

Fall, 2020

Python Programming Workshop

Workshop 1

Python Programming Environment and Basics

Kevin Kam Fung Yuen
Department of Computing
Hong Kong Polytechnic University
kevin.yuen@polyu.edu.hk

Outline

- General information
- Python Programming Environment and Basics

<ol style="list-style-type: none">1. Execute Python program in the Python interpreter or the PowerShell/Terminal command line2. Use comments in Python scripts properly3. Solve problems using Python scripts with appropriate variable names, types and operations4. Construct formatted printing using format strings, the format() method and escape sequences5. Create user input and implement appropriate operations based on the input	<ol style="list-style-type: none">1: Python programming environment2: Variables, types and operators3: Printing4: Input
---	--

Lecture notes

- Lecture notes are available in
- <https://github.com/kkfyuen/PythonWorkshop2020>
- The previous version of the slides can be found in
- <https://github.com/kkfyuen/ANL251Python>

Class Schedule (Tentative)

19 Sep: Python Programming Environment and Basics

26 Sep: Control Flow and Lists

3 Oct: Tuples and Dictionaries

10 Oct: Functions, Modules, OO and Packages

17 Oct: Scientific Computing and Plotting

About coding for beginners

不聞不若聞之，
聞之不若見之，
見之不若知之，
知之不若行之。
學至於行之而止矣。
行之，明也。

-- 《荀子·儒效》

I hear and I forget.
I see and I remember.
I **do** and 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!”

Why Python

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python	☉ ☒ ☒	100.0
2	Java	☉ ☐ ☒	95.3
3	C	☐ ☒ ☒	94.6
4	C++	☐ ☒ ☒	87.0
5	JavaScript	☉	79.5
6	R	☐ ☒	78.6
7	Arduino	☒	73.2
8	Go	☉ ☐ ☒	73.1
9	Swift	☐ ☒	70.5
10	Matlab	☐	68.4

2020

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python	☉ ☐ ☒	100.0
2	Java	☉ ☐ ☒	96.3
3	C	☐ ☒ ☒	94.4
4	C++	☐ ☒ ☒	87.5
5	R	☐ ☒	81.5
6	JavaScript	☉	79.4
7	C#	☉ ☐ ☒ ☒	74.5
8	Matlab	☐	70.6
9	Swift	☐ ☒	69.1
10	Go	☉ ☐ ☒	68.0

2019

Language Rank Types Spectrum Ranking

1.	Python	☉ ☐ ☒	100.0
2.	C++	☐ ☒ ☒	99.7
3.	Java	☉ ☐ ☒	97.5
4.	C	☐ ☒ ☒	96.7
5.	C#	☉ ☐ ☒	89.4
6.	PHP	☉	84.9
7.	R	☐ ☒	82.9
8.	JavaScript	☉ ☐	82.6
9.	Go	☉ ☐ ☒	76.4
10.	Assembly	☐	74.1

2018

Language Rank Types Spectrum Ranking

1.	Python	☉ ☐ ☒	100.0
2.	C	☐ ☒ ☒	99.7
3.	Java	☉ ☐ ☒	99.5
4.	C++	☐ ☒ ☒	97.1
5.	C#	☉ ☐ ☒	87.7
6.	R	☐ ☒	87.7
7.	JavaScript	☉ ☐	85.6
8.	PHP	☉	81.2
9.	Go	☉ ☐ ☒	75.1
10.	Swift	☐ ☒	73.7

2017

Language Rank Types Spectrum Ranking

1.	C	☐ ☒ ☒	100.0
2.	Java	☉ ☐ ☒	98.1
3.	Python	☉ ☐ ☒	98.0
4.	C++	☐ ☒ ☒	95.9
5.	R	☐ ☒	87.9
6.	C#	☉ ☐ ☒	86.7
7.	PHP	☉	82.8
8.	JavaScript	☉ ☐	82.2
9.	Ruby	☉ ☐ ☒	74.5
10.	Go	☉ ☐ ☒	71.9

2016

K.K.F. Yuen

Source: spectrum.ieee.org/static/interactive-the-top-programming-languages-2020

Recommended Textbook

- Toby Donaldson, Starting out with Python, 5th Edition, 2021
- Zed A. Shaw, Learn Python 3 **the Hard Way**: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code, July 2017
 - <https://learnpythonthehardway.org/python3/>
 - <http://www.informit.com/promotions/book-registration-learn-python-3-the-hard-way-141409> (videos)

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

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.

- No source codes cleaned will be provided for these workshops.
- You must type and code by yourself.
- The workshops are not to aim to provide solutions, but expect learners to learn from mistakes.

There are vast of source codes in the internet. You may try them in the future.

