# data types

May 13, 2025

### 0.0.1 Literals

1) String literals ”or” ”or”””” ”””” eg. ”abc”,”123” 2) Numeric literals eg.123,8.9 3) Boolean literals eg. true or false 4) Collection eg.list,tuple,sets,range 5) Special eg.dict

[ ]:

**There are 7 Data types** 1)int,float,complex eg. int=1,float=1.0,complex=2+j6 2)string-str eg.a=”adbg” ”123” 3)list eg. l=[1,2,3] 4)tuple eg. t=(1,2,3) 5)set eg.s=1,2,3 6)dict eg.d=key:value 7)bool eg. true or false

[ ]:
```
#int=123
```

### 0.0.2 To checking the data types

[ ]:
```
#type(), print()
```

[2]:
```
x=45
print(type(x))
```

```
<class 'int'>
```

[14]:
```
y=2.0
print(type(y))
```

```
<class 'float'>
```

[3]:
```
name="nikita"
print(type(name))
```

```
<class 'str'>
```

[5]:
```
x=True
print(type(x))
```

```
<class 'bool'>
```

### 0.0.3 Operators

operators use in python

1) + for addition 2) - for substraction 3) * for multiplication 4) / for division 5) 6) // for floor division 7) ** for power

[1]:
```
15/3
```

[1]:
```
5.0
```

[3]:
```
15%3
```

[3]:
```
0
```

### 0.0.4 1) + for addition

[9]:
```
a=1
b=3
#addition
c=a+b
print("addition is this:",c)
```

```
addition is this: 4
```

### 0.0.5 2) - for substraction

[4]:
```
c=a-b
print("substraction is this:",c)
```

```
substraction is this: -2
```

### 0.0.6 3) * for multiplication

[3]:
```
c=a*b
print("multiplication is this:",c)
```

```
multiplication is this: 3
```

### 0.0.7 4) / for division

[4]:
```
c=a/b
print("division is this:",c)
```

```
division is this: 0.3333333333333333
```

### 0.0.8 5) % for modulus

[5]:
```
a=6
b=4
c=a%b
print("modulus is this:",c)
```

```
modulus is this: 2
```

#### 0.0.9 6) Floor Division

[11]:
```
c=a//b
c
```

[11]: 0

#### 0.0.10 7) Power

[14]:
```
c=a**b
c
```

[14]: 1

**Operators : solve the exercise in script mode and jupyter also**

# 1 Getting input from the user

[3]:
```
x=56
b=90
c=x+b
c
```

[3]: 146

[5]:
```
a=int(input("enter the 1st no."))
b=int(input("enter the 2nd no."))
c=a+b
print(c)
```

```
enter the 1st no. 2
enter the 2nd no. 2
```

```
4
```

[7]:
```
name=str(input("Enter ur name: "))
surname=str(input("Enter ur surname: "))
print(name+" "+surname) #concatenation
print(name+surname)
```

```
Enter ur name:  yash
Enter ur surname:  ghate
```

```
yash ghate
yashghate
```

## 1.1 Conversion from one data type to another

### 1.1.1 Type casting

[19]:
```
x=123  #int
y=52.5  #float
c="chetana" #str: literals: char
b="123"    #str: literals: numeric
```

[20]:
```
p=float(x)
p
```

[20]: 123.0

[21]:
```
q=int(y)
q
```

[21]: 52

[22]:
```
r=float(c)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[22], line 1
----> 1 r=float(c)

ValueError: could not convert string to float: 'chetana'
```

[23]:
```
s=int(b)
s
```

[23]: 123

[24]:
```
t=float(b)
t
```

[24]: 123.0

# 2 Exercise

**find area of rectangle**

[26]:
```
#area=l*b
length=12
breadth=2.3
area=length*breadth
print("area of rectange is this:",area)
```

```
area of rectange is this: 27.599999999999998
```

[11]:
```
#area=l*b
length=int(input("enter length:"))
breadth=int(input("enter breadth:"))
area=length*breadth
print("area of rectange is this:",area)
```

```
enter length: 5.6
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[11], line 2
      1 #area=l*b
----> 2 length=int(input("enter length:"))
      3 breadth=int(input("enter breadth:"))
      4 area=length*breadth

ValueError: invalid literal for int() with base 10: '5.6'
```

[15]:
```
#area=l*b
length=float(input("enter length:"))
breadth=float(input("enter breadth:"))
area=length*breadth
print("area of rectange is this:",area)
```

```
enter length: 7
enter breadth: 7
```

```
area of rectange is this: 49.0
```

