**Python Basix**

**Starting off with Sha-Bang!!!!**

#!/bin/python3

**Invoking the Script**

* python3 <script name>
* ./ <script\_name>
* /home/venkat/<script name>

Create script folder & place all script there & use that path as global path variable. So you can invoke script anywhere.

Sudo vim /etc/profile

export SCRIPT\_HOME="/home/ec2-user/Shell\_programing/"

export PATH=$SCRIPT\_HOME:$PATH

**What is Python?**

Data in python is strongly typed [so attempting to add a number and string will give an error] but also dynamically typed, so you are freed from worrying abt variable declarations.

**Hello World**



print(01234)

^

SyntaxError: leading zeros in decimal integer literals are not permitted; use an 0o prefix for octal integers

**Literal**

* A value of some type.
* Example of numeric literals : 1, 45, 999
* Examples of string literals: “Hello” “Welcome

**Function**

* A named block of code that we can call by using its name
* We can write our own functions or we can use functions that are built into python [such as print]
* In Python, all functions return the value

**Argument**

* A value passed to a function in order to give its value to work with.
* There may be no arguments or one or more arguments
* Arguments appear in parentheses ()

**The Escape Character**

****

\n new line

\t tab

\ escape character

\\ escape single \

\b backspace

\r carriage return

**Identifiers**

* In Python any variable, function, class, module and object are identified using a name which is called as **identifier**. An identifier can start with uppercase or lowercase character or an underscore (\_) followed by any number of underscores, letters and digits. All identifiers in python are case sensitive.
* **Example:**weight=10
* In the above example weight is an identifier.

**Keywords**

* Keywords are the **reserved words** in python. So keywords cannot be used as an identifier and to name variables or functions. Few of the keywords are listed below.
* **Example:** if, else, elif, for, where, break, continue

**Variables and types**

****

* Store the value
* Variables are like containers for data (i.e. they hold the data) and the value of variable can vary throughout the program
* Case sensitive
* Rules for variable names
* Must begin with letter or underscore\_
* They can contain letters numbers or underscore
* Case sensitive

**Syntax:**var\_name = “literal\_value”

Computer\_parts = [“computer”, “monitor”, “keyboard”]

**Python is a strongly typed Languages**



* How to identify the data types stored in a variable?
* print(type(age), type(name));
* **Concatenation will not work on combinations of string and int**
* print(name + "is" + age1 + age)
* TypeError: can only concatenate str (not "int") to str

**Python Data Types**

Python has several built-in data types that can be classed as:

* Numeric
* Iterator
* Sequence [which are also iterator]
* Mapping
* File
* Class
* Exception

**Numeric Data types**

Python doesn’t has type declaration

* **Int**
* Python integer data type is called **int**
* Integers are whole number : 0 to 9999….
* There is no limit to the size of the values that you can store in a python int
* **Float**
* Fractional numbers are called float. 1.5, 5.55….
* Max float value on a 64 bit computer is 1.7976931…. which means move the decimal point 308 place right
* Smallest float: 307 zeros before the decimal
* Ppython floats have 52 digits of precision which should be adequate for most purposes. If you need precise decimal numbers, python 3 now includes **decimal data type**.
* Float value cannot be interpreted with int value it cause type error [for loop]
* **Complex** >>Data science [contains a real and imaginary part based on the square root of minus one]

**Numeric Operator**

****

a / b gives float value 10 / 5 = 2.0

a // b gives int value 10 / 5 = 2

**Expressions**

Expressions is anything that can be calculated to return a value**.**

**Operator precedence**

****

* BODMAS Bracket Order Div/Mul Add/Sub
* Add & sub as equal precedence
* Div & Mul has equal precedence
* Left to right

**String DataType**

****

**Indexing**

Printing the character using indexing

* Parrot = “”Green or Red”
* Print(Parrot[0]) >>>G
* Print(Parrot[6] >>>o
* A=5;B=2
* Print(Parrot[A-B]
* Print(Parrot[5-2])

**Negative Indexing**

* Print(Parrot[-1]) >>>>d
* Print(Parrot[3-5]

**Slicing**

Variable[ range: range]