Set up Python programming Environment (Window)

1. **Install Python**
2. **Install IDE**
3. Install Jupyter Notebook (in Workshop 3/4).
4. **Use jupyter notebook on the cloud services**

Install Python

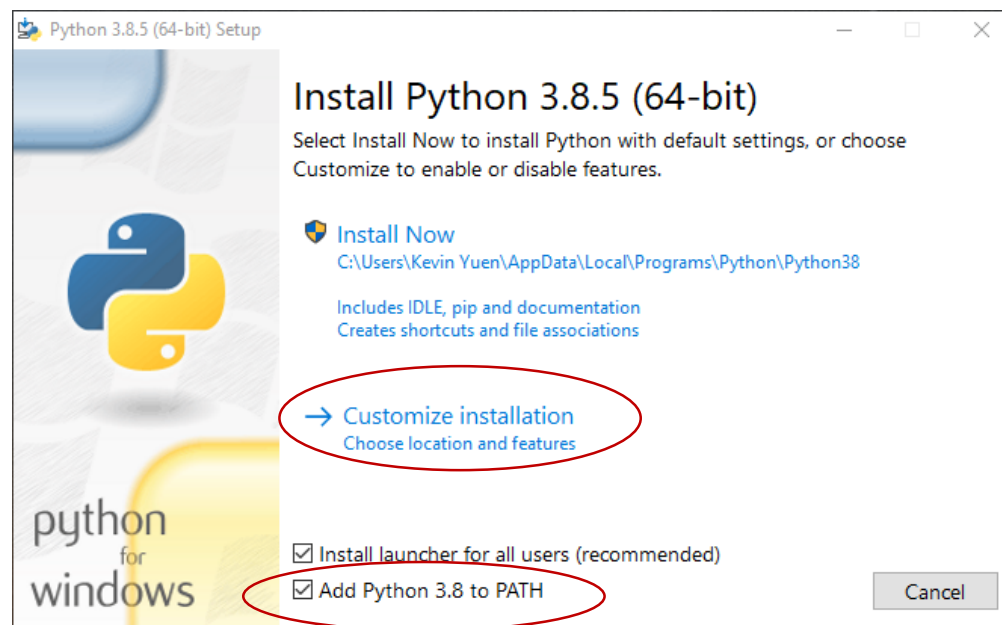
<https://www.python.org/downloads/>

Current version is 3.8.5

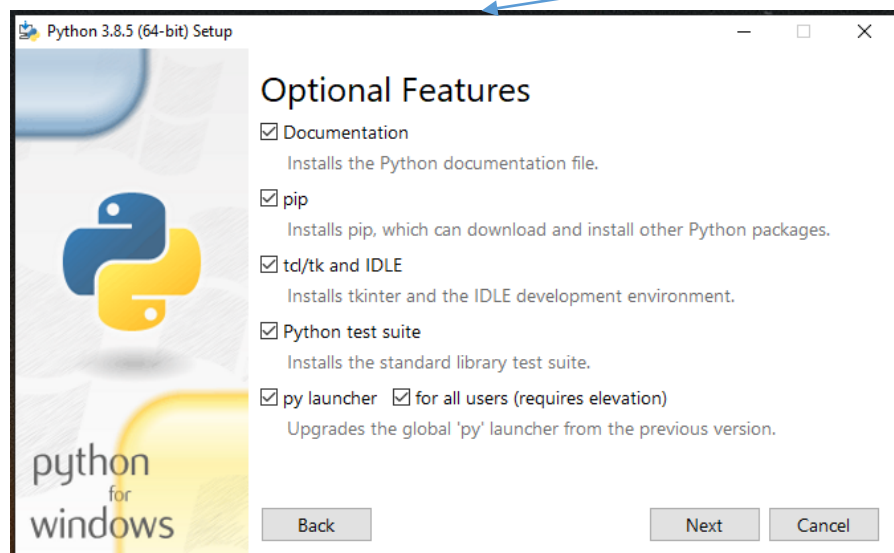
<https://www.python.org/downloads/release/python-385/>

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e2f52bcf531c8cc94732c0b6ff933ff0	24149103	SIG
XZ compressed source tarball	Source release		35b5a3d0254c1c59be9736373d429db7	18019640	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	2f8a736eeb307a27f1998cfd07f22440	30238024	SIG
Windows help file	Windows		3079d9cf19ac09d7b3e5eb3fb05581c4	8528031	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	73bd7aab047b81f83e473efb5d5652a0	8168581	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	0ba2e9ca29b719da6e0b81f7f33f08f6	27864320	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	eeab52a08398a009c90189248ff43dac	1364128	SIG
Windows x86 embeddable zip file	Windows		bc354669bffd81a4ca14f06817222e50	7305731	SIG
Windows x86 executable installer	Windows		959873b37b74c1508428596b7f9df151	26777232	SIG
Windows x86 web-based installer	Windows		c813e6671f334a269e669d913b1f9b0d	1328184	SIG