**find area of circle**

[ ]:
```
#area=pi*r*r or pi*r**2
```

[31]:
```
radius=2
pi=3.14
area=pi*radius*radius
area
```

[31]:
```
12.56
```

[17]:
```
radius=2
pi=3.14
area=pi*radius**2
area
```

[17]:
```
12.56
```

### 2.0.1 perimeter of circle

[1]:
```
import math
print(dir(math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb',
'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2',
'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd',
'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm',
'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter',
'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt',
'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

[5]:
```
math.pi
```

[5]:
```
3.141592653589793
```

[8]:
```
#2*pi*r
import math
radius=2.3
perimeter_of_circle=2*math.pi*radius
print("perimeter of circle is:",perimeter_of_circle)
```

```
perimeter of circle is: 14.451326206513047
```

[19]:
```
import math
x=int(input("enter the number"))
y=math.sqrt(x)
print(y)
```

```
enter the number 64
```

```
8.0
```

[36]:
```
radius=2

area=math.pi*radius*radius
area
```

[36]:
```
12.566370614359172
```

## 2.1 datetime module

### 2.1.1 package/library==>modules==>functions

[1]:
```
import statistics
print(dir(statistics.math))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb',
```

```
'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2',
'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd',
'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm',
'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter',
'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt',
'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

[23]:
```
import datetime  #package
print(dir(datetime))
```

```
['MAXYEAR', 'MINYEAR', 'UTC', '__all__', '__builtins__', '__cached__',
'__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__',
'date', 'datetime', 'datetime_CAPI', 'time', 'timedelta', 'timezone', 'tzinfo']
```

[11]:
```
print(dir(datetime.datetime))
```

```
['__add__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getstate__', '__gt__',
'__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__ne__',
'__new__', '__radd__', '__reduce__', '__reduce_ex__', '__repr__', '__rsub__',
'__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__',
'astimezone', 'combine', 'ctime', 'date', 'day', 'dst', 'fold',
'fromisocalendar', 'fromisoformat', 'fromordinal', 'fromtimestamp', 'hour',
'isocalendar', 'isoformat', 'isoweekday', 'max', 'microsecond', 'min', 'minute',
'month', 'now', 'replace', 'resolution', 'second', 'strftime', 'strptime',
'time', 'timestamp', 'timetuple', 'timetz', 'today', 'toordinal', 'tzinfo',
'tzname', 'utcfromtimestamp', 'utcnow', 'utcoffset', 'utctimetuple', 'weekday',
'year']
```

### 2.1.2 now()

[25]:
```
print(datetime.datetime.now())
```

```
2025-05-13 16:09:47.917455
```

[29]:
```
from datetime import datetime as tp
```

[31]:
```
current_datetime=tp.now()
print(current_datetime)
```

```
2025-05-13 16:10:55.117481
```

### 2.1.3 strftime()

[33]:
```
current_datetime.strftime("%a%d/%b/%y) and time is %H.%M.%S")
```

[33]:
```
'Tue13/May/25) and time is 16.10.55'
```

[35]:
```
current_datetime.strftime("%a,%d/%b/%Y and time is %H.%M.%S")
```

[35]: 'Tue,13/May/2025 and time is 16.10.55'

[37]:
```
current_datetime.timestamp()
```

[37]: 1747177855.117481

The timestamp() function in Python's datetime module is used to convert a datetime object to a Unix timestamp. A Unix timestamp represents the number of seconds that have elapsed since January 1, 1970, 00:00:00 UTC (also known as the "epoch").

### 2.1.4 To get the version

[39]:
```
import sys   #first way
print(sys.version)
```

```
3.11.3 | packaged by Anaconda, Inc. | (main, Apr 19 2023, 23:46:34) [MSC v.1916
64 bit (AMD64)]
```

[40]:
```
!python --version   #second way
```

```
Python 3.11.3
```

[42]:
```
import pandas as pd
```

```
C:\Users\admin\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60:
UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version
'1.3.5' currently installed).
  from pandas.core import (
```

[43]:
```
pd.__version__
```

[43]: '2.2.1'

[9]:
```
import cmath
print(dir(cmath))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atanh', 'cos', 'cosh', 'e', 'exp', 'inf',
'infj', 'isclose', 'isfinite', 'isinf', 'isnan', 'log', 'log10', 'nan', 'nanj',
'phase', 'pi', 'polar', 'rect', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau']
```

[10]:
```
#a*x*x+b*x+c=0
#(-b+-sqrt(b*b-4*a*c))/2*a
a,b,c=1,5,6
d=b*b-4*a*c
sol1=(-b+cmath.sqrt(d))/2*a
sol2=(-b-cmath.sqrt(d))/2*a
print(sol1)
print(sol2)
```

```
(-2+0j)
(-3+0j)
```

