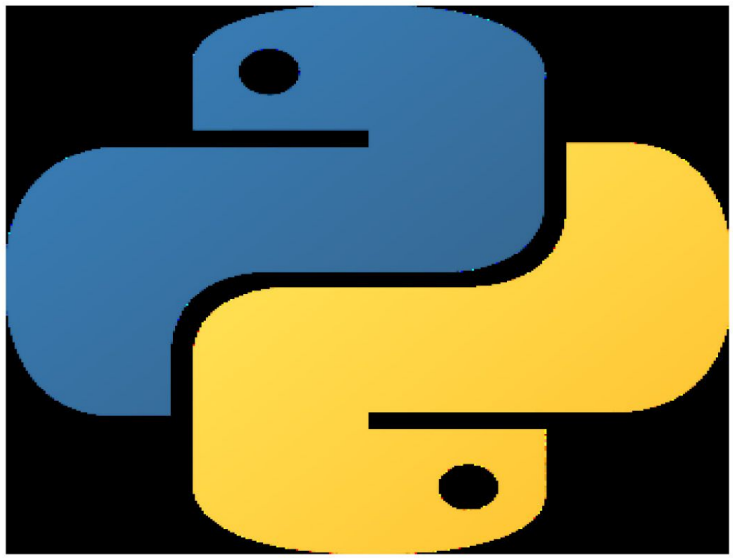


Programming with Python

Introduction to Python

- Python is an interpreted, object-oriented, high-level programming language .
- Designed By: **Guido van Rossum**
- Name came from a 1970s British television show: *Monty* ***Python's Flying Circus***
- First released in 1990

Why Python

- **Open source, so anyone can contribute to its development**
- Suitability for everyday tasks, **allowing for short development time**
- **Extensible: Reusable code using modules and packages**
- Python is processed at runtime by the interpreter. There is no need to compile your program before executing it.

Why Python

- **Very easy to install and Setup**
- **Easy to learn**
- **Readable**
- **Simplicity**
- **Multipurpose**

Why Python

- **Python code is typically one-third to one-fifth the size of equivalent C++ or Java code. That means there is less to type, less to debug, and less to maintain after the fact.**

- **Support Libraries It has portable standard library, and many more free packages. Component Integration Python code can invoke C and C++ libraries, can be called from C and C++ programs, can integrate with Java and .NET components.**

- **Enjoyment Because of Python's ease of use and built-in toolset, it can make the act of programming more pleasure**

Why Python..Portable

- **Portability: Most Python programs run unchanged on all major computer platforms. Porting Python code between Linux and Windows, for example, is usually just a matter of copying a script's code between machines.**

- Linux
- Windows
- Mac
- HPC Clusters
- Cell phones running Symbian OS, and Windows Mobile
- Tablets and smartphones running Google's Android and Apple's iOS

How is Python developed and Supported?

- A popular **open source system, has very active development community.**
- **Developed by: The *PSF (Python Software Foundation), a formal nonprofit group, organizes conferences and deals with intellectual property issues.***
- ***License – PSF license***

