CAH11-01 Networks and Operating Systems

Python Fundamentals

Aims of the Seminar

In this module, we will use Python to develop networking applications. Therefore, it is essential to have a strong foundation in Python and to practice its core concepts before delving into advanced topics. Building a solid understanding of the basics will ensure you are well-prepared to tackle more complex challenges in networking development.

Introduction to Python Programming: We'll kick things off by building a solid foundation in Python. Python is a powerful and versatile programming language widely used in AI development due to its readability and extensive libraries.

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990.

Python 3 is available for Windows, Mac OS and most of the flavours of the Linux operating system.
Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your script before executing it.

Python is Interactive − You can interact with the python interpreter directly.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Feel free to discuss your work with peers, or with any member of the teaching staff.

Reminder

We encourage you to discuss the content of the workshop with the delivery team and any findings you gather from the session.

Workshops are not isolated, if you have questions from previous weeks, or lecture content, please come and talk to us.

Exercises herein represent an example of what to do; feel free to expand upon this.

Helpful Resources

The programming language that we will be using is Python which is a great for learning AI basics. In this workshop we will be using functions, lists, conditions and any other basic python commands. If you would like a refresh on these concepts, you can check out one of the following tutorials:

- Overview of Python http://www.tutorialspoint.com/python/python_quick_guide.htm
- Good starting course https://www.tutorialspoint.com/python/python_overview.htm
- Python Online Interactive http://www.learnpython.org/en/Hello%2C_World%21
- Python Hard Way http://learnpythonthehardway.org/book/ex20.html
- Python http://anh.cs.luc.edu/331/notes/PythonBasics.pdf
- Python http://learnpythonthehardway.org/book/ex20.html

The 2 courses in bold will be particularly useful if you are new to this coding environment.

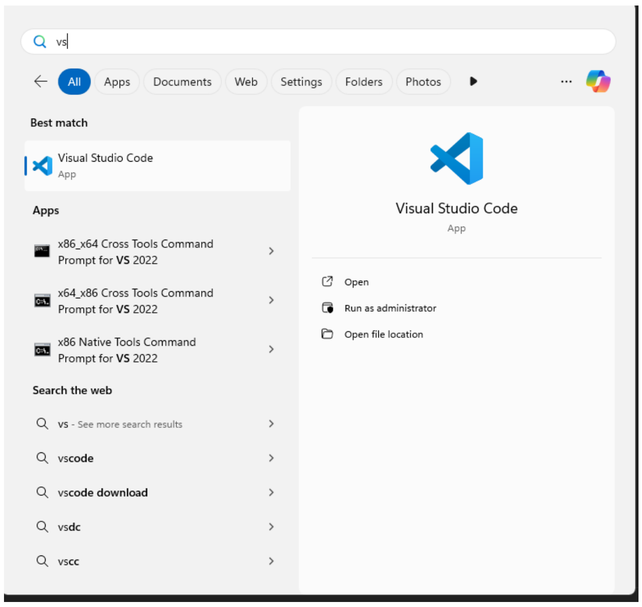
To write the scripts we will be using Jupyter notebooks, which is a helpful IDE for writing the scripts. To download it, you can use the link below:

