Use comments in Python scripts properly Solve problems using Python scripts with appropriate variable names, types and operations Construct formatted printing using format strings, the format() method and escape sequences Create user input and implement appropriate operations based on the input 	<ol style="list-style-type: none"> Python programming environment Variables, types and operators Printing Input 	<p>Provide outline of assessment schedule.</p> <p>Access e-learning material: Study Unit 1 and Textbook Exercises 0 ~ 7, 9 ~12, and recommended online readings.</p> <p>Seminars: discussion and activities to reinforce students' understanding</p>

Attendance

“For attendance, we can simply consider 0.5% per class, so max 3% for attending all 6 classes.

Students who missed a class can submit a **one-page summary** of the study unit missed to make up for the absence.”

GBA grouping

- ◆ Please register your groups in canvas on the first come first served basis.
- ◆ Each group consist of 3 students for each.
- ◆ Only **one group** is allowed for 4 students.

Discussion

- Python Interpreter (SU1 Chapter 1.2, Textbook Video and Exercise 1)
- Execute Python program in the PowerShell/Terminal command line (SU1 Chapter 1.3, , Textbook Video and Exercise 1)
- Any other desktop Python editors or development environments?
- Are **cloud** Python development environment available?

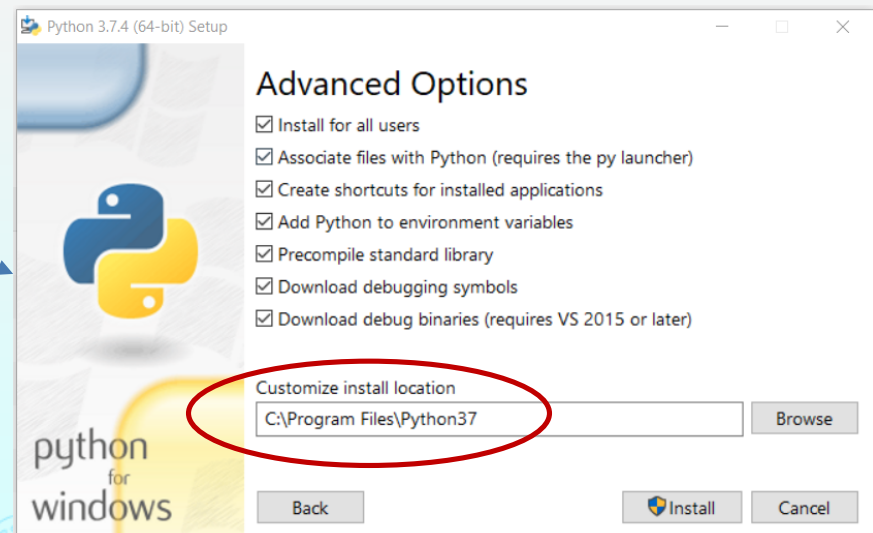
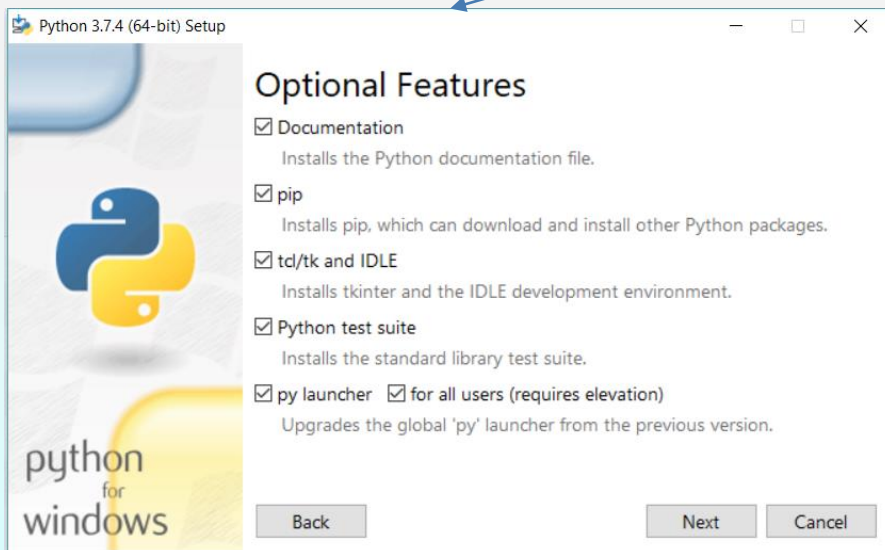
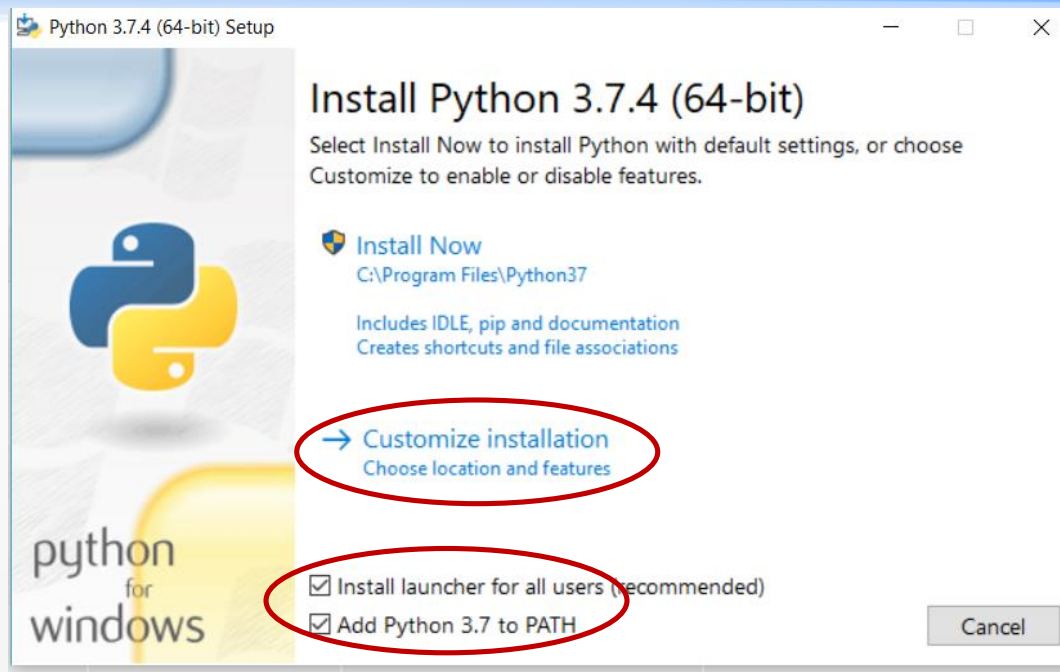
Ex 0: Set up (Window)