- PSF conducts PyCon conference.
- **PEP – Python Enhancement proposals**

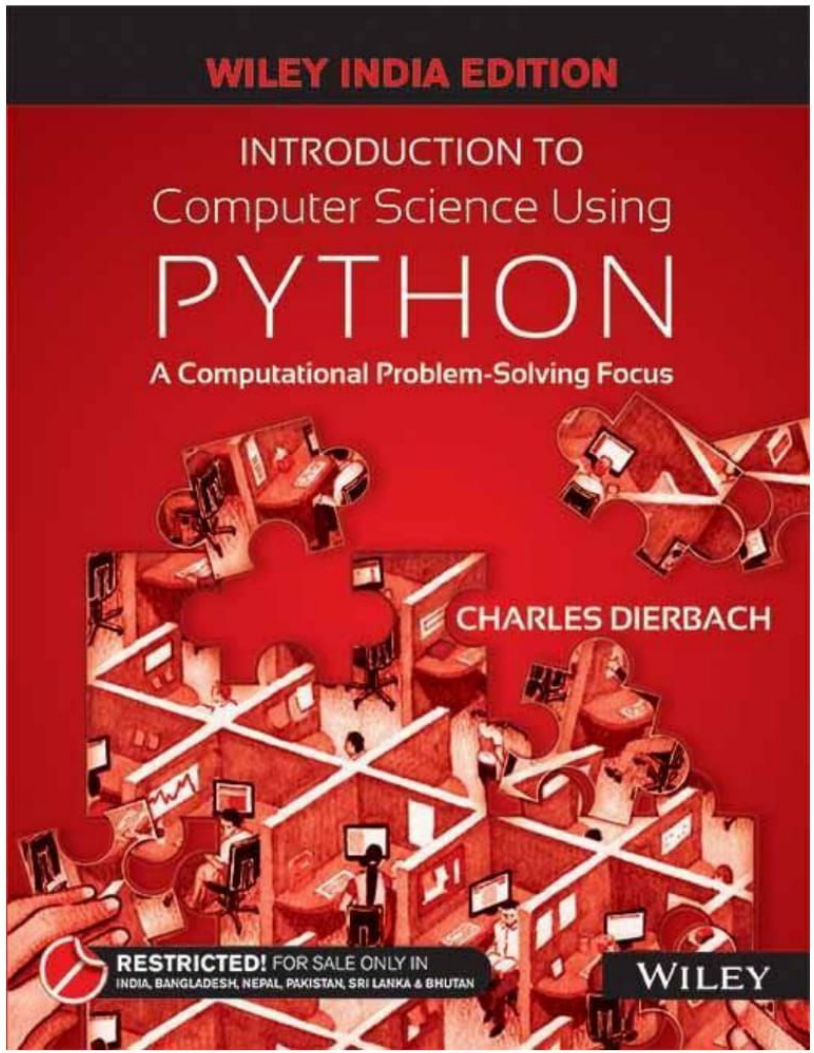
PEP

- https://www.python.org/dev/peps/

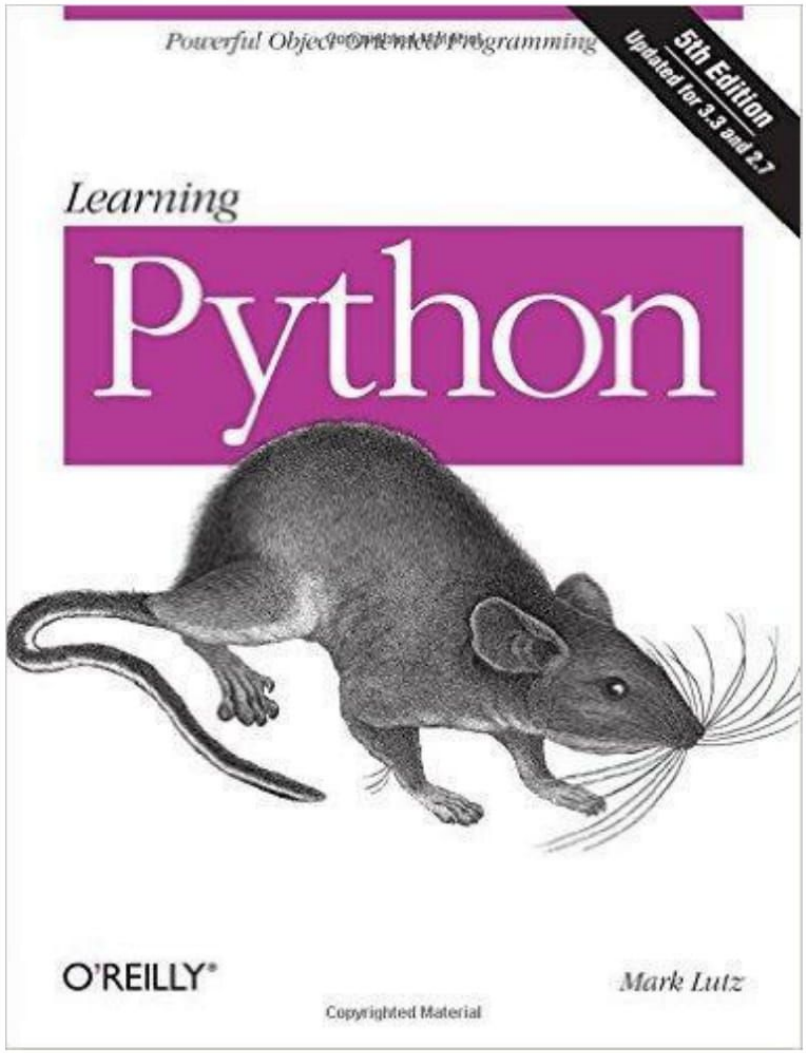
- This PEP contains the index of all **Python Enhancement Proposals, known as PEPs.**

