Introduction on

Python Fundamental Programming I

Kamal

AptComputingAcademy

Contents

Numbers and Operators
Comments
Variables of different types
Input and output
Loops
Conditions
Functions
Variable scopes

Built-in data types:

- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Numbers and Operators**
- Comments
- Variables of different types
- Input and output
- Loops
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Representing Numbers:**
  - Decimal, Hexadecimal, Octal, Binary
- **Operators:**
  - Arithmetic : +, -, *, /, %, **, // (divide as integer explicitly)
  - Binary : |, &, ^, ~, <<, >>
  - Logical : ==, !=, <, <=, >, >=, <>, and, or, not
  - Assignment : =, +=, -=, *=, /=, %=, **=, //=

- **Numbers and Operators**
- **Comments**
- Variables of different types
- Input and output
- Loops
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - Dictionaries
  - Sets

- **Comments:**
  - **#** ---- **Comment a line**
  - **“”** ---- **Comment few lines (Special purpose)**

- **Numbers and Operators**
- Comments
- **Variables of different types**
- Input and output
- Loops
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **List of basic data Types:**
  - Integer, Floating point, Complex , None, Boolean
  - Callable data types:
    - int(), float(), complex(), bool(), None is not callable
  - Assigning the variables
  - Operations

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- Loops
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Input:**
  - Using “input()”
    - Str1 = input(“Comment”)
    - Str2 = input(“Please enter your option”)

- **Output:**
  - Using “print()”

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **While**
  - **Syntax:**

```
while <Condition>:
---->Statement1
---->Statement2
---->Statement3
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **For**
  - **Syntax:**
    - for <Var> in <List>:

```
for <item> in (list/dict/tuple/range/set):
---->Statement1
---->Statement2
---->Statement3
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

Loops with "pass"

Syntax:

```
for <item> in (list/dict/tuple/range/set):
---->Statement1
---->Statement2
---->if <Condition>:
---->---->pass
---->else:
---->---->Startement3
---->Statement4
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

Loops with "continue"

Syntax:

```
for <item> in (list/dict/tuple/range/set):
---->Statement1
---->Statement2
---->if <Condition>:
---->---->continue
---->else:
---->---->Startement3
---->Statement4
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

Loops with "break"

Syntax:

```
for  <item> in  (list/dict/tuple/range/set):
---->Statement1
---->Statement2
---->if <Condition>:
---->---->break
---->else:
---->---->Startement3
---->Statement4
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- **Conditions**
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

If Condition:

Syntax:

```
if <Condition> :
---->Statement1
---->Statement2
---->Statement3
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- **Conditions**
- Functions
- Variable scopes

- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

If Condition:

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- **Conditions**
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

If, else Condition:

Syntax:

```
if <Condition>:
---->Statement1
---->Statement2
---->Statement3
else:
---->Statement1
---->Statement2
---->Statement3
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- **Conditions**
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

If, elseif, else Condition:

Syntax:

```
if <Condition> :
---->Statement1
---->Statement2
---->Statement3
elif <Condition>:
---->Statement1
---->Statement2
---->Statement3
else:
---->Statement1
---->Statement2
---->Statement3
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- **Conditions**
- Functions
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

Nested If, else Condition:

Syntax:

```
if <Condition>:
--->Statement1
--->Statement2
--->Statement3
--->if <Condition>:
--->--->Statement1
--->--->Statement2
--->Statement1
--->Statement2
--->Statement3
else:
--->Statement1
--->Statement2
--->Statement3
--->if <Condition>:
--->--->Statement1
--->--->Statement2
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- **Functions**
- Variable scopes
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

Functions

Syntax:

```
def function_name(arg1, arg2, arg=3):
---->Statement1
---->Statement2
---->Statement3
---->return Val
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- **Variable scopes**
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

Functions

Syntax:

```
a = 10
def function_name(arg1, arg2, arg=3):
---->global a
---->Statement1
---->Statement2
---->Statement3
---->return Val
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- **Variable scopes**
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Functions**
  - **Special cases dealing with arguments:**
    - 1) Formal positioning
    - 2) *args
    - 3) Keyword argument
    - 4) **keyword

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- **Variable scopes**
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Functions**
  - **Special cases dealing with arguments:**
    - 1) Formal positioning
    - 2) *args
    - 3) Keyword argument
    - 4) **keyword

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- **Variable scopes**
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Functions**
  - **Special cases dealing with arguments:**
    - 1) Formal positioning
    - 2) *args
    - 3) Keyword argument
    - 4) **keyword

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- **Variable scopes**
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Functions**
  - **Special cases dealing with arguments:**
    - 1) Formal positioning
    - 2) *args
    - 3) Keyword argument
    - 4) **keyword

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- **Variable scopes**
- **Built-in data types:**
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

- **Functions**
  - **Special cases dealing with arguments:**
    - 1) Formal positioning
    - 2) *args
    - 3) Keyword argument
    - 4) **keyword

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - **Strings**
  - Lists
  - Tuples
  - Dictionaries
  - Sets

- **Strings:** (Immutable sequence of chars)
  - **Defining Strings:**

```
String = "Python"
String = 'Python'
String = '''Python
Is a interpreted language'''
String = "Hello"*10
String = "Value of a = %d"%(a)
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - **Strings**
  - Lists
  - Tuples
  - Dictionaries
  - Sets

- **Strings: (Immutable sequence of chars)**
  - **Printing Strings:**
    - (String[0], String[-1]) ---------- Prints only a char
    - (String[0:3]) ---------- Prints chars from index 0 to 2
    - (String[0:10:2] --------- Prints chars from 0, 2,4,6,8
      - [init : final : inc]
      - **Default values:**
        - init = 0
        - final = length of string/list/tuple/set
        - inc = 1 (Case: If inc is -ve, default values of init = -1, final = ~~end~~1
    - String[::], String[::1], String[::-1], String[-3:-6:-1]