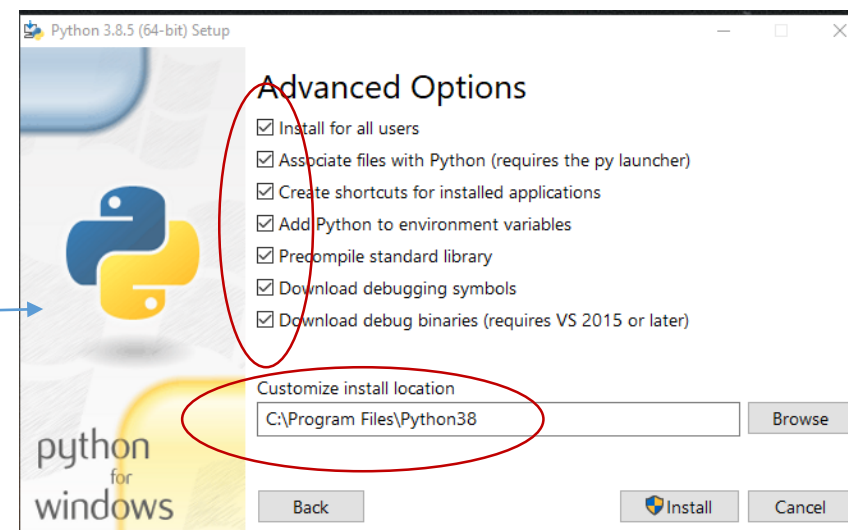
Install Python



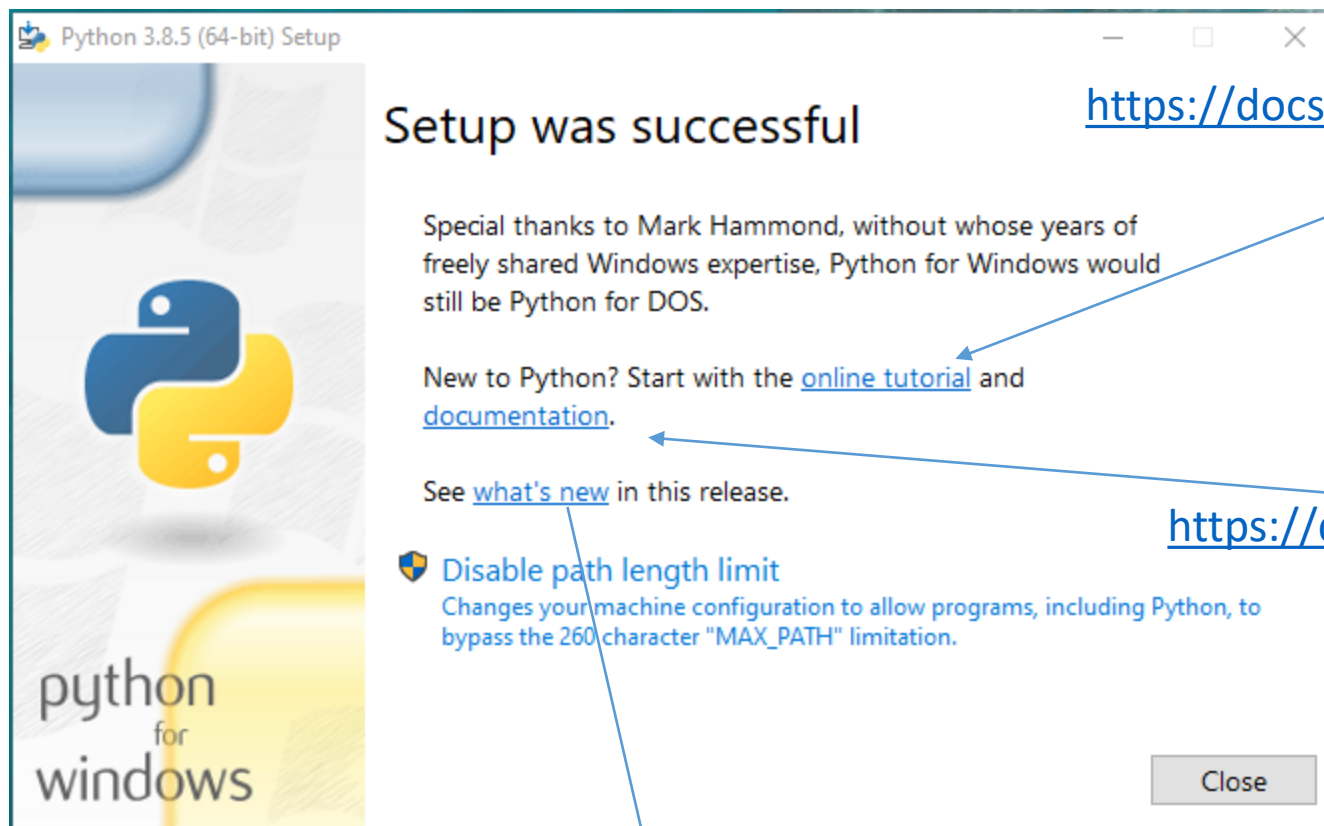
Check all



Check all



Congratulations!



<https://docs.python.org/3.8/tutorial/index.html>

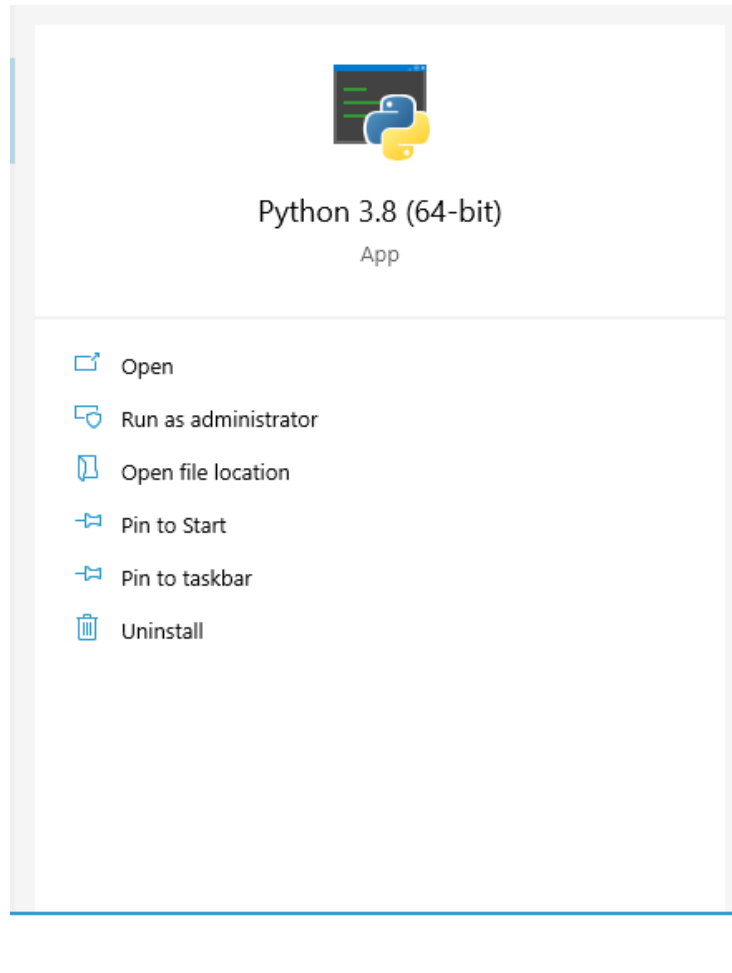
<https://docs.python.org/3.8/index.html>