◆ Step 1: set up Python

- ◆ <https://www.python.org/downloads/>
- ◆ Current version is 3.7.4
- ◆ Textbook used 3.6
- ◆ <https://www.python.org/downloads/release/python-360/>

◆ Step 2: set up editor

- ◆ <https://atom.io>



```
Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Kevin Yuen> python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 6
4 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit
Use quit() or Ctrl-Z plus Return to exit
>>> quit()
PS C:\Users\Kevin Yuen> █
```


Python Interactive Shell

You can type things directly into a running Python session

```
>>> name="Kevin Yuen"
```

```
>>> name
```

```
'Kevin Yuen'
```

```
>>> print("Hello", name)
```

```
Hello Kevin Yuen
```

```
>>> level= 4*5+7
```

```
>>> level
```

```
27
```

Editor & IDE

Editor Name	Works On	Where To Get It
Visual Studio Code	Windows, macOS, Linux	https://code.visualstudio.com/
Notepad++	Windows	https://notepad-plus-plus.org/
gEdit	Linux, macOS, Windows	https://github.com/GNOME/gedit
Textmate	macOS	https://github.com/textmate/textmate
SciTE	Windows, Linux	http://www.scintilla.org/SciTE.html
jEdit	Linux, macOS, Windows	http://www.jedit.org/

The IDE Programming Environment

- ◆ IDE (Integrated Development Program):
single program that provides tools to write,
execute and test a program
 - ◆ Automatically installed when Python language
is installed
 - ◆ Runs in interactive mode
 - ◆ Has built-in text editor with features designed
to help write Python programs

IDE



<https://atom.io/>



<https://www.anaconda.com/>



Get **Eclipse IDE 2019-06**

Install your favorite desktop IDE packages.