- PEP numbers are assigned by the PEP editors, and once assigned are never changed.

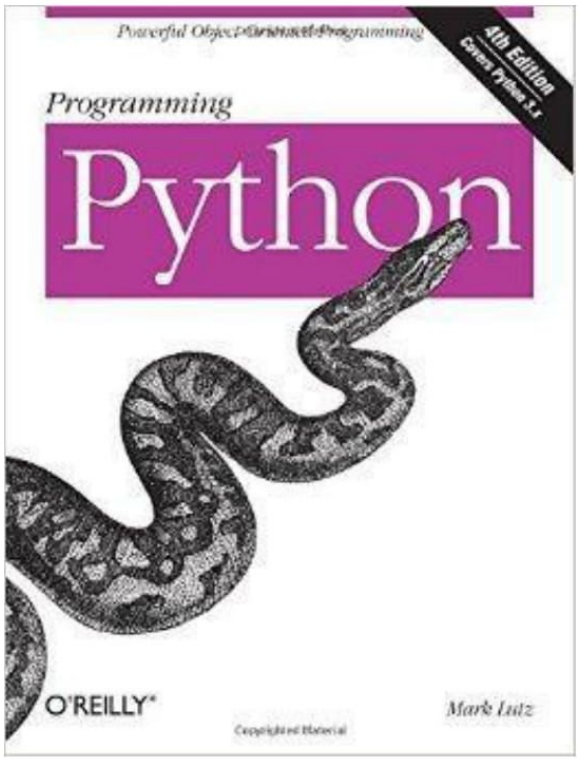
Python Material (Level 1) Text Book



Python Material (Level 2) Text Book



Python Material (Level 3) Text Book



Online Material

- **Official Python Documentation:**
- https://docs.python.org/3/

- https://www.python-course.eu/

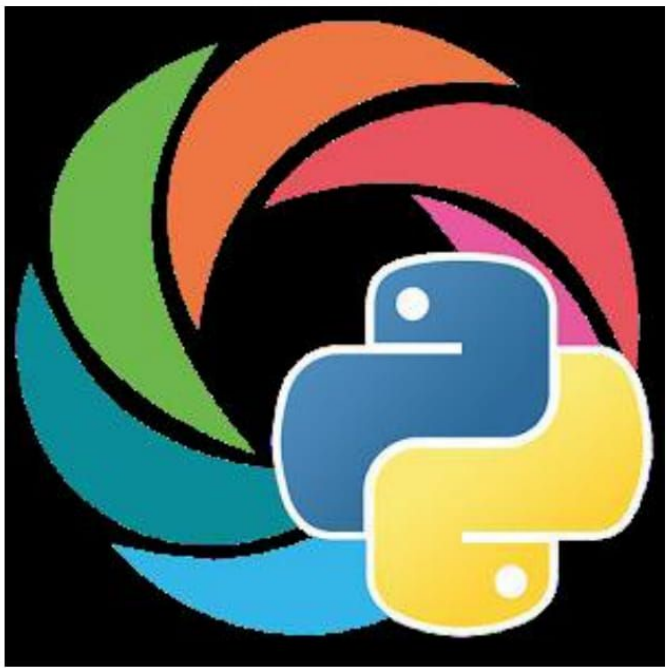
- http://www.sololearn.com/Course/Python/

