Basic Python

Variables (box)

In Python syntax:
variable = data value
box = "books"

Data Value

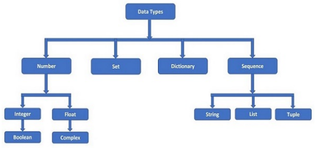
BASIC

Basic Manipulation & Analysis

Features
• Easy to Learn
• Dynamically Typed
• Interpreter Based
• Interactive
• Multi-paradigm
• Standard Library (NymPy, Pandas, Matplotlib, etc)
• Open Source and Cross Platform
• GUI Applications
• Database Connectivity
• Extensible
• Active Developer Community

Define Variable

Data Types	
String	• "Hello!", "23.34"
Integer	• 5364
Float	• 3.1415
Booleans	• True, False
List	• Collection of data sits between [ ]
Tuples	• Collection of data sits between ( )
Dictionary	• Collection of data sits between { }



Converts Value To Another Type

Type Casting	
str()	• Convert object x to string representation
int()	• Convert x to integer if x is string
floats()	• Convert x to floating-point number
chr()	• Convert integer to a character
lists()	• Convert string & tuple to a list
tuple()	• Convert string or list to a tuple

Make Code Readable

Comments	
Single-Line	• Starts with # and occupies a single line
Multi-Line	• Created using triple-quoted strings (''' or """)

Control Formating Output

Escape Characters	
\b	• Backspace
\n	• Newline
\s	• Space
\t	• Tab
\'	• Single quote
\"	• Double quote
\\	• Backlash

OPERATORS

Basic Mathematical

Arithmetic Operators	
+	• Addition x + y = 10, 3 + 7 = 10
-	• Subtraction x - y = 4, 3 - 7 = 4
*	• Multiplication x * y = 21, 3 * 7 = 21
**	• To the power of x**y = 2187, 3**7 = 2187
/	• Division x / y = 3.6666, 11 / 3 = 3.6666
//	• Floor Division (round down) x//y=4, 9//2=4
%	• Modulus (remainder) x % y = 2, 11 % 3 = 2

Compare & Return Boolean Result

Comparison Operators	
<	• Less #x < y
<=	• Less or equal #x <= y
>	• Greater #x > y
>=	• Greater or equal #x>= y
==	• Equal #x==y
!=	• Not equal #x!= y

Assign Value

Assignment Operators	
=	• Equal #x = 2
+=	• Add value of y to x #x = x + y
-=	• Subtract value of y from x #x = x - y
*=	• Multiply value of x by y #x = x * y
/=	• Divide value of x by y #x = x / y

Compare Binary Numbers

Bitwise Operators	
&	• AND #a & b
|	• OR #a | b
^	• XOR #a ^ b
~	• NOT #~a
<< or >>	• Zero fill left shift or Signed right shift

Check Object References

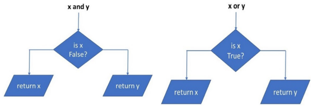
Identity Operators	
is	• Checks for identical object references
is not	• Check for different object references

Check If Item In Container (List & Tuples)

Membership Operators	
in	• Checks substring is present in bigger string
not in	• Checks for absence in sequences

Combine Conditions To Evaluate Result

Logical Operators	
and	• Returns **True** if both x & y are True
or	• Returns **True** if either x or y are True
not	• Reverses result, not **True** becomes **False**



FUNCTIONS & MODULES

Input & Output Operations

Input & Output Functions	
input()	• Reads input from the console
open()	• Opens file and returns file object
print()	• Prints text stream or console
format()	• Converts value to formatted representation
random()	• Generate a pseudo-random number

Common Math Operations

Math-Related Built-in Functions	
abs()	• Calculates absolute value of a number
divmod()	• Computes integer division results
max()	• Returns greatest value in sequence
min()	• Returns smallest value in sequence
pow()	• Raises a number to a power
round()	• Rounds a floating-point value
sum()	• Sums the values in an iterable

Common String Operations

String-Related Built-in Functions	
+	• Concatenate two strings
*	• Repeat string multiple times
upper()	• Converts all letters to uppercase
lower()	• Converts all letters to lowercase
replace()	• Replaces substrings with new values
count()	• Counts substring occurrences in string
join()	• Join list into single string
startswith()	• Check if string begins with given substring
endswith()	• Check if string ends with given substring
split()	• Split strings into lists of substrings
strip()	• Trims leading/trailing characters
title()	• Capitalizes the first letter of each word

Boolean Expression

Boolean-Related Built-in Functions	
bool()	• Evaluate value & give **True** or **False** result

Processing Iterables and Iterators

Iterate-Related Built-in Functions	
len()	• Calculates length of sized object
reversed()	• Creates a reversed iterator
sorted()	• Creates sorted list from an iterable
all()	• Verifies all iterable elements are true
any()	• Verifies any iterable elements are true
range()	• Generates range of integer values
slice()	• Creates a slice object
next()	• Retrieves next item from an iterator
filter()	• Filters elements from an iterable