<https://docs.python.org/3.8/whatsnew/3.8.html>

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

Python Interactive Shell



A screenshot of the Python 3.8 (64-bit) interactive shell window. The window title bar shows 'Python 3.8 (64-bit)'. The shell displays the following text:

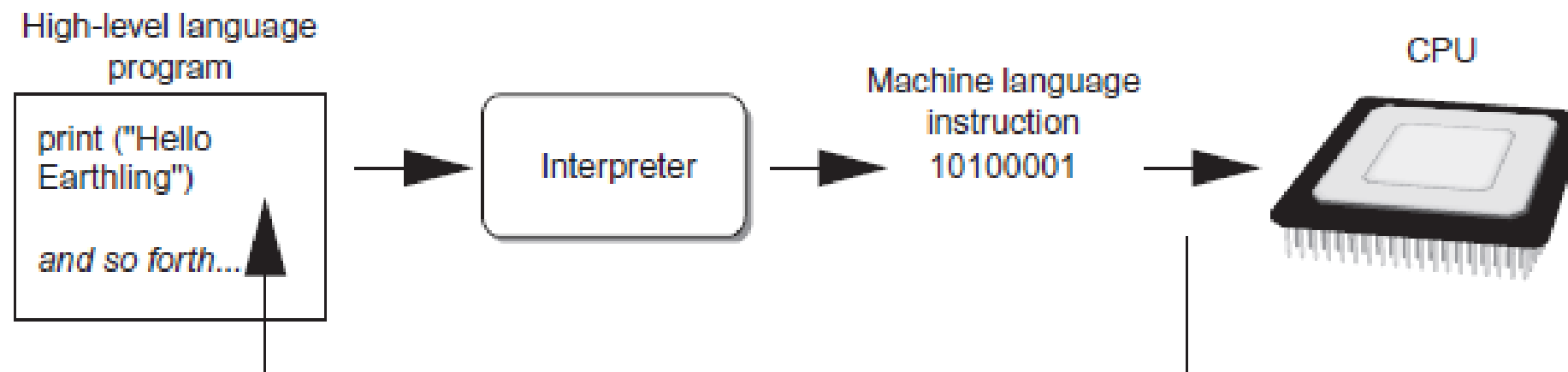
```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64  
bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> name="Kevin Yuen"  
>>> name  
'Kevin Yuen'  
>>> print("Hello", name)  
Hello Kevin Yuen  
>>> level= 4*5+7  
>>> level  
27  
>>> quit()
```

Use quit() to leave

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

Compilers and Interpreters



The interpreter translates each high-level instruction to its equivalent machine language instructions and immediately executes them.

This process is repeated for each high-level instruction.