[11]:
```
a*sol1*sol1+b*sol1+c
```

[11]:
```
0j
```

[12]:
```
a*sol2*sol2+b*sol2+c
```

[12]:
```
0j
```

## 2.2 1)string handelling

- string operators

### a) + operator

[29]:
```
2+8
```

[29]:
```
10
```

[31]:
```
a=2
b=8
c=a+b
c
```

[31]:
```
10
```

[53]:
```
"nikita"+" "+"ghatbandhe"
```

[53]:
```
'nikita    ghatbandhe'
```

[28]:
```
"nikita"+"vbnn"
```

[28]:
```
'nikitavbnn'
```

[55]:
```
#concatenate(join)
a="chetana"
b="vasave"
a+" "+b
```

[55]:
```
'chetana vasave'
```

### b)* operator

[4]:
```
2*4
```

[4]:
```
8
```

[1]:
```
5*7
```

[1]:
```
35
```

[32]:
```
a=2
b=4
c=a*b  #multiplication
c
```

[32]:
```
8
```

[ ]:
```

```

[21]:
```
"chetana"*3
```

[21]:
```
'chetanachetanachetana'
```

[1]:
```
"hello "*5
```

[1]:
```
'hello hello hello hello hello '
```

**c)in and not in operator**

[35]:
```
#membership checked
s="python"
s1="data science"
s in s1
```

[35]:
```
False
```

[34]:
```
s="python"
s1="data science python"
s in s1
```

[34]:
```
True
```

[57]:
```
s1 in s
```

[57]:
```
False
```

[58]:
```
"a" in s1
```

[58]:
```
True
```

[59]:
```
s1 not in s
```

[59]:
```
True
```

[61]:
```
"z" in s
```

[61]:
```
False
```

[7]:
```
a="hello"
b="hello world"
a in b
```

[7]: True

[9]:
```
b not in a
```

[9]: True

[36]:
```
#membership checked
s="python and data science"
s1="data"
s1 in s
```

[36]: True

[38]:
```
"chetana" in s
```

[38]: False

[10]:
```
"chetana" not in s
```

[10]: True

[11]:
```
s1 not in s
```

[11]: False

## 2.3 String functions

### 1) chr

[62]:
```
#ascii code to character
chr(35)
```

[62]: '#'

[63]:
```
chr(97)
```

[63]: 'a'

[39]:
```
chr(89)
```

[39]: 'Y'

[41]:
```
chr(45)
```

[41]: '-'

**2)ord()**

[16]:
```
#char to ascii
ord("#")
```

[16]: 35

[13]:
```
ord("@")
```

[13]: 64

[43]:
```
ord("y")
```

[43]: 121

[45]:
```
ord("o")
```

[45]: 111

**3) len()**

[41]:
```
#space counts
#len counts starts with 1
s="python and data science"
len(s)
```

[41]: 23

[17]:
```
b="hi i am here"
len(b)
```

[17]: 12

**4)str()**

[43]:
```
a=123 #int
b=str(a)
```

[44]:
```
b
```

[44]: '123'

**5)upper()**

[47]:
```
s="python and data science"
s.upper()
```

[47]: 'PYTHON AND DATA SCIENCE'

mutable : changeble ==> we can add, remove or delete after declaration ==> list,dict immutable: non changeble ==> we cant add, remove or delete after declaration==>int,string,tuple

**6)lower()**

[6]:
```
s="Python and data Science"
s.lower()
```

[6]: 'python and data science'

[28]:
```
s2="PYTHON"
s2.lower()
```

python

**7)capitalize()**

[ ]:
```
#converts only first letter of string into upper
```

[50]:
```
s="python"
s.capitalize()
```

[50]: 'Python'

[51]:
```
t="data science"
t.capitalize()
```

[51]: 'Data science'

[19]:
```
s3="python and data science"
s3.capitalize()
```

[19]: 'Python and data science'

**8)title()**

[ ]:
```
#converts all words first letter of string into upper
```

[43]:
```
s3="python and data science"
s3.title()
```

[43]: 'Python And Data Science'

**9)replace()**

[1]:
```
s="data science"
s.replace("science","analytics")
```

[1]: 'data analytics'

[40]:
```
s.replace("Science","analytics")
```

[40]: 'data science'

**10)count()**

[15]:
```
s1="data and science"
s1.count("a")
```

[15]: 3

[25]:
```
s1.count("z")
```

[25]: 0

[27]:
```
s1.count("data")
```

[27]: 1

[17]:
```
s1.count("python")
```

[17]: 0

[ ]:
```

```