Python

# Master class

## 2. Cài đặt trên windows:

Khi cài nhớ click chọn add 3.6 to path: <https://www.python.org/>



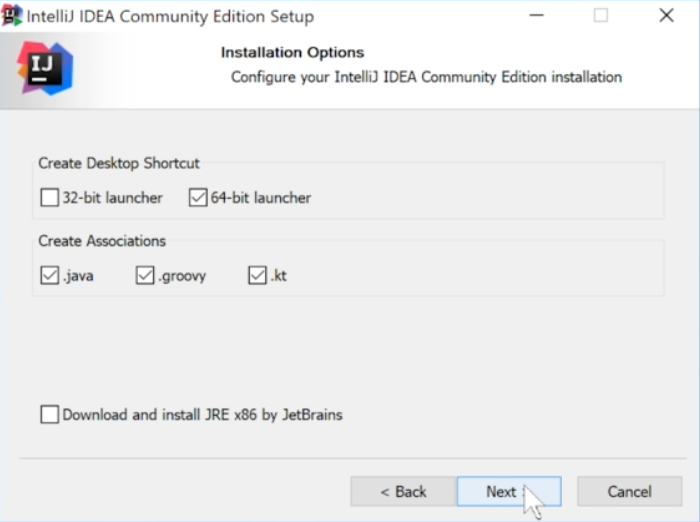
Gõ lệnh: python

exit

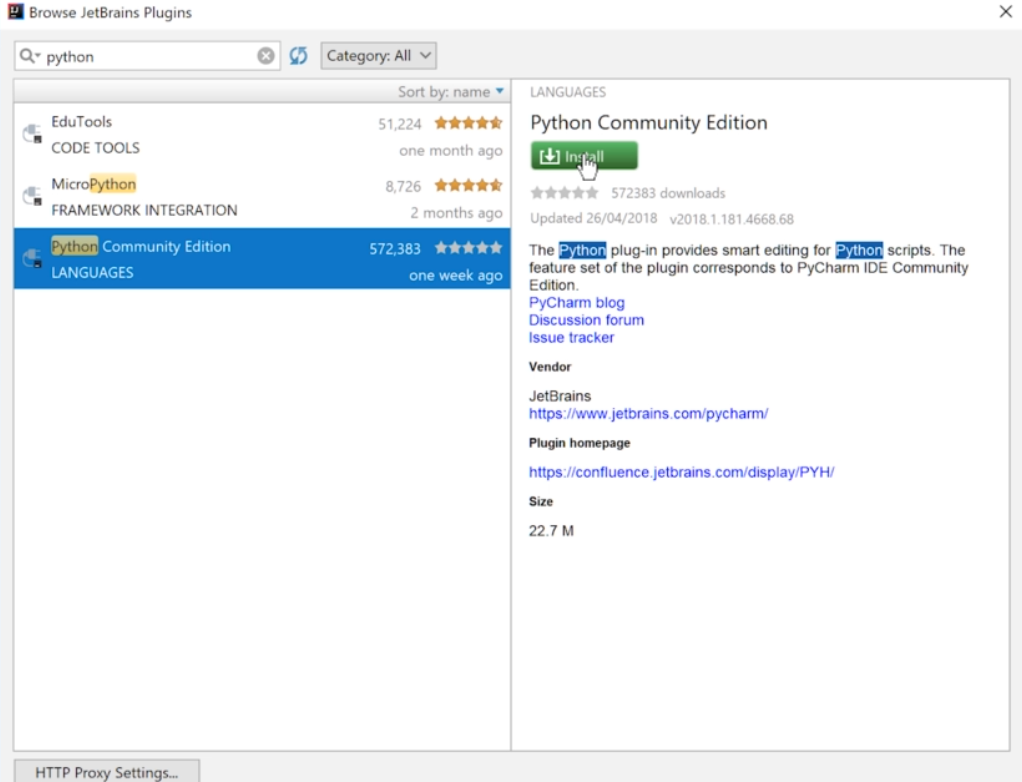
=> để check version hiện tại

**IDE**: <https://www.jetbrains.com/>

Chọn community version for free



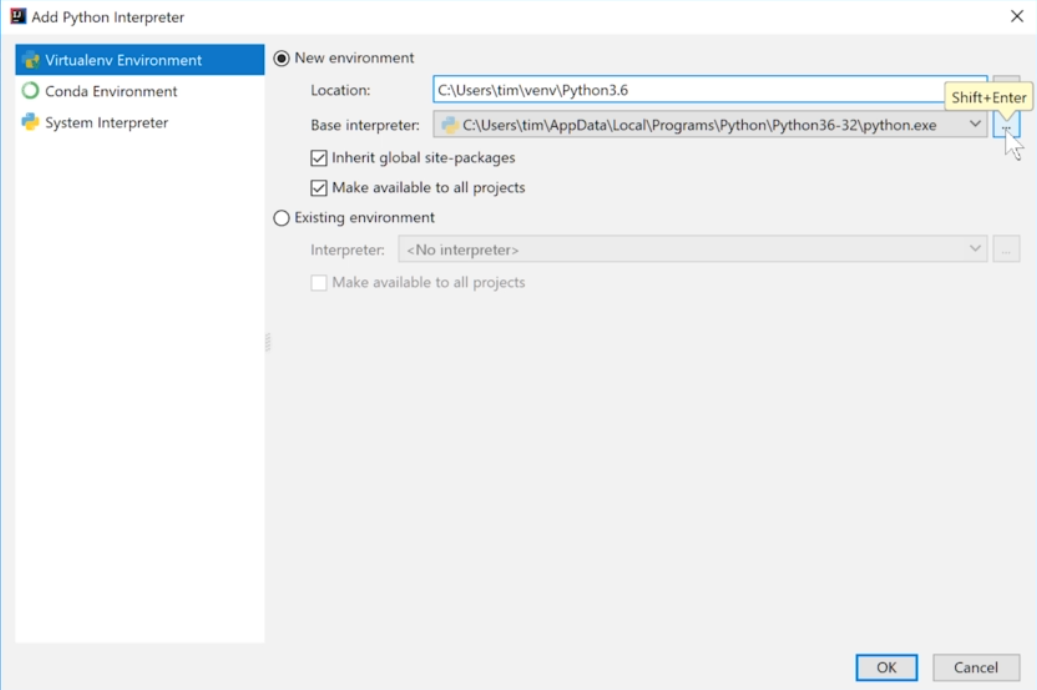
Sau đó **cài plugin cho inteliji** bằng cách ở màn hình chào chọn **settings/plugin**



Sau đó ấn vào nút **Restart** sau khi cài

Config SDK : Vào **Project structure/+/python SDK** trong Inteliji và Vào **setting/ Project interpreter/Add** trong pycharm





Thêm line number : **General/Appreance/show line num…**

## 5. The Basics of Python

Về string format khi print, in các kí tự trong string

age = 24  
print("My age is " + str(age) + " years")  
  
age = 24  
print("My age is {0} years".format(age))

Vào **general/ editor/ code style/** chỉnh indent tick chọn tab và ấn Reset

Phím tắt format CTRL ALT L

Flow if else, loop

# The Modern Python 3 Bootcamp

## 3. WINDOWS Command Line Fundamentals

Slide: acc phuong123 – duyphuong1020

How do I find out where I am?

The cmdlet "**pwd**" (**p**rint **w**orking **d**irectory) will tell you the full absolute path of where you're at!

C:\

Check version of python: python

Navigating *Absolutely*

The command "**cd**" ( "**c**hange **d**irectory") followed by the absolute path of the folder will navigate you directly there.

Navigating *Relatively*

The dot "**.**" stands for current directory, and dot-dot "**..**" stands for parent directory. This allows for relative navigation:

What's Inside?

The keyword "**ls**" will "**l**i**s**t" the contents of a directory.

Creating Directories

The command "**mkdir"** ("**m**a**k**e **dir**ectory") followed by the name of the new directory will create a new child directory inside the current directory.

Creating Files

**New-Item -ItemType file filename.txt** followed by the filename and file-type extension will create a new file of that type.

Moving / Renaming Things

Files can be moved or renamed using the "**mv**" (" **m**o**v**e") keyword, which takes two arguments: the source and the destination.

Removing Files

Files can be deleted using the "**rm**" ("**r**e**m**ove") keyword.