- https://www.learnpython.org/

- https://thepythonguru.com/

- Success Stories: https://www.python.org/about/success/

“Sololearn Python“ (A Simple Android App for Beginners)



Application Areas

- web programming
- Scripting
- scientific computing
- Artificial Intelligence

Python Versions

- The three major versions of Python are 1.x, 2.x and 3.x.
- These are subdivided into minor versions, such as 2.7 and 3.3.
- Code written for Python 3.x is guaranteed to work in all future versions.
- Both Python Version 2.x and 3.x are used currently.

- Python2 – Very Stable (Python-2.7 Mid 2010) – No release after that.

- Python3 (Python3.0 in 2008)

- Current Release – 3.5.6 (Released on 2-08-2018)

- Some major changes and clean-ups
- Not backward compatible (cannot execute 2.x code)
- What is new in Python3? - https://docs.python.org/3/whatsnew/3.0.html
- Many important packages not yet ported to Python 3.
- We use Python3 in this course

Comparison with other languages

- Python code is typically **3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code!**

Users

- **Google – Many components of search engine were written in Python**
- **YouTube - *YouTube video sharing service is largely written in Python.***
- **Yahoo - maps were developed using Python**
- **RHEL – Installer developed using Python**
- **NASA – Uses Python as the main scripting language**
- **Pixar - and others use Python in the production of animated movies.**

So how do I get started?

- **Linux: (Debian/Ubuntu)**

  $ sudo apt-get install python3
- **Windows:**
- Download from: https://www.python.org/downloads/
- Click on the executable and install it.

- **Installing Anaconda**
- Download the Anaconda installer for Linux
- In your terminal window type the following:

bash Anaconda4-3.1.0-Linux-x86_64.sh

- add this line to the .bashrc in your home directory:
- export PATH="/home/username/anaconda/bin:$PATH"
- Replace /home/username/anaconda with your actual path.
- Finally, close and re-open your terminal window for the changes to take effect.

Indentation

- **Do not use tabs at all for indentation – Makes your code non-portable**
- Use 3 or **4 spaces**
- Compiler ignores blank lines
- Line continuation character: ****

Start Coding

- print(‘Welcome to Pacific University”)