https://realpython.com/jupyter-notebook-introduction/

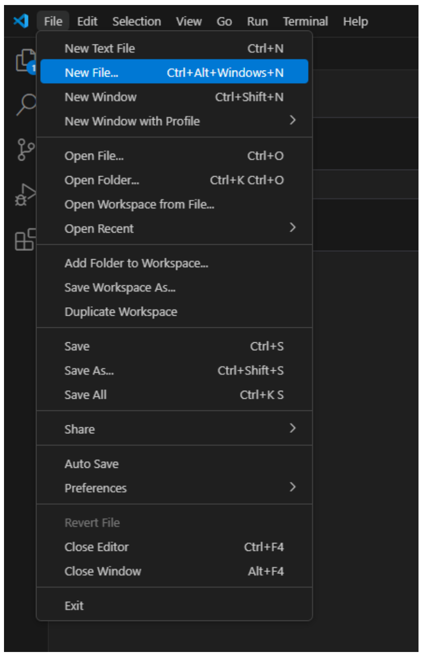
1. **Install VS code:**

   https://code.visualstudio.com/download

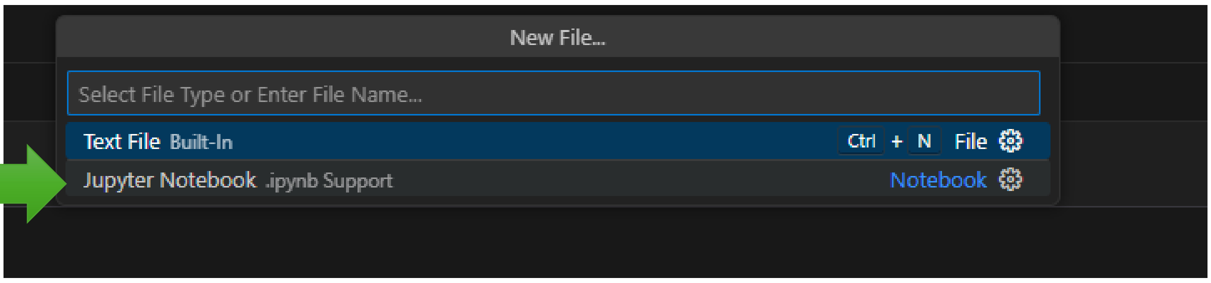
2. **Launch VScode**



3. **Install Python extension**
4. **Install Jupyter Notebook extension**
5. **Start a new file**



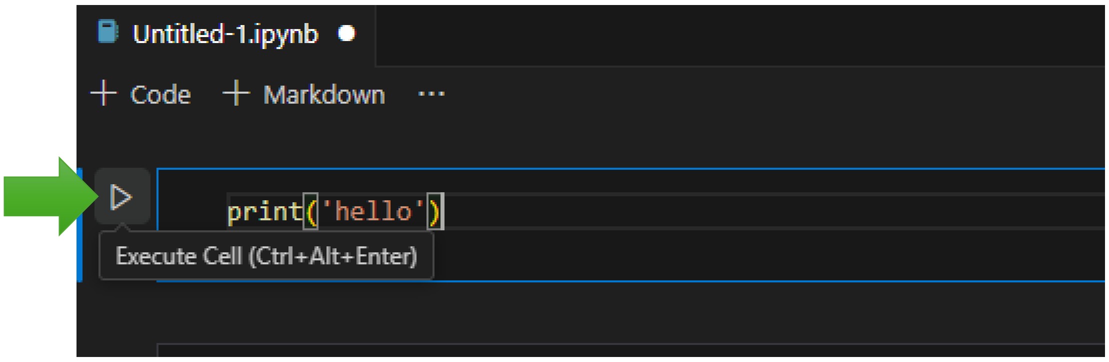
6. **Choose Jupyter Notebook**


New File...
Select File Type or Enter File Name...
Text File Built-In
Ctrl + N
File
Jupyter Notebook .ipynb Support
Notebook