[Download 64 bit](#)

[Download Packages](#) | [Need Help?](#)

<https://www.eclipse.org/downloads/>



SPYDER

The Scientific Python Development Environment

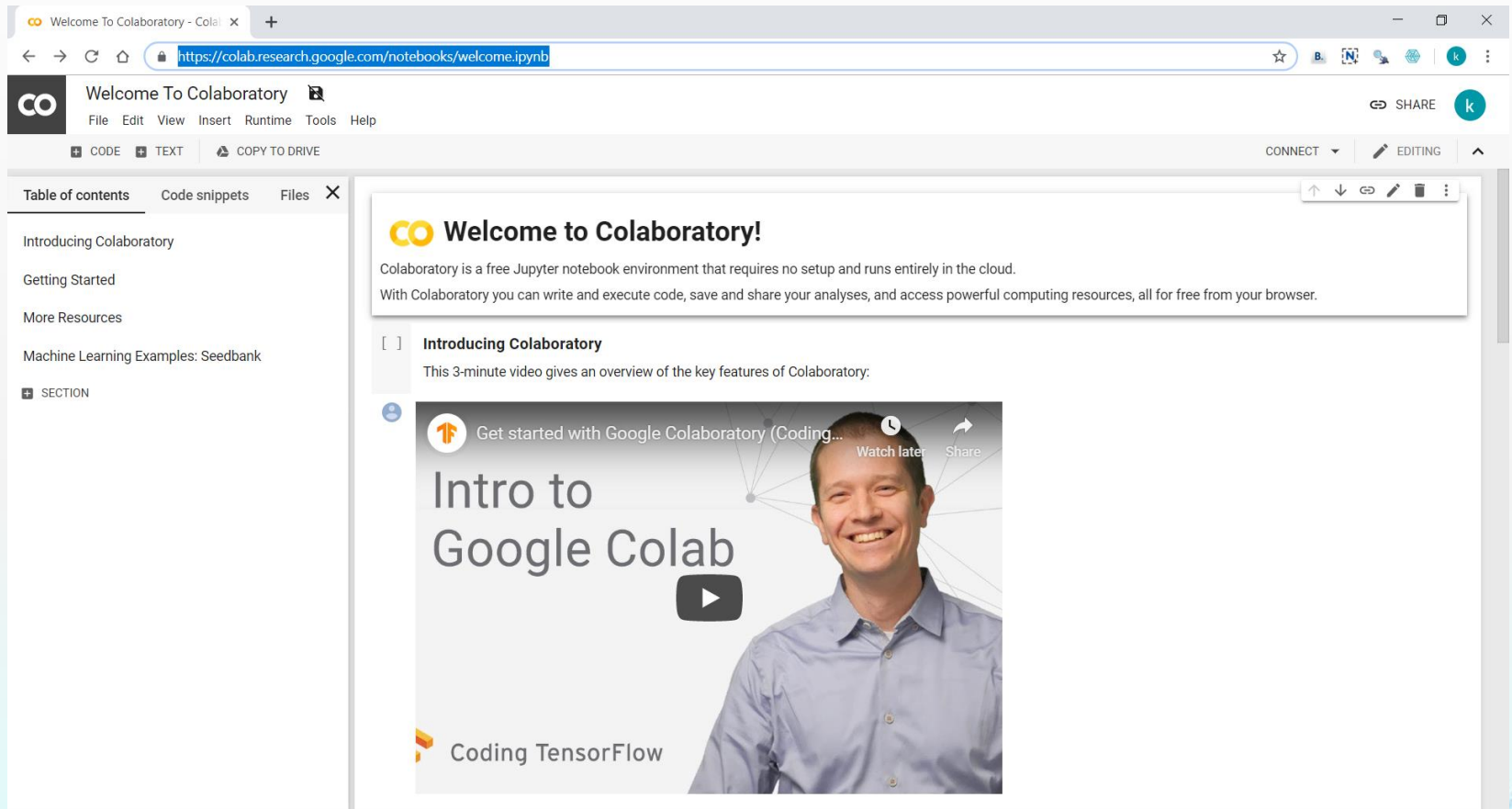
<https://www.spyder-ide.org/>



PyCharm

<https://www.jetbrains.com/pycharm/>

Cloud



The screenshot shows a web browser window with the URL <https://colab.research.google.com/notebooks/welcome.ipynb>. The page is titled "Welcome To Colaboratory" and features a navigation menu on the left with options like "Table of contents", "Code snippets", and "Files". The main content area displays a "Welcome to Colaboratory!" message, stating that it is a free Jupyter notebook environment. Below this, there is a section titled "Introducing Colaboratory" with a video player showing a man smiling and the text "Intro to Google Colab" and "Coding TensorFlow".

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

CODE TEXT COPY TO DRIVE

CONNECT EDITING

Table of contents Code snippets Files

Introducing Colaboratory

Getting Started

More Resources

Machine Learning Examples: Seedbank

SECTION

Welcome to Colaboratory!

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

Introducing Colaboratory

This 3-minute video gives an overview of the key features of Colaboratory:

Get started with Google Colaboratory (Coding...)