Executing a high-level program with an interpreter

Powershell

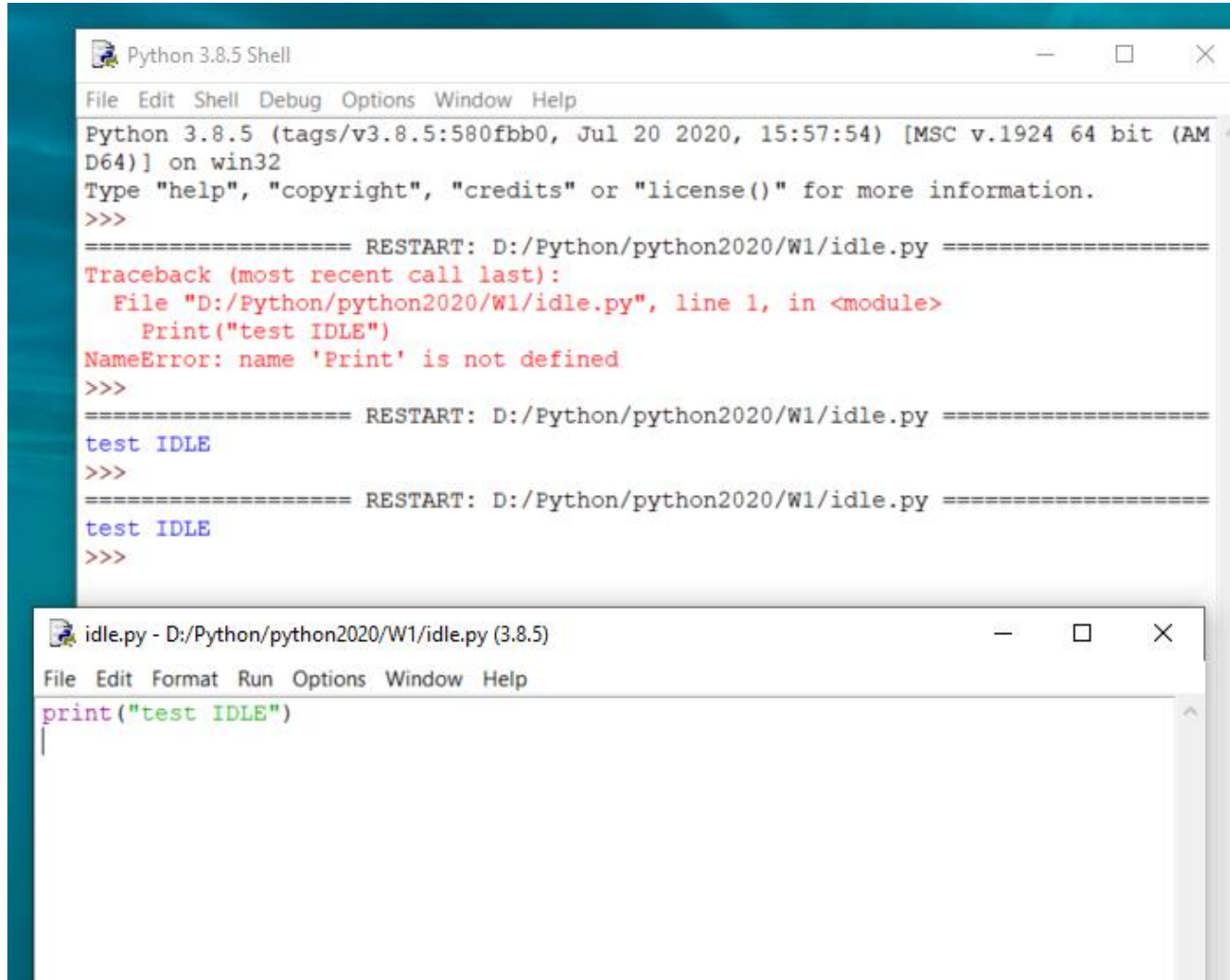
1. Create a py file, e.g. ex1.py
2. Type python code
3. Open window PowerShell
4. Run the code

<https://learnpythonthehardway.org/python3/ex1.html>

```
Administrator: Windows PowerShell

PS D:\Python\python2020\W1> python 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 D:\Python\python2020\W1>
```

IDLE (Integrated Development and Learning Environment)



The screenshot displays two windows from the Python 3.8.5 environment. The top window, titled 'Python 3.8.5 Shell', shows the command prompt interface. It includes a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a status bar indicating 'Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32'. The shell shows a series of commands and outputs: a help message, a restart command, a traceback for a 'NameError: name 'Print' is not defined' (caused by using 'Print' instead of 'print'), and two successful executions of 'test IDLE' after using the correct 'print' function. The bottom window, titled 'idle.py - D:/Python/python2020/W1/idle.py (3.8.5)', shows the IDLE code editor with a menu bar (File, Edit, Format, Run, Options, Window, Help) and a single line of code: `print("test IDLE")`.

```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Python/python2020/W1/idle.py =====
Traceback (most recent call last):
  File "D:/Python/python2020/W1/idle.py", line 1, in <module>
    Print("test IDLE")
NameError: name 'Print' is not defined
>>>
===== RESTART: D:/Python/python2020/W1/idle.py =====
test IDLE
>>>
===== RESTART: D:/Python/python2020/W1/idle.py =====
test IDLE
>>>

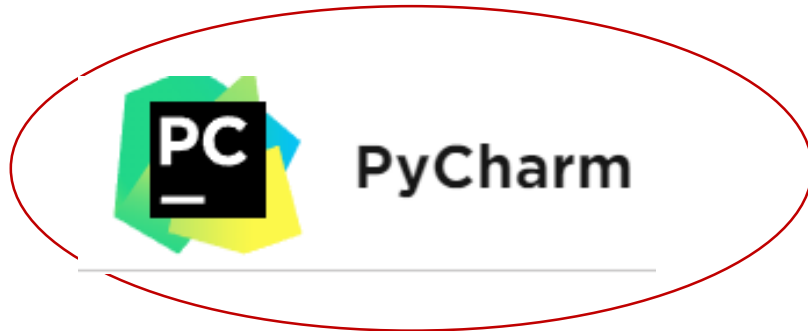
idle.py - D:/Python/python2020/W1/idle.py (3.8.5)
File Edit Format Run Options Window Help
print("test IDLE")
|
```


Editor

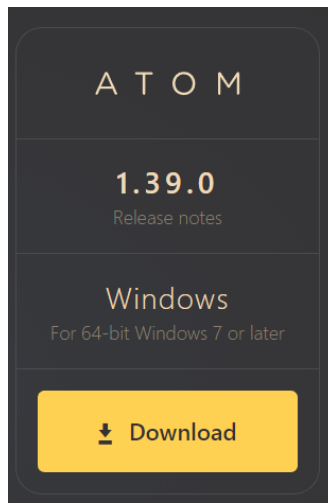
Editor Name	Works On	Where To Get It
Visual Studio Code	Windows, macOS, Linux	https://code.visualstudio.com/
Notepad++	Windows	https://notepad-plus-plus.org/
gEdit	Linux, macOS, Windows	https://github.com/GNOME/gedit
Textmate	macOS	https://github.com/textmate/textmate
SciTE	Windows, Linux	http://www.scintilla.org/SciTE.html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