7. **Type**

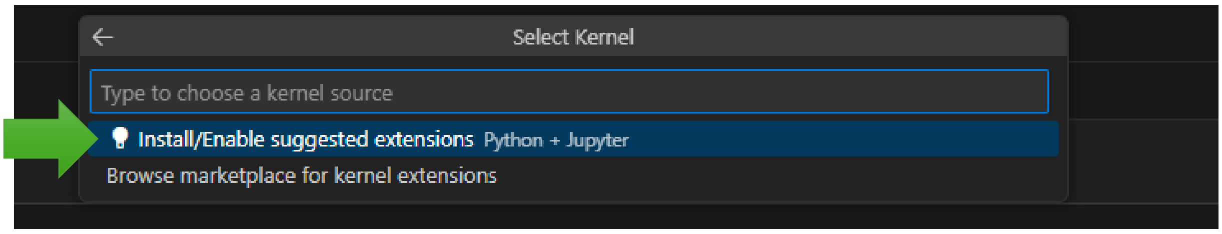
```
print('Hello Word')
```

8. **Then click Run**

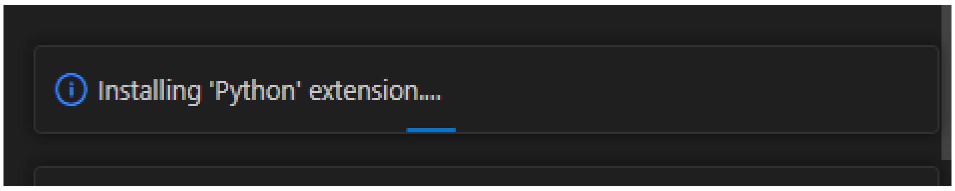


```
print('hello')
```

You will need to instal the extension

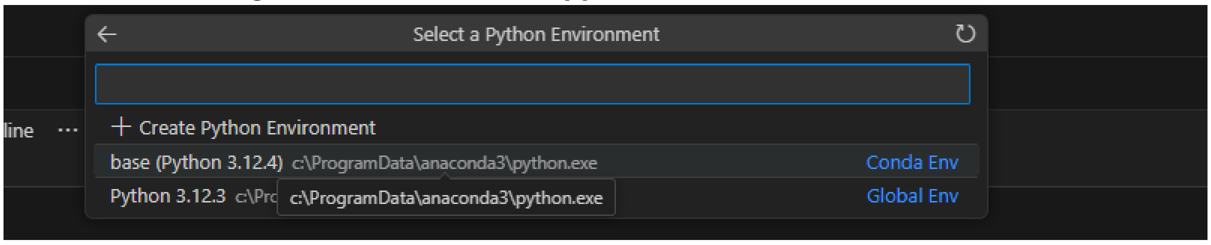

Select Kernel
Type to choose a kernel source
Install/Enable suggested extensions Python + Jupyter
Browse marketplace for kernel extensions

It will take a few minutes to instal and show this notification



Installing 'Python' extension....

Run Hello Word again and choose base python


Select a Python Environment
line ···
+ Create Python Environment
base (Python 3.12.4) c:\ProgramData\anaconda3\python.exe
Conda Env
Python 3.12.3 c:\Pro c:\ProgramData\anaconda3\python.exe
Global Env

Now your PC is ready for programming 😊

Exercises

You may find it useful to keep track of your answers from workshops in a separate document, especially for any research tasks.
Where questions are asked of you, this is intended to make you think; it would be wise to write down your responses formally.

Exercise 1: We will start with a simple introduction to Python Programming. Please go through Jupyter notebook in learn.gold page:
https://learn.gold.ac.uk/pluginfile.php/2473460/mod_resource/content/1/Intro_to_Python.ipynb?forcedownload=1

Revision: Basic Math Operations

i. Write a script to add two numbers and print the result.
ii. Write a script to subtract two numbers and print the result.
iii. Write a script to multiply two numbers and print the result.
iv. Write a script to divide two numbers and print the result.
v. Write a script to calculate the remainder of two numbers and print the result.
vi. Write a script to calculate the power of a number and print the result.

Exercise 2: Python Data Types

i. Create a list of 5 different fruits and print the list.
ii. Create a tuple of 3 different animals and print the tuple.
iii. Create a dictionary of 3 different key-value pairs where the keys are names of programming languages and the values are their release dates.
iv. Create a set of 3 different colours and print the set.
v. Create a string variable that contains the sentence "Python is an easy-to-learn programming language."