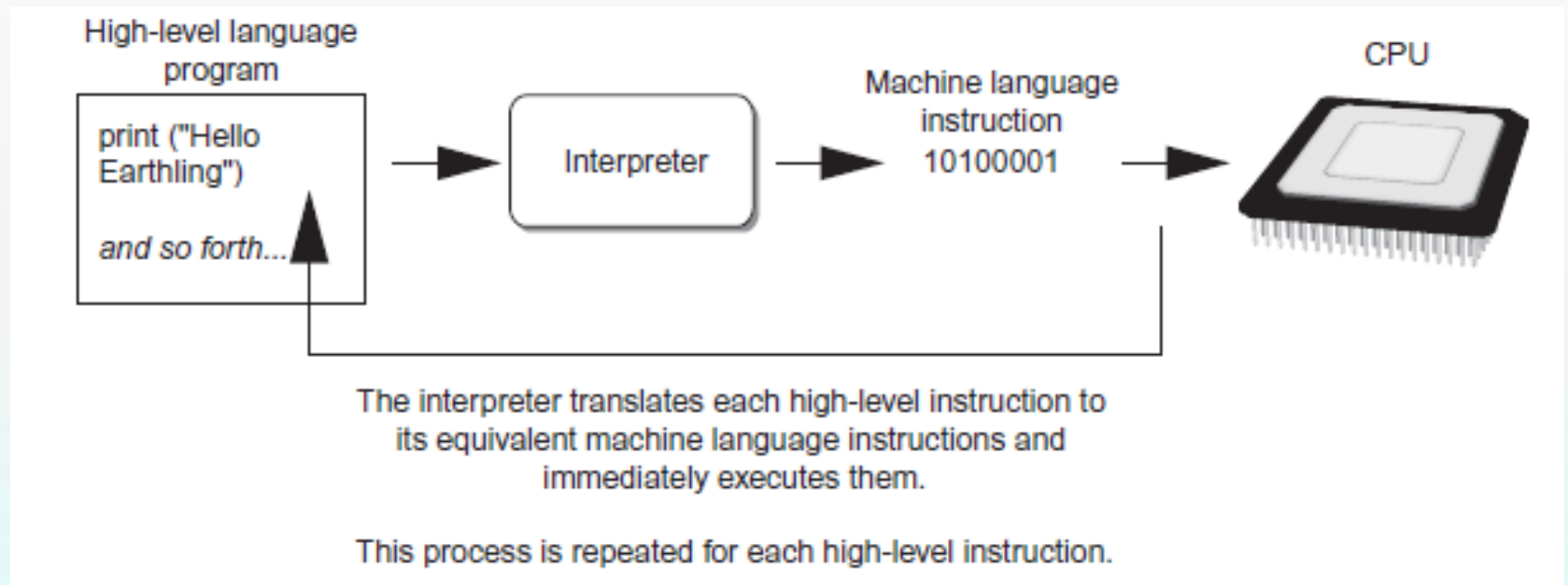
Watch later Share

Intro to Google Colab

Coding TensorFlow

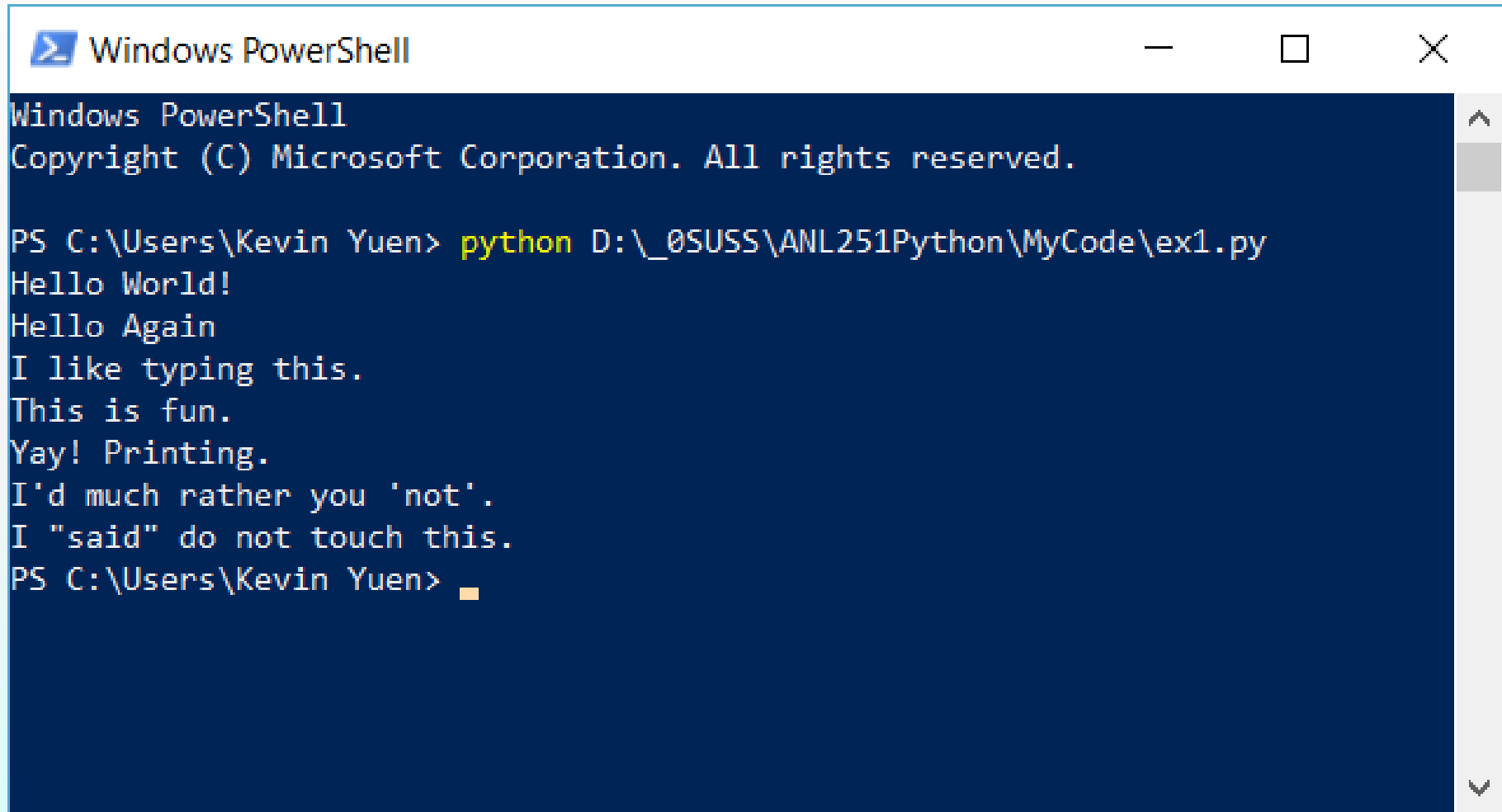
<https://colab.research.google.com>

Compilers and Interpreters



Executing a high-level program with an interpreter

Exercise 1: A Good First Program



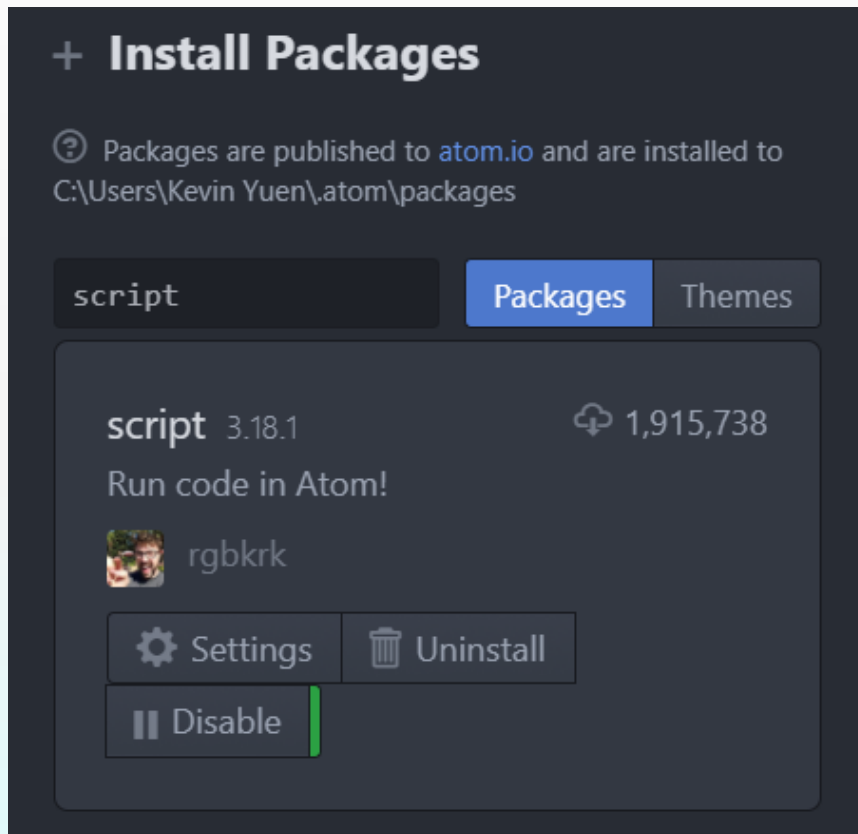
A screenshot of a Windows PowerShell terminal window. The title bar at the top reads "Windows PowerShell" and includes standard window controls (minimize, maximize, close). The terminal content shows the following sequence of text: "Windows PowerShell", "Copyright (C) Microsoft Corporation. All rights reserved.", a prompt "PS C:\Users\Kevin Yuen>" followed by the command "python D:_0SUSS\ANL251Python\MyCode\ex1.py", and the program's output: "Hello