Data Handling

(Data types and operations)

Built-in core data types in Python

1. Numbers (int, boolean, float, complex)
2. String
3. List
4. Tuple
5. Dictionary
6. sets

1. Numbers

Data type	Range
Integers	An unlimited range, subject to available memory
Booleans	Two values True (1) and False (0)
Floating point numbers	An unlimited range, subject to available memory
Complex numbers	Same as floating point numbers as real & imaginary parts are represented as floats

Complex numbers:

Imaginary number is represented using j (J), in place of traditional i

```
a = 1.4 + 2j


b = ( 2 + 1.98j ) + ( -4 – 3.57j )
print(b)
print(b.real)
print(b.imag)
```

Complex numbers: Example

```
>>> C1  = 1.2  +  3j

>>> C2  =  3  +  2.4j

>>> C3 = C1 + C2

>>> print( C3 )
( 4.2 + 5.4j )
```

Importing Modules

- Modules are imported using **import** statement

>>> **import math**

- This will import everything from the module named math that can be displayed using:

>>> **dir(math)**

dir() is a powerful inbuilt function in Python3, which returns list of the attributes and methods of any object (say functions , modules, strings, lists, dictionaries etc.)

Importing Modules

- E.g.

```
>>> import math
>>> math.pi
3.141592653589793
```

Importing Modules

- Other form of import statement

We can directly refer the const and functions available in math

e.g.

```
>>>  from math import sin, pi
>>>  print('Pi = ', pi)
>>>  print('sin(0) =', sin(0))
```

Math Module

1. ceil() :- This function returns the **smallest integral value greater than the number**. If number is already integer, same number is returned.

2. floor() :- This function returns the **greatest integral value smaller than the number**. If number is already integer, same number is returned.

Math Module

- e.g. :The floor and ceiling functions give us the **nearest integer** up or down

```
>>> print ("The ceil of 2.3 is : ", end="")
>>> print (math.ceil(a))
```

- o/p: 3

```
>>>  print ("The floor of 2.3 is : ", end="")
>>>  print (math.floor(a))
```

- o/p: 2

Math Module

3. fabs() :- This function returns the **absolute value** of the number.

4. factorial() :- This function returns the **factorial** of the number. An error message is displayed if number is not integral.

Math Module

>>> print ("The absolute value of -10 is : ")

>>> print (**math.fabs**(a))

- o/p: 10

>>> print ("The factorial of 5 is : ", end="")

>>> print (**math.factorial**(b))

- o/p : 120

Math Module

5. copysign(a, b) :- This function returns the number with the **value of ‘a’ but with the sign of ‘b’**. The returned value is float type.

6. gcd() :- This function is used to compute the **greatest common divisor** of 2 numbers mentioned in its arguments. This function **works in python 3.5 and above.**

Math Module

```
>>>  print ("copysigned value of -10 and 5.5 is : ")
>>>  print (math.copysign(5.5, -10))
```

- o/p : -5.5

```
>>>  print ("The gcd of 5 and 15 is : ", end="")
>>>  print (math.gcd(5,15))
```

- o/p: 5

Math Module

- print (math.exp(4)) # e raised to the power a
- print (math.log(2,3)) # logarithmic value of a with base b
- print (math.log10(10000)) # log a with base 10
- print (math.pow(3,2))
- print (math.sqrt(25))

Math Module

- **Trigonometric and Angular Functions:**

```
>>> print (math.sin(a))
>>> print (math.tan(a))
```

- Value passed in this function should be in **radians**.

```
>>>  print ("conversion from radians to degrees :")
>>>  print (math.degrees(a))

>>>  print ("conversion from degrees to radians : ")
>>>  print (math.radians(b))
```

Random Module

- The Random **module** contains some very useful functions
- E.g.

>>> import random

>>> random.random() #generates no. between 0 to 1
0.5634968614690888

>>> random.randint(0, 5) #generates int no. between 0 to 5

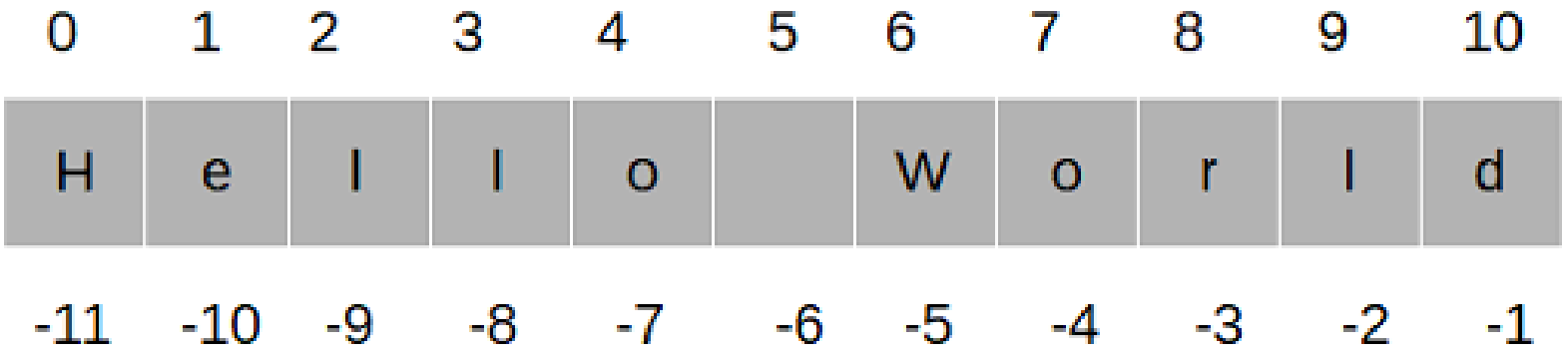
>>> random.randrange(0, 101, 5) #generates int no. between 0 to 101 with a step size of 5

