CENTER FOR TEACHING & LEARNING

Basics for Python

What is Python: Python is a popular coding language known for its readability, versatility, and simplicity. Coding languages give instructions to computers, letting you control them, automate tasks, and solve problems creatively.

Introduction to Python: It is a high-level interpreted programming language. It is easy to read, has an extensive standard library, and also supports multiple programming paradigms.

Why Python: Python is favored over other programming languages due to its easy-to-read syntax, making it ideal for beginners and quick development. Its versatility spans web development, data analysis, AI, and more. Python's extensive standard library and cross-platform compatibility enhance its efficiency, while strong community support and high demand in the job market further boost its appeal. Here is the link where you can find few resources about python and can access online versions of them.
UNM Libraries Python resources

What is IDE:

1. Python is an interpreted language, meaning it doesn't require a compiler because it uses an interpreter to execute the code line by line.
2. **IDE (Integrated Development Environment):** A software application that provides comprehensive facilities to computer programmers for software development. An IDE usually includes a code editor, a debugger, and build automation tools.
    - **Why Use an IDE?**
        - **Code Editing:** Simplifies writing and editing code with features like syntax highlighting, code completion, and error detection.
        - **Debugging Tools:** Allows you to find and fix errors in your code more easily.
        - **Project Management:** Helps manage multiple files and directories in a project.
        - **Integrated Tools:** Often includes version control, testing frameworks, and more, all in one place.

How to Set Up Python:

1. **Installation:**
    - Go to python.org.
    - Download and install the latest version for your operating system.
    - Ensure you check the option to add Python to your PATH during installation. The PATH is an environment variable that stores a list of directories where your operating system looks for executable files. By adding Python to the PATH, you can run Python and its associated tools from any command-line interface (like Command Prompt or Terminal) without having to navigate to its installation folder manually.
      Here are the tutorial links to install and setup the PATH in your system.
      Windows 11 Mac OS
2. **Popular Python IDEs:**
    - **IDLE:** Comes with Python. Simple to use for beginners.
    - **VS Code:** Lightweight and powerful. Install from code.visualstudio.com, then add the Python extension.
    - **PyCharm:** Feature-rich IDE for Python. Download the Community Edition from jetbrains.com/pycharm.

- **Jupyter Notebook:** Great for data science and interactive coding. Install via pip install notebook and run with jupyter lab/notebook. See https://jupyter.org/.

Python Basic Programming Concepts:

1. **Variables:** Variables are used to store data in Python. You can assign a value to a variable using the "=" operator.

   Ex: x = 10
   Name = "Alice"

2. **Data Types:** In programming the data type refers to the classification of data items that tells the compiler or interpreter how the programmer intends to use the data. Data types determine the type of operation or methods that can be applied to the data and the way data is stored in memory.

   I. **Numeric Types**
   - **Integers ('int'):** Whole numbers, both positive and negative without decimal points.
     Ex: -5, 0, 42
   - **Floating-Point Numbers ('float'):** Numbers with decimal points.
     Ex: 3.14, -0.001, 2.0
   - **Complex:** Numbers with real and imaginary parts.
     Ex: 3 + 4j, -1 + 2.5j (used in fields like engineering and physics for complex calculations)

   II. **Sequence Types**
   - **Strings ('str'):** Sequences of characters enclosed in quotes
     Ex: '"Hello, World!"', ''Python''
   - **Lists ('list'):** Ordered collections of items. They can hold elements of different types.
     Ex: [1, 2, 3], ['apple', 'banana'], [1, 'text', 3.14]
   - **Tuples('tuple'):** Ordered and indexed collections of items that cannot be changed after they are created, which is helpful when a collection of items needs to be defined but not changed in use.
     Ex: (1, 2, 3), ('apple', 'banana'), (1, 'text', 3.14)

   III. **Set Types**
   - **Sets ('set'):** Mutable, Unordered collections of unique elements.
     Ex: {1, 2, 3}, {'apple', 'banana'}

   IV. **Mapping Types**
   - **Dictionaries ('dict'):** Unordered collections of key-value pair
     Ex: {'name': 'Alice', 'age': 25}, {1: 'one', 2: 'two'}

   V. **Boolean ('bool'):** Represents truth values (0 or 1).
     Ex: True -> 1, False -> 0

3. **Basic Operations**

   I. **Arithmetic Operations:** Python supports standard arithmetic operations like addition ('+'), subtraction ('-'), multiplication ('*'), division ('/'), and modulus ('%') – modulus is where the reminder after the division is the result.
     Ex. Result = 5 + 3 -> # result is 8

   II. **String Operations:** Strings can be concatenated using the '+' operator and repeated using the '*' operator.
     Ex: greetings = "Hello" + " " + "World!" -> # output is Hello World!

III. Control Structures

- **Conditional Statements:** Use ‘if’, ‘elif’, and ‘else’ to execute code based on conditions. You use elif when you want to add secondary, tertiary, etc. checks when conditions are not met, and else for the last item or when nothing is met.

Ex 1:
```
age = 20
if age >= 18:
  print("Adult")
```
In this example, since the condition age >= 18 is true, the program prints "Adult".

Ex 2:
```
age = 16
if age >= 18:
  print("Adult")
else:
  print("Minor")
```
Here, the condition age >= 18 is false, so the program executes the code under else and prints "Minor".

Ex 3:
```
age = 17
if age >= 65:
   print ("Senior Citizen")
elif age >= 18:
   print("Adult")
else:
   print("Minor")
```
In this case, the first condition age >= 65 is false, and the second condition age >= 18 is also false, so the program moves to the else block and prints "Minor".

- **Loops:** Use loops to repeat actions. Common loops are ‘for’ and ‘while’

Ex:
```
for i in range (5):
    print(i)
 # Prints 0 to 4
```

4. Functions

I. Defining Functions:
Functions are reusable blocks of code that perform a specific task. Define a function using the ‘def’ keyword.

Ex:
```
def greet(name):
    return "Hello, " + [name]
```
Function named greet is defined and will return “Hello, [name]” as name is defined elsewhere, such as in a table

II. Calling Functions:
Invoke a function by passing arguments to it.

Ex:
```
print(greet("Alice"))
  # Outputs "Hello, Alice"
```