Exercise 3: Conditions

Write a script that takes a user input of a number and checks if it is positive, negative or zero. Print the results.
Write a script that takes a user input of a string and checks if it starts with a vowel or consonant. Print the results.
Write a script that takes a user input of two numbers and checks if they are equal or not. Print the results.

Exercise 4: Loops

i. Write a script that prints the numbers from 1 to 10 using a for loop.
ii. Write a script that prints the even numbers from 2 to 10 using a for loop.

iii. Write a script that prints the first 5 numbers of the Fibonacci sequence using a while loop.

Exercise 5: Functions

i. Write a function that takes two numbers as parameters and returns their sum.
ii. Write a function that takes a list of numbers as a parameter and returns the largest number in the list.
iii. Write a function that takes a string as a parameter and returns the number of vowels in the string.

Exercise 6: Password Strength Checker

In this exercise, participants will create a script that checks the strength of a password based on certain criteria. Write a Python script that checks the strength of a password based on the following criteria:

1. The password must contain at least 8 characters.
2. The password must contain at least one uppercase letter.
3. The password must contain at least one lowercase letter.
4. The password must contain at least one special character (e.g. @, #, $, %, etc.).

If the password meets all the above criteria, print "Password is strong". If the password fails to meet any of the above criteria, print a message indicating which criteria the password failed to meet.

Example Input/Output:

Input:
password = "Pa$$word123"
Output:
Password is strong

Input:
password = "password"
Output:
Password must contain at least one uppercase letter, one special character, and be at least 8 characters long.

Hints:

- Use the len() function to check the length of the password.
- Use the isupper(), islower(), and isdigit() methods to check if a character is uppercase, lowercase, or a digit, respectively.

- Use the in keyword to check if a character is a special character.

<u>Exercise 7</u>: Caesar Cipher

In this exercise, create a script that encrypts and decrypts messages using the Caesar cipher.

The Caesar cipher is a simple encryption technique that involves shifting each letter of the message by a certain number of positions down the alphabet. For example, if we shift the letter "A" by 3 positions, we get the letter "D". Similarly, if we shift the letter "B" by 2 positions, we get the letter "D".

Write a Python script that takes a message and a key as inputs and encrypts/decrypts the message using the Caesar cipher. The key is the number of positions each letter should be shifted down the alphabet.

Example Input/Output:

```
Input:
message = "HELLO"
key = 3
mode = "encrypt"
Output:
KHOOR

Input:
message = "KHOOR"
key = 3
mode = "decrypt"
Output:
HELLO
```

Hints:

- Use the ord() function to get the ASCII value of a character.
- Use the chr() function to get the character corresponding to an ASCII value.
- Use the modulo operator (%) to wrap around the alphabet when shifting letters.

<u>Exercise 8</u>: Hangman Game

Write a Python script that plays the game of Hangman. The script should randomly select a word from a list of words, and the user should have to guess the letters in the word one by one. The user is allowed a certain number of incorrect guesses before the game is over.

Example Input/Output:

Input:

```
(word list) = ["python", "java", "javascript"]
(max guesses) = 6
```

Output:

```
_ _ _ _ _ _ _ (displayed to user)
Guess a letter: a
_ _ _ _ _ _ _ (displayed to user)
Guess a letter: e
_ _ _ _ _ _ _ (displayed to user)
Guess a letter: i
_ _ _ _ _ _ _ (displayed to user)
Guess a letter: o
_ _ _ o _ _ (displayed to user)
Guess a letter: t
_ _ _ o _ _ (displayed to user)
Guess a letter: p
_ _ _ o _ _ (displayed to user)
Guess a letter: n
_ _ _ o n _ (displayed to user)
Guess a letter: y
_ y _ o n _ (displayed to user)
Guess a letter: h
_ y _ o n h (displayed to user)
Congratulations, you guessed the word "python"!
```

Hints:

Use the random module to randomly select a word from the word list.
Use a loop to iterate over the word and check if the guessed letter is present in the word.
Use a counter to keep track of the number of incorrect guesses.

END OF EXERCISES