2. String

- Strings are stored as sequence of characters:

  0 to len-1 : forward direction

  -len to -1 : backward direction



String functions

- Python has a set of built-in methods that you can use on strings.

- **len()** function returns the length of a string:

```
a = "Hello, World!"
print(len(a))
```

- **Concatenation**

```
c = a + b
c = a + “   “ + b
```

Slicing

- You can return a range of characters by using the slice syntax.

```
print(a[1])

b = "Hello, World!"
print(b[2:5])                    #display 2 to 4 chars

b = "Hello, World!"
print(b[-5:-2])                  #display -5 to -3 chars
```

Slicing

```
String1 = "Hello World"
String2 = "Python is Fun!"
print(String1[0])
print(String1[0:5])
print(String1[:5])
print(String1[6:])
String3 = String1[:6] + String2[:6]
print(String3)
print(String2[:7]*3)
```

OUTPUT:

```
H
Hello
Hello
World

Hello Python
Python Python Pytho
```

String functions

- **strip()** removes any whitespace from the beginning or the end:

```
a = "  Hello, World!  "
print(a.strip())          (returns "Hello, World!”)
```

- **lower()** returns the string in lower case:

```
print(a.lower())
```

- **upper()** returns the string in upper case:

```
print(a.upper())
```

String functions

- **replace()** replaces a string with another string:

  print(a.replace("H", "J"))

- **in or not in** to check if a certain phrase or character is present in a string.

  x = “wor" in a

String functions

- **split()** splits the string into substrings if it finds instances of the separator:

  print(a.split(",")) (returns ['Hello', ' World!'])

- **# taking two inputs at a time separated by space**

```
x, y = input("Enter a two value: ").split()
```

- **# taking two inputs at a time separated by comma**

```
x, y = input("Enter a two value: ").split(",")
```

String functions

```
MyString = " Hello World "
print(MyString.upper())
print(MyString.strip())
print(MyString.center(21, "*"))
print(MyString.strip().center(21, "*"))
print(MyString.isdigit())
print(max(MyString))
print(MyString.split())
print(MyString.split()[0])
```

OUTPUT:

```
HELLO WORLD
Hello World
**** Hello World ****
*****Hello World****
False
r
['Hello', 'World']
Hello
```

String functions

```
s1= "The apple is red and the berry is blue!"
print(s1.find("is"))

print(s1.count("is"))

print(s1.startswith("The"))

print(s1.endswith("The"))

print(s1.replace("apple", "car").replace("berry", "truck"))
```

OUTPUT:

```
10

2

True

False

The car is red and the truck is blue!
```

format() method

It takes unlimited number of arguments of any data types, and are placed into the respective placeholders

e.g.

```
str = "This article is written in {}"

print (str.format("Python"))
```

Example

e.g.

```
quantity = 3
itemno = 567
price = 49.95

myorder = "I want {} pieces of item {} for {} dollars."

print(myorder.format(quantity, itemno, price))
```

format(): example

```
>>> mystring = "value {0} equals {1}: {message}"

>>> mystring.format('x','23', message = 'ok')



'value x equals 23: ok'
```

format(): example

```
a, b = input("Enter two values: ").split()

print("First number is {} and second number is {}".format(a, b))
```

- Type conversion:

```
print("The {0} of 100 is {1:b}".format("binary", 100))
```

Output:

The binary of 100 is 1100100

- Type conversion:

```
>>> "Hello {0}, your balance is {1 : 9.3f } "
.format("John",55000.5898)

'Hello John, your balance is 55000.590'
```

Number formatting

```
print("{:5d}".format(20))
- - -20                          # - represents whitespaces

print("{:2d}".format(5000))
5000

print("{:8.2f}".format(29.3465))
- - - 29.34

print("{:05d}".format(20))
00020

print("{:08.2f}".format(29.3465))
00029.34
```

Number formatting

```
# float numbers with center alignment
print("{:^10.3f}".format(12.2346))


# integer left alignment
print("{:<5d}".format(12))
```

Example

1. Write a python program to check the given string is palindrome or not

```
s1=input("enter string")
s2=s1 [ : : -1 ]                    # Reverse string

if s1==s2:
    print("palindrome")
else:
    print("Not palindrome ")
```