- Output –
- Welcome to Pacific University

How to code in editor

- Open any editor

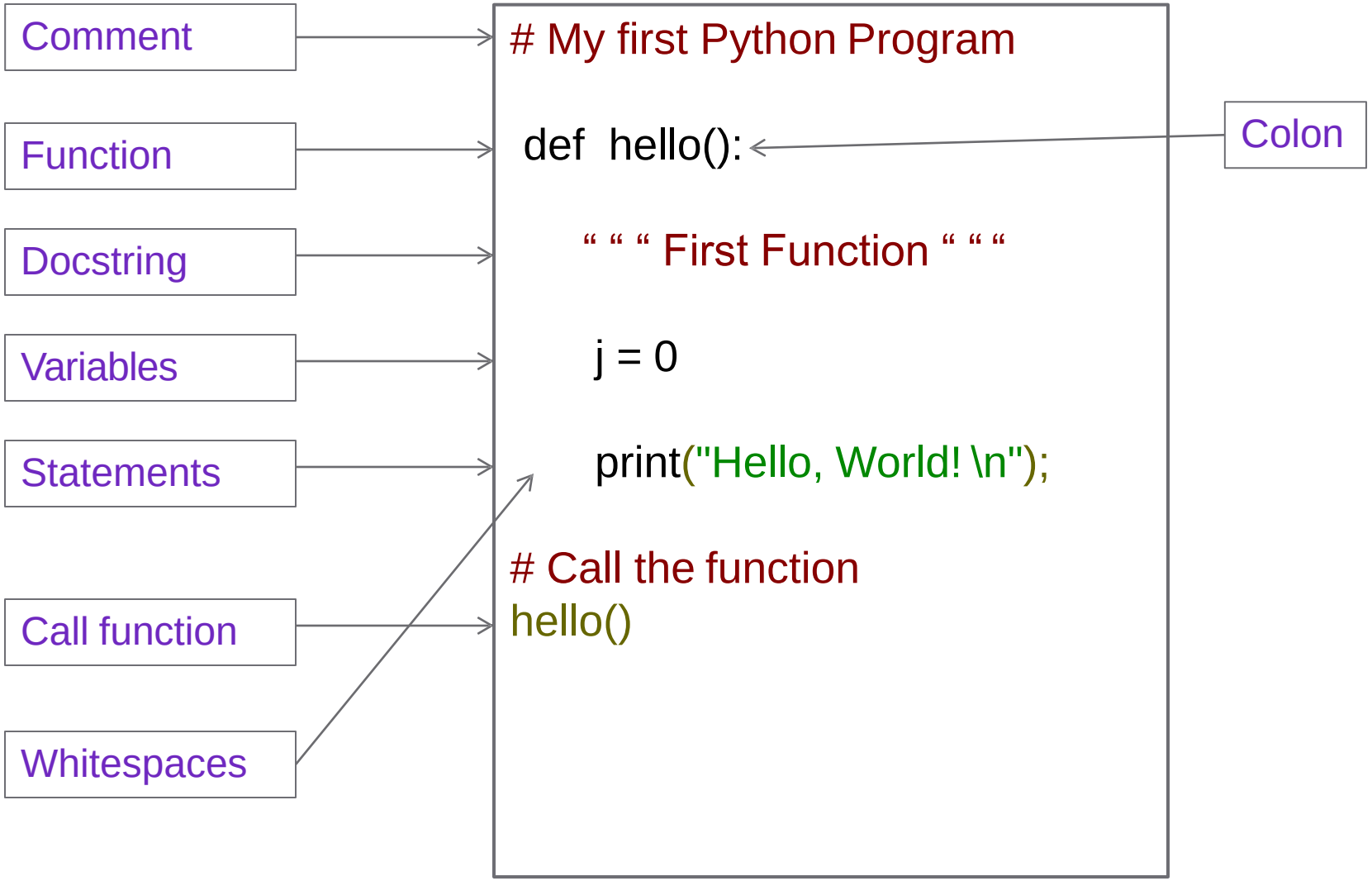
  vi ankita.py

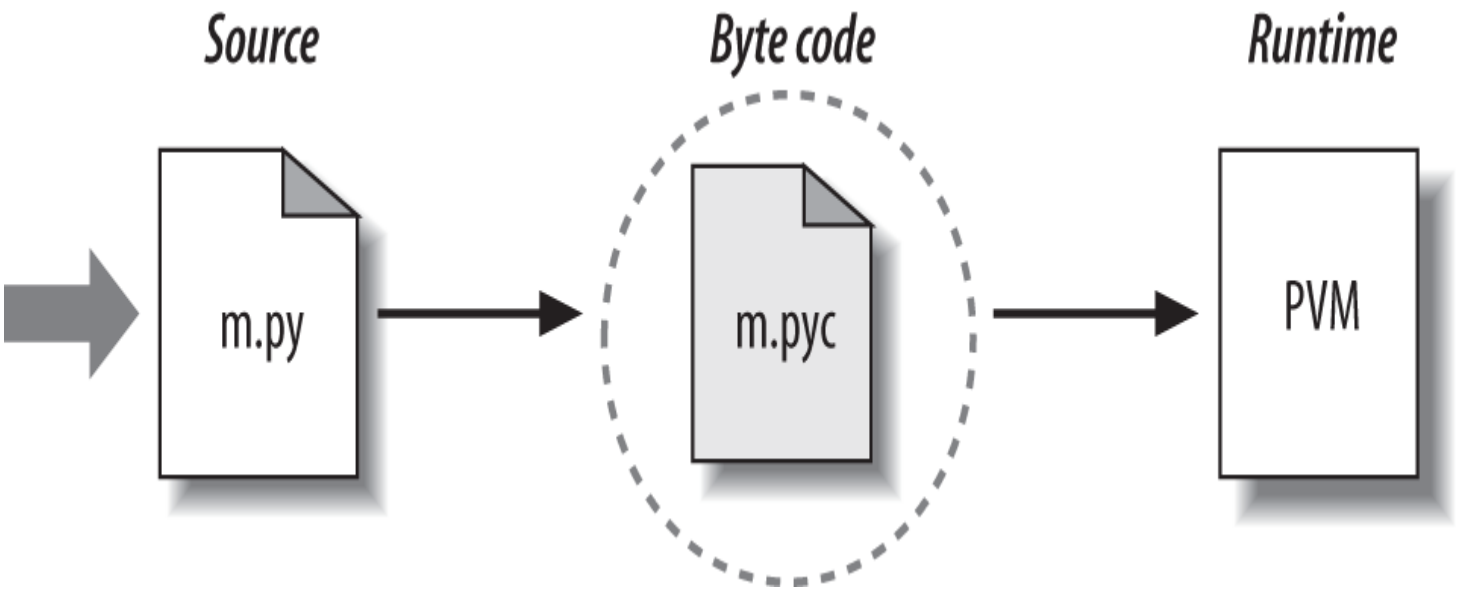
Save the script by – esc : wq

Run this by –

python3 ankita.py

A Simple Python Program – hello.py




Source
Byte code
Runtime
m.py
m.pyc
PVM

1.Generate Byte Code

- Python Interpreter translates your source code into machine independent byte code (.pyc)
- Stores .pyc file __PyCache__ folder
- When you run the same program (Without changes) then it will use this byte code without translating it again.