Variable[0:10]

0:10 Upto 10 but not included

var = “123456789”

* Print(Parrot[0:5]) >>> 0 to 4 not 5 >>output 12345
* Print(Parrot[:] >>>> 123456789
* Print(Parrot[:5] >>>> 01234
* Print(Parrot[5:] >>>> 56789
* Print(Parrot[:4] + Parrot[4:])

**Negative Slicing**

* var=”012345789”
* Print(var[-3:55555] >> 789
* Print(var[-3:-1] >> 78
* Print(var[-3:2] >> nothing display bcz it cant go back
* Print(var[-3:9]) >> 78
* **Print(var[9:0]) >>>>null**

Print(var[-3]) >>>> out: 7

Print(var[-5:2] >>>> out: null only fprwards not backward

Print(var[-5:8] >>>>5678 from –5 upto 8the place but 8

**Step in a Slicing**

**Var=”0123456789”**

* Print(var[0:6:3]) >>> o to upto 6 but not 5 but only every 3rd digit from 0

012345 >>0:6

03 >>0:6:3

* Print(var[::]) >>> Acesding order 0123456789
* Print(var[::-1] >>> Reverse order 9876543210
* Print(var[9::-1]) >>9 >>>9876543210
* Print(var[9:0:-1] >>> 90 >> 987654321
* Print(var[9:0:1] >>> nill
* Print(var[9:2:-1] 9012 >>>9875643

**Negative step**

* var=”012345789”
* Print(var[9:0:-1] >>>> 9 till o but not zero & -1 so 987654321

**String / Sequence operator**

Python has 5 sequence types built in:

* The string type
* List
* Tuple
* Range
* Bytes and Bytearray

**What is a Sequence? [Character or word]**

* A sequence is defined as an ordered set of items.

For example: the string “Hello world” contains 11 item and each item is a character.

A list is also a sequence type.

Example: [“computer”, “monitor”, “Keyboard”, “mouse”, “mat”]

That list contains 5 item each of which is string.

This is called list if strings in other words, it’s a sequence where each item is also sequence.Because sequence is ordered , we can use indexes to access individual items in the sequence.

**Ex: Computer\_parts = [“computer”, “monitor”, “keyboard”]**

**Computer\_parts[1] >>>>>>>>> “monitor” >>output**

**Computer\_parts[1][0] >>>>>>>>>> “m” >>>output**

**Str sequence type**

****

**Str function used to concantanate strings with numeric value**

Everything you can do with str sequence type can also be done with other sequence types. Everything one exception. Not all sequence types can be concantenated or multiplied. Range is an example of a seq that cant be concantaned

* + String type
  + List
  + Tuple
  + Range
  + Bytes and bytearray

**Alternative way of str function is Replacement field . format(varaible) method**

**String Replacement Fields { }**

* Print(“My age is {0}”.format(20) 
* Print(“My age is {0}”.format(varaible)
* Print("There are {0} days in {1}, {2} and {3} " .format(5+5, "Jan", "Feb", "Mar" ))
* Print("There are {} days in {}, {} and {} " .format(5+5, "Jan", "Feb", "Mar" ))

It process in order if its empty

* Field value can be in any order
* When printing strings and numbers it would often be convienient to display both values using a single call to print. For example we want to print a description of what a value is, before the value itself.
* We have seen that numbers cant be concantanated with strings using +, as the presence of a number instruct python to attempt addition and that fails.

**String Replacement Field formating { 2:5}**

****

* {2:5} >>>filed width is 5>
* {2:<5} >>> left aligned with filed width 5
* {2:>4} >>> right aligned with field width 4
* {2:^4} >>> centre aligned with filed width 4

Precesion is higher priority than filed width

* Print(“Pi is approximately {0:12}”.format(22/7))

general precision 15 digit after decimal

{0:12f} 6 digit precsion

{0:12

**Another way is f string [works on 3.8 ]**

* print(f"My age is my {variable:pricision}"

**IN OPERATOR**

* #IN operator evalutes true if first thing exist on second  
  today = "friday"  
  print("day" in today) #true
* print("parr" in today) #fals