World!", "Hello Again", "I like typing this.", "This is fun.", "Yay! Printing.", "I'd much rather you 'not'.", and "I 'said' do not touch this.". The prompt "PS C:\Users\Kevin Yuen>" appears again at the end, followed by a cursor. A vertical scrollbar is visible on the right side of the terminal window.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Kevin Yuen> python D:\_0SUSS\ANL251Python\MyCode\ex1.py
Hello World!
Hello Again
I like typing this.
This is fun.
Yay! Printing.
I'd much rather you 'not'.
I "said" do not touch this.
PS C:\Users\Kevin Yuen> █
```

Packages:

script, autocomplete-python, kite



Packages → setting view → install packages\theme

Using Python

- ◆ Python must be installed and configured prior to use
 - ◆ One of the items installed is the Python interpreter
- ◆ Python interpreter can be used in two modes:
 - ◆ Interactive mode: enter statements on keyboard
 - ◆ Script mode: save statements in Python script

Writing Python Programs and Running Them in Script Mode

- ◆ Statements entered in interactive mode are not saved as a program
- ◆ To have a program use script mode
 - ◆ Save a set of Python statements in a file
 - ◆ The filename should have the .py extension
 - ◆ To run the file, or script, type
`python filename`
at the operating system command line

OPERATORS, VARIABLES AND TYPES

Maths Operations

- `+` plus
- `-` minus
- `/` slash
- `*` asterisk
- `%` percent
- `<` less-than
- `>` greater-than
- `<=` less-than-equal
- `>=` greater-than-equal

More in

<https://docs.python.org/3/reference/expressions.html#operator-precedence>