Removing Directories

Directories can be deleted using the "**rm**" keyword, with the added option "**-r**" ("**r**ecursive"). You can also add the "**-fo**" ("**fo**rce") to prevent warnings.(rm -r -fo foldername)

Slide: <http://python.slides.com/colt/command-line-28#/>

## 6. Numbers, Operators, and Comments

Variables can be:

1. assigned to other variables
2. reassigned at any time
3. assigned at the same time as other variables

Naming Restrictions

* Variables must start with a letter or underscore

\_cats - 2cats

* The rest of the name must consist of letters, numbers, or underscores

cats2 - hey@you

* Names are case-sensitive

Slide: <http://python.slides.com/colt/oop-21#/12>

Naming Conventions

* Most variables should be **snake\_case** (underscores between words)
* Variables that start and end with two underscores (called "dunder" for **d**ouble **under**score) are supposed to be private or left alone
* Most variables should be also be **lowercase**, with some exceptions:
* **UpperCamelCase** usually refers to a class (more on that later)
* **CAPITAL\_SNAKE\_CASE** usually refers to constants (e.g. PI = 3.14): \_\_no\_touchy\_\_

**data type description**

bool True or False values

int an integer (1, 2, 3)

str (string) a sequence of Unicode characters

list an ordered sequence of values of other data types, e.g. [1, 2, 3] or ["a", "b", "c"]

dict a collection of key: values, e.g. { "first\_name": "Colt", "last\_name": "Steele" }

Dynamic Typing

Python is highly flexible about reassigning variables to different types:

Declaring Strings

String literals in Python can be declared with either single or double quotes.

Either one is perfectly fine; but make sure you stick to the same convention throughout the same file.

String Escape Characters

In Python there are also "escape characters", which are "metacharacters" - they get interpreted by Python to do something special: \

String Concatenation

Concatenation is combining multiple strings together. In Python you can do this simply with the "+" operator

Formatting Strings

There are also several ways to format strings in Python to **interpolate** variables.

The new way (new in Python 3.6+) => **F-Strings**

x = 10

formatted = f"I've told you {x} times already!"

The tried-and-true way (Python 2 -> 3.5) => **.format** **method**

x = 10

formatted = "I've told you {} times already!".format(x)

The old way => **% operator** (deprecated)

x = 10

formatted = "I've told you %d times already!" % (x)

Converting Data Types

In string interpolation, data types are implicitly converted into string form (more on this later in OOP).

You can also explicitly convert variables by using the name of the builtin type as a function (more on functions later):

decimal = 12.56345634534

integer = int(decimal) # 12

my\_list = [1, 2, 3]

my\_list\_as\_a\_string = str(my\_list) # "[1, 2, 3]"

## 7. Variables and Strings

## 8. Boolean and Conditional Logic

<http://python.slides.com/colt/variables-and-data-types>

Conditional Statements

if some condition is True:  
    do something  
elif some other condition is True:  
    do something  
else:  
    do something

In Python, all conditional checks resolve to **True** or **False**.

Truthiness

x = 1

x is 1 # True

x is 0 # False

We can call values that will resolve to True "**truthy**", or values that will resolve to False "**falsy**".

Besides False conditional checks, other things that are naturally falsy include: empty objects, empty strings, *None*, and zero.

Comparison Operators

|  |  |  |
| --- | --- | --- |
| **Op** | **What it does** | **Example** |
| == | Truthy if **a** has the same value as **b** | a **==** b  # True |
| != | Truthy if **a** does **NOT** have the same value as **b** | a **!=** b  # False |
| > < | Truthy if **a** is greater than **b** Truthy if **a** is less than be **b** | a **>** b  # False a **<** b  # False |
| >= <= | Truthy if **a** is greater than or equal to **b** Truthy if **a** is less than or equal to **b** | a **>=** b  # True a **<=** b  # True |

Logical Operators: and or not

is vs. "=="

In python, "**=="** and "**is**" are very similar comparators, however they are not the same.

a = 1

a == 1 # True

a is 1 # True

a = [1, 2, 3] # a list of numbers

b = [1, 2, 3]

a == b # True

a is b # False

c = b

b is c # True

## 10. Looping in Python

<http://python.slides.com/colt/functions-part-1#/>

<http://python.slides.com/colt/lists-13#/>

<http://python.slides.com/colt/lists-22#/>

<http://python.slides.com/colt/lists-23>

<http://python.slides.com/colt/generators-and-decorators#/14>

<http://python.slides.com/colt/lists-24#/>

<http://python.slides.com/colt/testing-25#/>

<http://python.slides.com/colt/lists-18#/>