- Numbers and Operators
- Comments
- Variables of different types
- Input and output
- Loops
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - **Strings**
  - Lists
  - Tuples
  - Dictionaries
  - Sets

- **Strings:** (Immutable sequence of chars)
  - **Operations:**
    - **Finding**:
      - find(), rfind(),index(),eindex()
    - **Adjusting**:
      - ljust(), rjust(),center()
    - **Manipulating**:
      - lower(), upper(), capatitalize(), title()
      - strip(), replace(), split()
    - **Checking**:
      - islower(), isupper(), isdigit(), isspace(), isascii()

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - **Lists**
  - Tuples
  - Dictionaries
  - Sets

- **Lists:** (Mutable sequence of python objects)
  - **Defining:**

- List = [1]
- List = [1,2,3,4]
- List = [1,'string', 10.2, True, None, [1,2]]
- List = [[],[],[]] -- Multi demential, can be empty
- List = [i**2 for i in range(1,10) if x%2 == 0]
- List = list(range(1,10))

Note: Can convert tuple --- list, list --- tuple

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - **Lists**
  - Tuples
  - Dictionaries
  - Sets

- **Lists:** (Mutable sequence of any data type)
  - **Accessing:**

    - List[1]
    - List[0:2]
    - List[0][1]
    - List[::]
    - List[0,10,2]
    - List[-1:-4:-2]
    - Loop:

```
for i in List:
    ---------------
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - **Lists**
  - Tuples
  - Dictionaries
  - Sets

- **Lists: (Mutable sequence of any data type)**
  - **Operations:**

- Manipulating:
  - sort(), insert(), pop(),clear(), append(), extend()
  - remove(), reverse()
- Other:
  - copy(), count(),index()
- With functions:

```
def Fun():
    return [1,2,"String", True, None]
result = Fun()  ----------> result is a list
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - **Tuples**
  - Dictionaries
  - Sets

- **Tuples: (Immutable sequence of any data type)**
  - **Defining:**

- Tup = (1,"String",1.2)
- Tup = 1,"String",1.2
- Tup = ()
- Tup = (10)-----> It is integer, notice in **list**
- Tup = (10,) -----> It is tuple
- Tup = (("String",1,2),(1,2,3),(1.2,))
- Tup = (("String",1,2),[1,2,3,("String",1.2)],(1.2,))

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - **Tuples**
  - Dictionaries
  - Sets

- **Tuples:** (Immutable sequence of any data type)
  - **Accessing:**

- Same as Lists..... :) :)

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - **Tuples**
  - Dictionaries
  - Sets

- **Tuples:** (Immutable sequence of any data type)
  - **Operations:**

- count()
- Index()
- **Other**:
  - Tup1 = Tup2,Tup3 ----> Creating new with two tuples...
  - del Tup1 ----> Deleting
  - 1 in (1,2,3) ----> True/False
  - return 1,2,3 ----> Function returning tuple

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - **Dictionaries**
  - Sets

- **Dictionaries: (Mutable sequence of key and value)**
  - **Defining:**

- Dict **= {Key1:Value1, Key2:Val2}**
- **Rules:**
  - **Key:**
    - Better to be unique.
    - Should be immutable (int,float,boolean, None string,tuple)
  - **Value:**
    - Can be any type, cab be mutable/immutable

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - **Dictionaries**
  - Sets

- **Dictionaries:** (Mutable sequence of key and value)
  - **Accessing:**

- **Dict**[Key1]
- **For loop:**

```
for Key,Val in Dict.items():
    print(Key,Val)
```

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - **Dictionaries**
  - Sets

- **Dictionaries:** (Mutable sequence of key and value)
  - **Modifying:**

- **Updating:**
  - Dict[“Name”]= “Rahe”
- **Deleting:**
  - del Dict[“Name”]
  - del Dict

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - **Dictionaries**
  - Sets

- **Dictionaries:** (Mutable sequence of key and value)
  - **Operations:**

- Dict.clear()
- Dict.copy()
- Dict.get(key)
- Dict.items()
- Dict.keys()
- Dict.values()
- Dict.setdefault(key,"Default Value")

Dict.update(Dict1)
Pop()
Popitem()
fromkeys()

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - Dictionaries
  - **Sets**

- **Sets: (Ordered collection of unique elements(immutable))**
  - **Defining:**

  - Set1 = {1,2,3,4,1,2,8,9,0,2}
  - Set1 = {1,2,"String",0.1223, None, True}
  - Set1 = set([1,2,3,4,"String"])
  - Set1 = {"This world is beautiful"}
  - Set1 = set("This world is beautiful")

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - **Dictionaries**
  - **Sets**

- **Sets: (Unordered collection of unique elements(immutable))**
  - **Accessing:**

- for i in set1:

  --------Statement------

- Set1[0] ---------- Does not support indexing

- **Special operations:**
  - 1 in Set1
  - 1 not in Set1

- **Numbers and Operators**
- Comments
- **Variables of different types**
- **Input and output**
- **Loops**
- Conditions
- Functions
- Variable scopes
- **Built-in data types:**
  - Strings
  - Lists
  - Tuples
  - **Dictionaries**
  - **Sets**

- **Sets: (Unordered collection of unique elements (immutable))**
  - **Operations:**

- Intersection() --------- set1 & set2
- Union() --------- set1 | set2
- Difference() --------- set1 – set2
- Differenceupdate(), Intersectionupdate(), synmmetricupdate()
- Update()
- Add(), Clear(), Pop(), discard()
- Copy()
- Issubset(), Issuperset()

Thank You.....