Python Basics

August 20, 2025

Print

[2]:
```
# Print
a="HelloWorld"
print(a)
```

```
HelloWorld
```

Naming Variables - Camel Case - myVariableName - Pascal Case - MyVariableName - Snake Case - my_variable_name

Data types: These are some of the most common data types you'll run into. - Text Type: `str` - Numeric Types: `int`, `float` - Sequence Types: `list`, `tuple` - Mapping Type: `dict` - Set Types: `set` - Boolean Type: `bool`

[3]:
```
#  Find Type
type(a)
```

[3]:
```
str
```

Booleans and Logical Operators

[ ]:
```
# Booleans and Logical Operators
t, f = True, False
print(type(t))
print(type(f))
print(t and f) # Logical AND; - (a,b) both true -> true
print(t or f)  # Logical OR; - (a,b) one true -> true
print(not t)   # Logical NOT; - inverts what t is -> t was true, so not t =␣
 ↪false
print(t != f)  # Logical XOR;  -> a doesn't equal b, so returns true
```

Lists: - Lists are used to store multiple items in a single variable - List items are ordered, changeable, and allow duplicate values. - List items are indexed, the first item has index [0], the second item has index [1] etc. - List items can be of any data type. - Lists can contain different data types.

[ ]:
```
# Create a List
mylist = []
mylist = [10,20,30]
mylist[0] # print 10
```

```
len(mylist) # find the length of mylist
max(mylist)
min(mylist)

mylist.append(50)
del mylist[4]  # delete the item with index 4 (5th item) -- base on index
mylist.remove(50)  #delete the item with value 50 -- base on value
mylist[0:2]  # mylist[index1 : index2]  give a new list which from index1 to␣
 ↪index2, index2 not inluce
```

Dictionaries - Dictionaries are used to store data values in key:value pairs. - Dictionary items are ordered, mutable, and do not allow duplicate keys. - Keys are unique and immutable types (e.g., string, number, tuple), while values can be of any type. - Dictionary items are accessed via keys, not by numeric index. - A dictionary can contain values of different data types.

[ ]:
```
# Create a Dictionary in 'key' : 'value' pair
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

car['year']  # dic['key'] return value of this key
car.keys()  # dic.keys() return all keys in a list
car.values()  # dic.values() return all values in a list
car['year'] = 2022  #  change value, update new value
car["Trim"] = "raptor" #  add a new element
```

If- Else Statment

[ ]:
```
#  If- Else
a = 2
if a == 2:
    print("Hello.") # first line
else:
    print("Goodbye.") # second line
```

Iteration, For Loop and While Loop

[ ]:
```
# iteration
nums = [10, 20, 30]
it = iter(nums)
print(next(it))  # 10
print(next(it))  # 20
print(next(it))  # 30

# for loop
for i in range(5): # for i in range(len(mylist))
    print(i)
```

```
    # i += 1

for n in nums:
    print(n)

# while loop
x = 0
while x < 3:
    print(nums[x])
    x += 1
```

Create a new Function

[ ]:
```
# We want this function to print every string in a list.
# 'list_string' is the function name. 'list1' is the parameter name (this is␣
 ↪just a placeholder name). - what the function will take in
# You can have more than one parameter for your function.

def list_string(list1):
    temp = []
    for i in list1:
        if type(i) == str:
            print(i)
            temp.append(i)
    return temp
```

NumPy NumPy is a Python library used for working with arrays. - It also has functions for working in domain of linear algebra, fourier transform, and matrices. - In Python we have lists that serve the purpose of arrays, but they are slow to process. - NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

[ ]:
```
import numpy as np  # create a numpy array
arr = np.array([1, 2, 3, 4, 5])
print(type(arr))  #  return <class 'numpy.ndarray'>
print(arr.dtype)  #  return int64,  get datatype of an array - the value the␣
 ↪array is composed of
np.sort(arr)  # sort the array
np.mean(arr)  # find the mean of arrays
np.arange(0,4)  # create an array of values 0 through 4 np.arange(start, stop,␣
 ↪step)
np.sqrt(arr)

arr1 = arr * 10  # return [10,20,30,40,50]
lst = [1, 2, 4, 4, 7]
print(lst * 3)   # return [1, 2, 4, 4, 7,1, 2, 4, 4, 7,1, 2, 4, 4, 7]

b = np.copy(a)
np.empty(3)   # [0. 0. 0.] or random values
```

```
np.zeros((2,2))   # [[0. 0.],[0. 0]]
arr = np.array([1, 2, 3], dtype=float)   # array([1., 2., 3.])
arr.astype(int)   # [1 2 3] Convert array to another data type.

np.array_split(arr, n)  # Split array into n sub-arrays.
np.concatenate(([1,2],[3,4]))  #  Join arrays together [1 2 3 4]
np.repeat([1,2], 3)  # np.repeat(arr, n)Repeat each element n times.
np.transpose([[1,2,3],[4,5,6]])  # [[1 4],[2 5],[3 6]]
np.all([1,2,3])   # Check if all elements are True, return Ture
np.any([0,0,5])   # Check if any elements are True, return Ture here
np.nonzero([0,1,2]) # return the index(indices) of nonzero element, return␣
 ↪[1,2] here
```

np.argmax(arr) → index of max np.argmin(arr) → index of min np.argsort(arr) → indices that would sort array np.max(arr) → max value np.min(arr) → min value np.sort(arr) → sorted array

Operations np.choose(indices, choices) - Construct array from index list & choice arrays. ndarray.fill(val) - Fill entire array with a value. np.prod(arr) - Product of elements.

Basic Statistics np.cov(m) → covariance matrix np.mean(arr) → mean np.std(arr) → standard deviation np.var(arr) → variance

Basic Linear Algebra np.cross(a,b) → cross product (3D vectors) np.dot(a,b) → dot product np.outer(a,b) → outer product