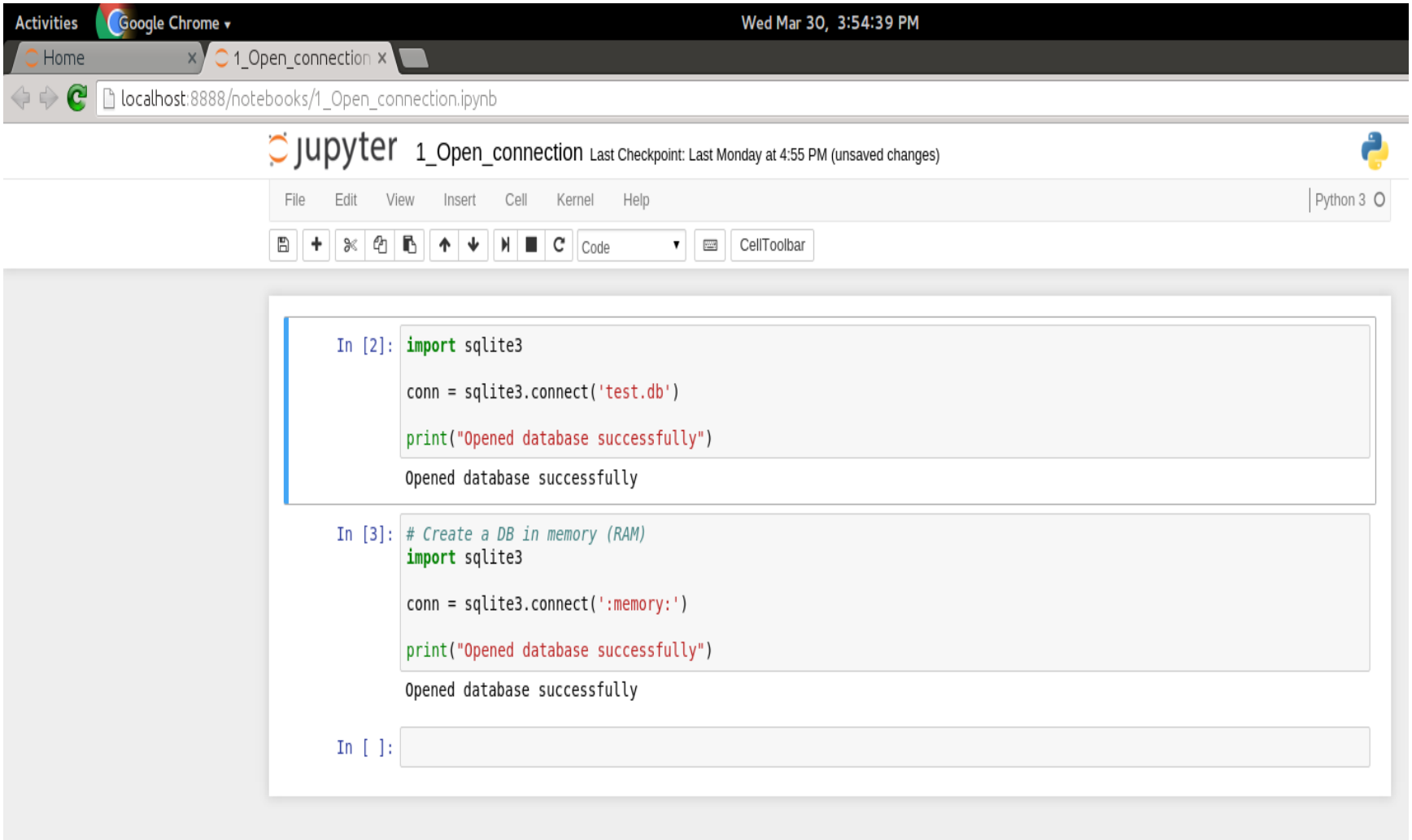
2.Run on PVM

- Byte Code (.pyc) will be shipped to PVM
- It executes the code

Python shells:

- IDLE shell
- **Ipython/Jupytor**
- Eclipse
- **vim**
- nano

Jupyter Notebook



Converting notebook to script

- jupyter-nbconvert ankita.ipynb --to script
- It will create ankita.py

How to use input, print?

- *>>>name =* ***input('What is your name?:')***
- *What is your name?: Ankita*
- *>>>****print(name)***
- *Ankita*

- All input is returned by the input function as a string type.

- For the input of numeric values, the response must be converted to the appropriate type.
- Python provides built-in type conversion functions **int( ) and float( ) for this purpose,**

- *>>> salary =* ***input('What is your salary?:')***
- *What is your salary?:94532.50*
- *>>>type(salary)*
- *<class 'str'>*
- *>>> salary =* ***float(salary)***
- *>>>* ***type(salary)***

- *>>> salary =* ***input("What is your salary?:")***
- *What is your salary?:94532.50*
- *>>>type(salary)*
- *<class 'str'>*
- *>>> salary =* ***float(salary)***
- *>>>* ***type(salary)***

Comments & Docstrings

- **Single Line Comments:**

- *# Print a welcome message*
- *print ('Hello world !') # Be friendly…write comments*
- **Doc Strings:**
- Used to describe a function, module, method, class etc.
- Used for code documentation
- These are interpreted, use if necessary.