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



<https://atom.io/>



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



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



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

PyCharm

Windows

Mac

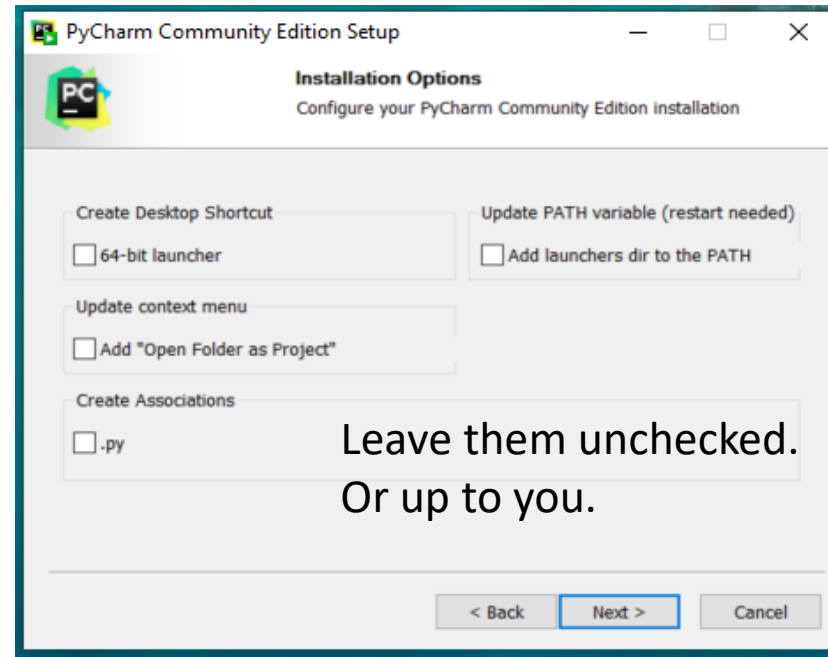
Linux

Community

For pure Python development

Download

Free, open-source

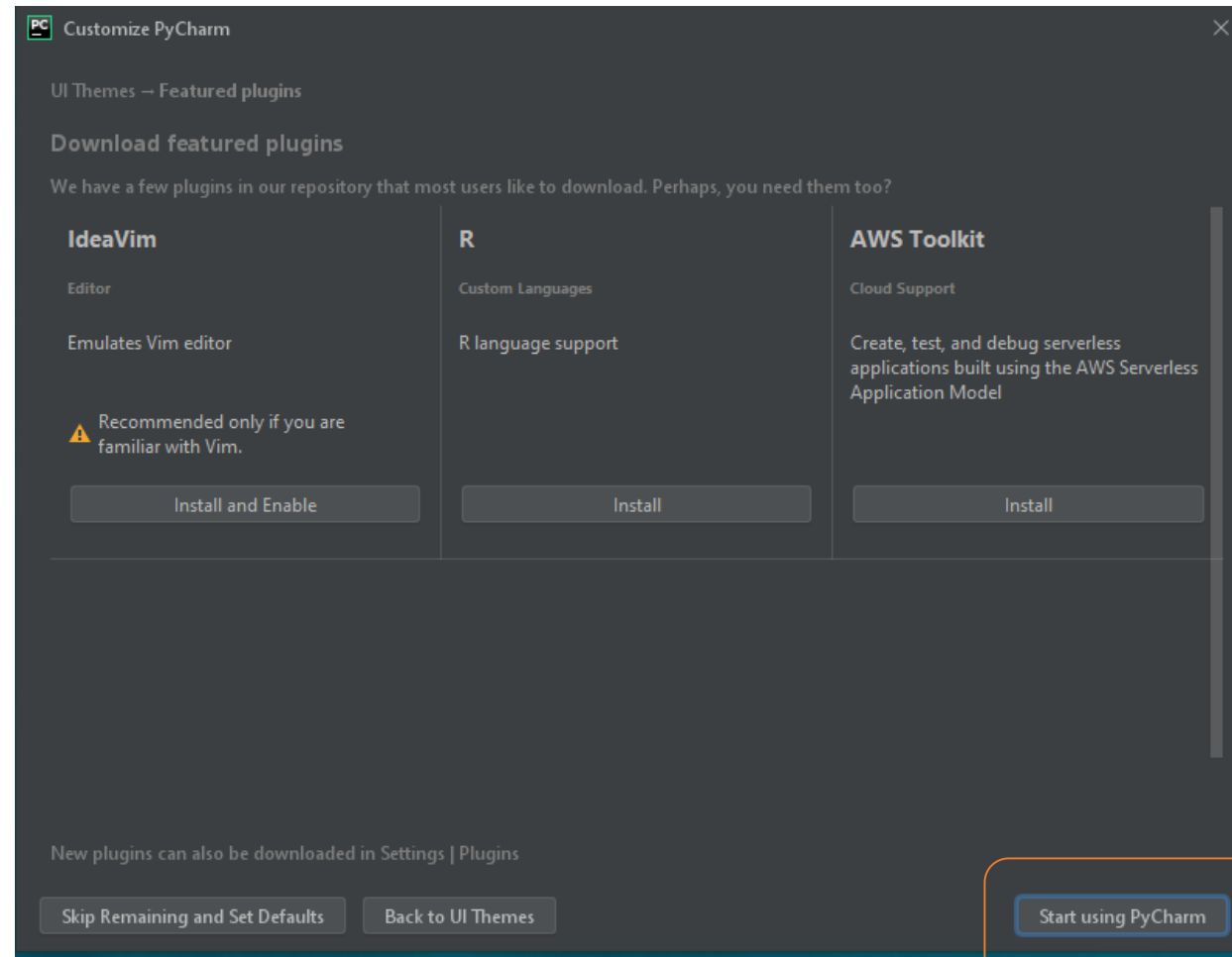
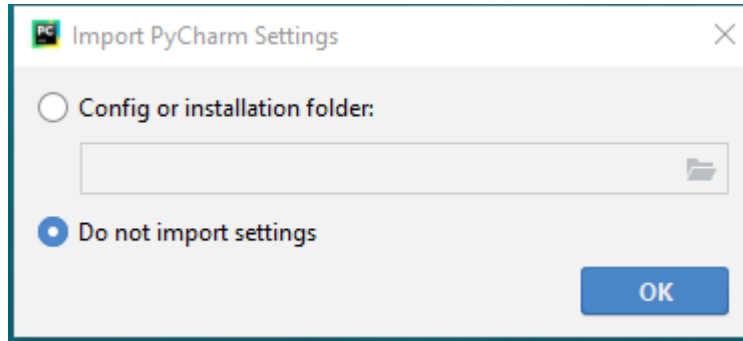