```
[>>> 2 + 2
4
[>>> 50 - 5*6
20
[>>> (50 - 5*6) / 4
5.0
[>>> 8 / 5 # division always returns a floating point number
1.6
[>>> 17 // 3 # floor division discards the fractional part
5
[>>> 17 % 3 # the modulus % operator returns the remainder of the division
2
[>>> 2 ** 7 # 2 to the power of 7
128
[>>> "Happy" + "New Year"
'HappyNew Year'
```

Figure 1.5 Math operations in Python

Discussion

Observe the code in Figure 1.5. Discuss:

- What operation does the symbol `**` represent?
- What operation does the `/` represent?
- What operation does the `//` represent?
- When applied to two int operands, which operation always evaluates to type float?
- In what orders are the operations evaluated?
- What are the three basic Python data types used?

Concept of Operator and operands

- ◆ $2+4=6$
- ◆ $+$ is the notation for **operator**, called addition
- ◆ 2 and 4 are **operands**
- ◆ 6 is the **result**

```
# clean the screen by code  
import os  
os.system('cls')
```

```
4
>>> 13/6    # division
2.1666666666666665
>>> 13//6   # floor division
2
>>> 13%6    # remainder . it is not percentage.
1
>>> 4**3    # 4 to power of 3; Exponentiation
```

```
>>> type(2)
<class 'int'>
>>> type(2.0)
<class 'float'>
>>> type(3.1)
<class 'float'>
>>> type(1/5)
<class 'float'>
>>> type(1*5)
<class 'int'>
>>> type(1*0.5)
<class 'float'>
>>> type(2/2)
<class 'float'>
>>> 2/2
1.0
```

Quiz

Expression	Result
------------	--------

11 + 56	
---------	--

23 - 52	
---------	--

4 * 5	
-------	--

2 ** 5	
--------	--

9 / 2	
-------	--

9 // 2	
--------	--

9 % 2	
-------	--

'ab' + 'c'	
------------	--

'a' * 5	
---------	--

4 * 'bc'	
----------	--

'hello' + 5	
-------------	--

'good' * 'day'	
----------------	--

'a' - 'b'	
-----------	--

'a' / 'b'	
-----------	--

```
>>> 11 + 56
67
>>> 23 - 52
-29
>>> 4 * 5
20
>>> 2 ** 5
32
>>> 9 / 2
4.5
>>> 9 // 2
4
>>> 9 % 2
1
>>> 'ab' + 'c'
'abc'
>>> 'a' + 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> 4 * 'bc'
'bcbcbcbc'
>>> 'hello' + 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> 'good' * 'day'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
>>> 'a' - 'b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and 'str'
>>> 'a' / 'b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /: 'str' and 'str'
>>>
```

Quiz

What is printed by the code below?

```
print("work" + "hard" * 2 + "and" + "happily")
```

```
>>> print("work" + "hard"*2 + "and" + "happily")  
workhardhardandhappily
```

Quiz

Select the expression(s) that result in a **SyntaxError**.

☐ `8 / (3 / (2 / 3)))`

☐ `6 + -2`

☐ `4 **`

☐ `5 * (3 + 2)`

```
>>> 8/(3/(2/3)))
      File "<stdin>", line 1
        8/(3/(2/3)))
              ^
SyntaxError: invalid syntax
>>> 8/(3/(2/3))
1.7777777777777777
>>> 6 + -2
4
>>> 6 +- 2
4
>>> 6 + - 2
4
>>> 4**
      File "<stdin>", line 1
        4**
          ^
SyntaxError: invalid syntax
>>> 5 * (3+2)
25
>>>
```


Are the following legal Python names?

1_score
score1
score_1
hours@n
cube's
cubes

Are the following Python names identical?

Seconds
seconds

```
>>> 1_a = 3
      File "<stdin>", line 1
        1_a = 3
          ^
SyntaxError: invalid token
>>> a1 = 3
>>> a_1 = 3
>>> a@m = 4
      File "<stdin>", line 1
SyntaxError: can't assign to operator
>>> a's = 5
      File "<stdin>", line 1
        a's = 5
          ^
SyntaxError: EOL while scanning string literal
>>> abcs = 6
>>> seconds = 4
>>> print(Seconds)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Seconds' is not defined
>>> print(seconds)
4
```

FORMATTED PRINTING

Ex 5-10

try

```
>>> a=3
>>> b=f"there are {a} kids."
>>> print(b)
there are 3 kids.
>>>
```

escape sequences

Select the statement(s) that will result in SyntaxError.