- *def sum(a, b):*
- *“ “ “Compute the sum of two integers and return to the calling function “ “ “*
- *return a+b*
- *sum.__doc__ # Print docstring of the function sum*

Good Programming Practices

- Do not use semicolons, they are legal but unnecessary
- x = 5
- x = 5;
- Both are correct.
- Limit lines to 79 characters
- Python is case sensitive
- All key words are case sensitive
- Class Name should be written in CamleCase
- Every thing else should be in lower case (Underscore is acceptable)
- For more details see **PEP8 (Python Enhanced Proposals – 8).**

Variable Rules

- Must start with a letter or underscore _
- Must consist of letters and numbers and underscores
- Case sensitive
- Should not start with number
- **Examples:**
- *Good: dsbda pi pi34 _speed*
- *Bad: 23spam #sign var.12*

Reserved words

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	

Operators, operands and expressions

- Operators are symbols which take one or more operands or expressions and perform arithmetic or logical computations. Examples: +, -, *, /, =, ==

- An "operand" is an entity on which an operator acts. Example: a+b-5 a, b and 5 are operands
- An "expression" is a sequence of operators and operands that performs any combination of these actions:
- Computes a value
- Designates an object or function
- Example: a + (b / 6.0)

Operators

Python language supports the following types of operators:

Type	Operators	Examples
Arithmetic Operators	+, -, *, / Modulus - % Exponent - ** Flat Division - //	21 % 10 = 1 21 // 10 = 2 (Fraction is omitted) 21.0 // 10.0 = 21.0/10 = 21/10.0 = 2.0
Comparison (Relational) Operators	==, !=, <, >, >=, <= < > (Python2 only)	20 != 10 O/P: True
Assignment Operators	=, +=, -+, *=, /+, %=, **=, //=	a=24 a//=10 → a = a//10
Logical Operators	and, or, not	(10 and 20)
Bitwise Operators	&, |, ^ (XOR), <<, >> ~ (1‟s complement),	~(2) = -3
Membership Operators	in not in	x in y → Return 1 if x is a member of sequence y.
Identity Operators	is is not	x is y → True if x and y point to the same object.

Operator Precedence

tuple, list, dictionary, string	(), [], {}, „“
attribute, index, slide, function call	x.attr, x[], x[i:j], f()
Exponentiation	**
Bitwise not	~x
Positive and negative	+x, -x
Multiplication, division, remainder	*, /, %
Addition, subtraction	+, -
Bitwise shift	<<, >>
Bitwise AND	&
Bitwise XOR	^
Bitwise OR	|
Comparisons, membership, identity	<, <=, >, >=, <>, !=, ==, in, not in, is, is not
Boolean Not	not x
Boolean AND	and
Boolean OR	or
Lambda Expression	lambda

High

Low

Data Types

- Elementary Types
  - **NoneType**: None
  - **bool**: True, False
  - **int**: 42
  - **float**: 3.14, complex: (0.3+2j)
- Container or Sequence Types
  - **str**: 'Hello,, (Sequence of Unicode characters)
  - **list**: [1, 2, 3]
  - **tuple**: (1, 2, 3)
  - **dict**: {'A':1, 'B':2}
  - **set**: {1, 2, 3}
- file
- function, class, instance

Note: In Python everything is an object and every object has a type.

Dynamic Typing

- Type checks (making sure variables have the correct type for an operation) performed at runtime.
- No need to declare variable types.
- We can obtain the type of an object with `type(variable)„.

```
>>> x = 5*4
# create a new string object and let x point to it.
>>> x = „twenty'
>>> x
„twenty'
>>> type(x)
<type 'str '>
```

Strong Typing

- Operations may expect operands of certain types.
- Interpreter throws an exception if type is invalid.

```
>>> a = 5
>>> b = 'Python'
>>> a + b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int'
and 'str'
```

Lab Exercises…

Write a Python program that prompts the user to enter an upper or lower case letter and displays the corresponding Unicode encoding.

Develop and test a program that prompts the user for their age and determines approximately how many breaths and how many heartbeats the person has had in their life. The average respiration (breath) rate of people changes during different stages of development. Use the breath rates given below for use in your program:

	Breaths per Minute
Infant 1–	30–60
4 years	20–30
5–14 years	15–25
Adults	12–20

For heart rate, use an average of 67.5 beats per second.