Leave them unchecked.
Or up to you.

<https://www.jetbrains.com/pycharm/download/#section=windows>

Click the pycharm-community-2020.2.2.exe to install

Run PyCharm



Useful editor configurations

You can use the **Settings/Preferences** dialog `Ctrl+Alt+S` to customize the editor's behavior.

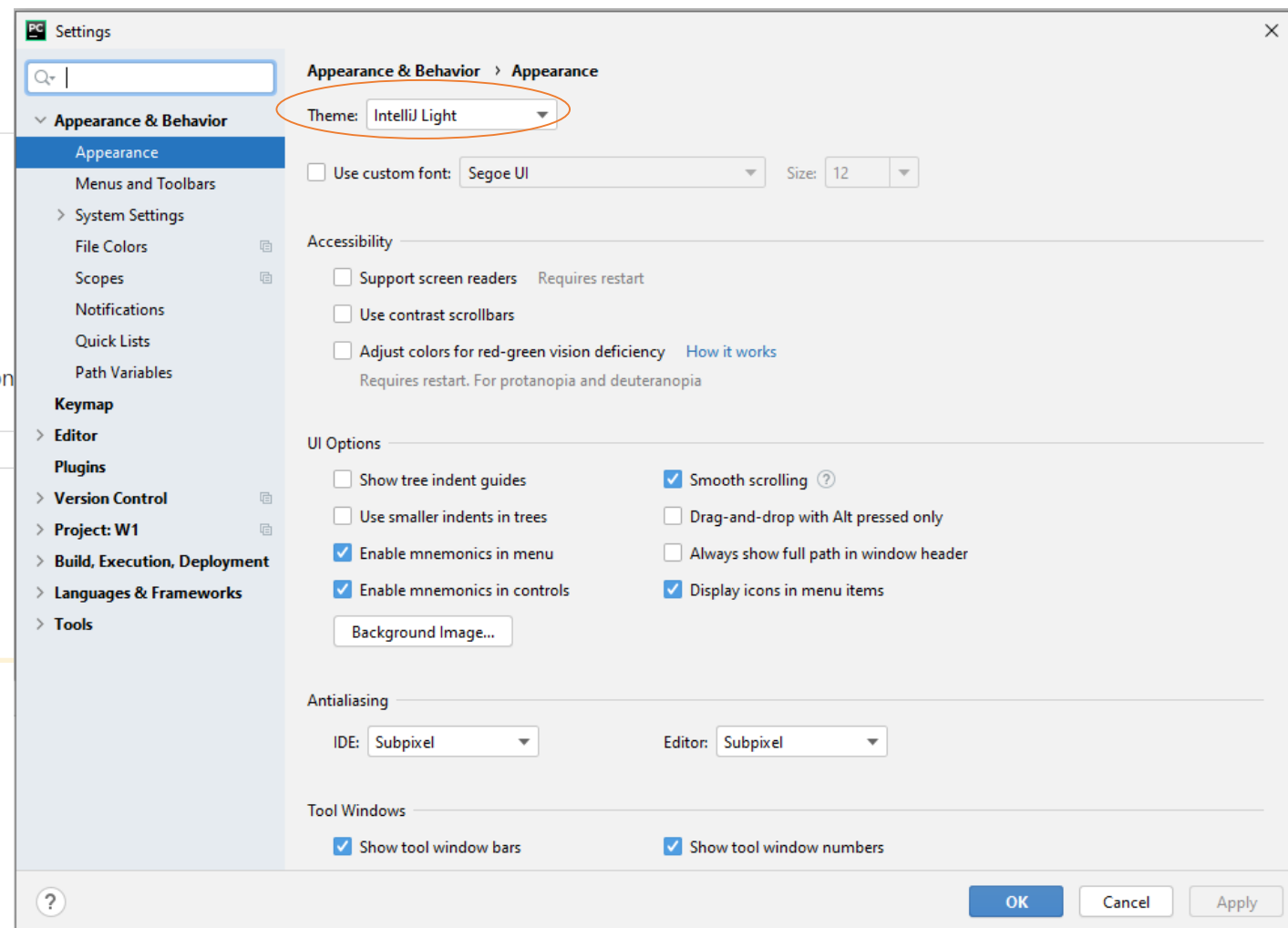
Check the following popular configurations:

Configure code formatting

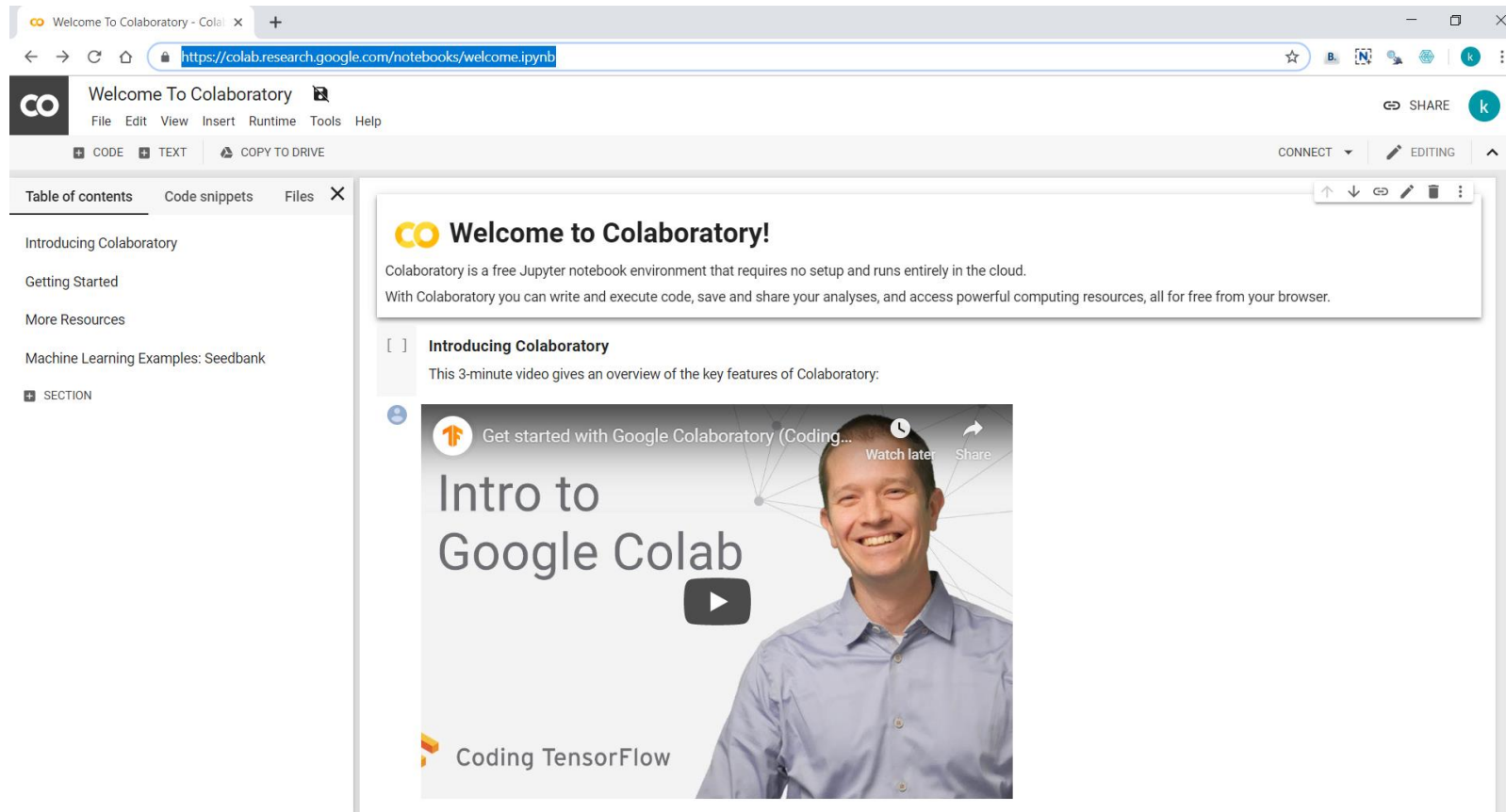
1. In the **Settings/Preferences** dialog `Ctrl+Alt+S`, go to **Editor | Code Style**.
2. From the list of languages select the appropriate one and on the language page, configure settings for tabs and indents, spaces, wrapping and braces, hard and soft margins, and so on.

Configure fonts, size, and font ligatures

- In the **Settings/Preferences** dialog `Ctrl+Alt+S`, go to **Editor | Font**.



Cloud



The screenshot shows the Google Colaboratory (Colab) interface in a web browser. The browser's address bar displays the URL <https://colab.research.google.com/notebooks/welcome.ipynb>. The Colab logo is visible in the top left corner of the interface. The main content area features a "Welcome to Colaboratory!" message, stating that Colab is a free Jupyter notebook environment that runs entirely in the cloud. Below this, there is a section titled "Introducing Colaboratory" which includes a video player. The video player shows a thumbnail for a video titled "Intro to Google Colab" by "Coding TensorFlow". The video player interface includes a play button, a "Watch later" button, and a "Share" button. The left sidebar contains a "Table of contents" section with links to "Introducing Colaboratory", "Getting Started", "More Resources", and "Machine Learning Examples: Seedbank".

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

If google cloud is used in your area, ...

- Go to baidu.com
- Search “Jupyter notebook web”
- Find the right one for you.

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

- 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?

```
[>>> 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

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.
>>>
```

Quiz

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 = "\'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

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

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?

```
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

```
>>> 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
```

Q&A
Thank you