```
greeting = "I'm feeling a bit lucky"
greeting = 'I'm feeling a bit lucky'
greeting = "I\'m feeling a bit lucky"
greeting = 'I\'m feeling a bit lucky'
```

```
>>> greeting1 = "I'm feeling a bit lucky"
>>> greeting2 = 'I'm feeling a bit lucky'
File "<stdin>", line 1
    greeting2 = 'I'm feeling a bit lucky'
                ^
SyntaxError: invalid syntax
>>> greeting3 = "I\'m feeling a bit lucky"
>>> greeting4 = 'I\'m feeling a bit lucky'
>>> greeting1
"I'm feeling a bit lucky"
>>> greeting2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'greeting2' is not defined
>>> greeting3
"I'm feeling a bit lucky"
>>> greeting4
"I'm feeling a bit lucky"
```

Quiz

Q4

What will be printed for each of the following?

```
print('How\nare\nyou?')
```

```
print('3\t4\t5')
```

```
print('\\')
```

```
print('\')
```

```
print('don\t')
```

```
print('He said, \'hi\'.')
```

```
print("It's fun!", "Don't you think?")
```

```
print("\n is the newline character")
```

```
print("this \n is the newline character in Python")
```

```
>>> #l1q4
... print('How\nare\nyou?')
How
are
you?
>>> print('3\t4\t5')
3      4      5
>>> print('\\')
\
>>> print('\')
File "<stdin>", line 1
    print('\')
          ^
SyntaxError: EOL while scanning string literal
>>> print('don\t')
don't
>>> print('He said, \'hi\'.')
He said, 'hi'.
>>> print("It's fun!", "Don't you think?")
It's fun! Don't you think?
>>> print("\n is the newline character")
\n is the newline character
>>> print("this \n is the newline character in Python")
this
is the newline character in Python
```


USER INPUT

Ex 11-12


```
name = input("Name? ")
age = input("How old are you? ")
height = input("How tall are you in metres? ")
weight = input("How much do you weigh in kilograms? ")

print(f"{name} is {age} old, {height} tall and {weight} heavy.")

bmi = float(weight)/float(height)**2
print(f"Your BMI is {bmi}.")
```

Figure 1.9 User input in Python

Discussion

Converting between int, float and str

After the second line of code in Figure 1.9 has been executed and the user types 21 followed by Enter/Return, what type of value does variable age refer to?

```
>>> print("How old are you?", end = ' ')
How old are you? >>> age = input()
14
>>> age
'14'
>>> type(age)
<class 'str'>
>>> agef = float(age)
>>> agef
14.0
>>> type(agef)
<class 'float'>
>>> ageInt = int(age)
>>> ageInt
14
>>> type(ageInt)
<class 'int'>
>>> ageInputFloat = float(input())
89
>>> ageInputFloat
89.0
>>> type(ageInputFloat)
<class 'float'>
```

Quiz

What is printed after executing each of the code below?

```
print('3' * 5)
```

```
print (float(str(45)))
```

```
print(str(int('99'))=='99')
```

```
print(int('-99.9'))
```

```
print(float('-9.9.9'))
```

```
print(int('7 eggs'))
```

```
print(str(int('4')+int('2'))+'eggs')
```

```
>>> print('3' * 5)
33333
>>> print (float(str(45)))
45.0
>>> print(str(int('99'))=='99')
True
>>> print(int('-99.9'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '-99.9'
>>> print(float('-9.9.9'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: '-9.9.9'
>>> print(int('7 eggs'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '7 eggs'
>>> print(str(int('4')+int('2'))+'eggs')
6eggs
```