Code Specific Task With File Extension .py

Use Built-in Modules (keyword import)	
math	• Provides math functions & constants
datetime	• Provides date & time manipulation classes
random	• Allows generation of random numbers
re	• Supports expressions for pattern matching
collection	• Provides additional data structures

DATA STRUCTURES

Sequence Data

List	Ordered & Mutable
[ ]	• Store different data types in sequence
[0]	• Index starts from 0-based
[-1]	• Reverse sequence of list, starts from -1
[start:end]	• Include Start index, but exclude End index
[:6]	• Omit Start index, but exclude End index
[6:]	• Include Start index & omit End index
append()	• Add single element to end of list
extend()	• Adds iterable elements to end of list
insert()	• Insert an element to the list
remove()	• Removes item from the list
pop()	• Removes element at the given index
del(List[x])	• delete element by referring its index number

Sequence & Constant Data Closely Related

Tuple	Ordered & Immutable
( )	• Items need not be of same data type
[0]	• Index starts from 0-based
[-1]	• Reverse sequence of list, starts from -1
[start:end]	• Include Start index, but exclude End index

length = 5

	"apples"	"bananas"	"cherries"	"dragonfruits"	"elderberry"
index	0	1	2	3	5
negative index	-5	-4	-3	-2	-1

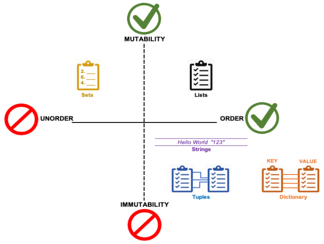
Collecting Unique Elements

Set	Unordered & Mutable
{ } or set()	• Defining & initializing unique elements
add()	• Include new elements into an existing set
pop()	• Removes element at the given index
remove()	• Eliminate specific elements from the set
discard()	• Remove element from set if it is present

Store Key-Value Pair Elements

Dictionary	Ordered & Immutable
{u:v, x:y,}	• Each key is unique & maps to value
[ ] or get()	• Access value associated with a specific key
[ ] = x	• Update elements with key-value assignment

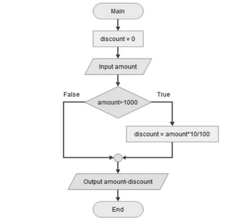
del variable
pop()
popitem()
	• Removes items using del statement and pop() or popitem() method



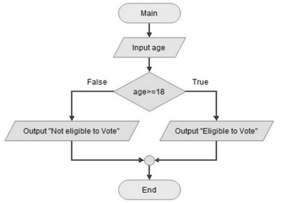
CONTROL STATEMENTS (ability to take arguments and perform actions)

Decision Making

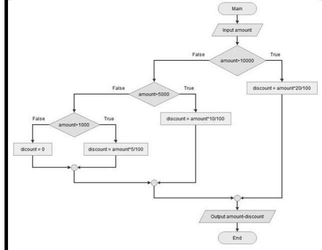
If	Execute based on condition
if condition:	• If **True** execute action if **False** skip action



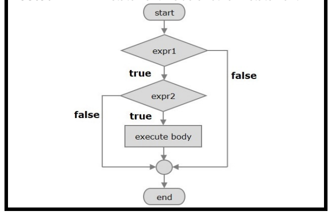
if:-else: • If **True** execute action if **False** execute another action



if:-elif:-else: • Checks multiple expressions for **True**

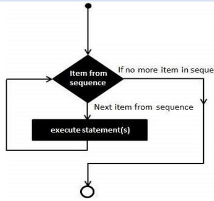


Nested if • If statement inside another if statement

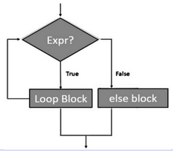


Iterates Over Sequence

For Loop	Iterating over known sequences
for x in y:	• Iterates over elements of sequence in order



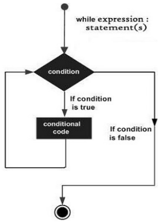
for: - else: • Specifies code to execute after completion



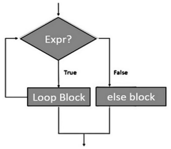
for + break	• Exit loop entirely #if x == "condition"
for+continue	• Stop current iteration & continue next
range()	• Loop elements specified times, starting at 0

Execute Based On Conditions

While Loop	Repeat actions until condition is met
while x < y:	• Execute statements if condition is true



x += 1	• Increase value of x with each iteration
while x == 1	• Continues to run indefinitely if **True**
while: - else:	• Specifies code to execute after completion



while + break	• Exit loop entirely #if x == "condition"
while+continue	• Stop current iteration & continue next
pass	• Placeholder statement