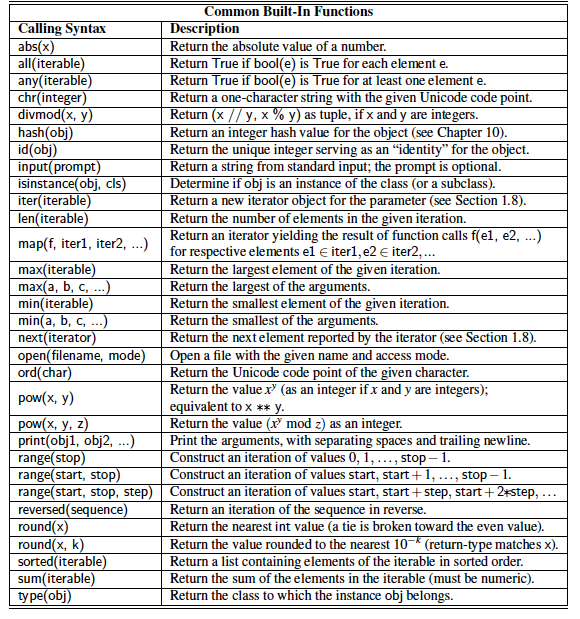
|  |  |
| --- | --- |
| IF  if a!=b:  a=0  elif a=b:  b=0  else:  a=1  b=1 | WHILE loop  while a!=b:  a+=1 |

|  |  |
| --- | --- |
| FOR loop  for j in range(len(data)):  data(j)=0 | for val in data:  a=val  [data is list, tuple or other iterable data str; val is std identifier] |

Function: sorted(data). [Method: data.sorted()]

|  |  |
| --- | --- |
| def count(data, target):  n=0 ...  return n | def foo(a, b=1, c=11)  return a+b+c  foo(1, b=3) =>foo(1, 3, 11)  foo(1) =>foo(1, 1, 11) |

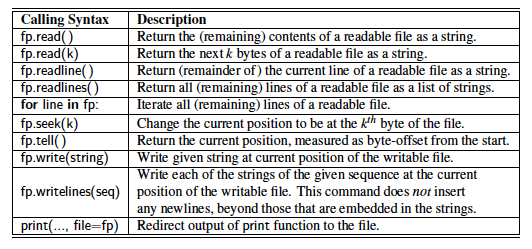
Built-in Function



INPUT

data=input(‘enter your day of birth separated by space: ’)

pieces=data.split() [returns lists of strings sep by space]



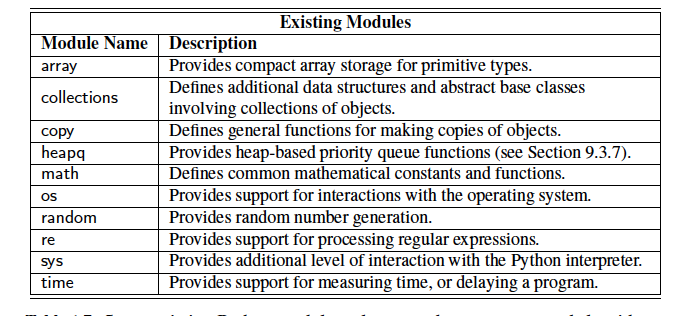
fp=open(‘sample.text’)

default access mode is ‘r’ [wr for binary files]

others: ‘w’ overwrite [wb for binary files]

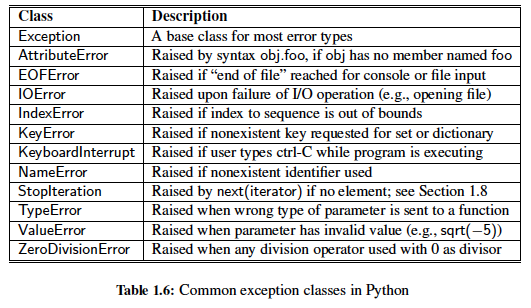
‘a’ append to end of existing

import \_\_ from math/numpy etc



Raising Exception

raise ValueError(‘x must be negative’)



Catching Exception

* “look before you leap” if y!=0:

ratio=x/y

else:

ddo smth else

* “it is easier to ask for forgiveness that it’s to get permission” [non exceptional cases run faster, so use when exceptions are unlikely]

try:

ratio=x/y

except zeroDivisionError:

do smth else

list(range(1000)) - a list instance with values from 0 to 999

list(d.values()) - a list that has elements based upon array of values of dictionary d

Generator - yield - to indicate each element of the series - deleted instantly, so cannot store values

Function - return

|  |  |
| --- | --- |
| def factors(n):  result=[]  for k in range(1, n+1):  if n %k==0:  results.append(k)  return results | def factors(n):    for k in range(1, n+1):  if n %k==0:  yield k |
| factors(100) | a=factors1(100) [<generator object factors1 at 0x104980d58>]  for i in a:  print(i) |
| [1, 2, 4, 5, 10, 20, 25, 50, 100] | 1 2 4 5 10 20 25 50 100 |

|  |  |
| --- | --- |
| if n>=0:  param=n  else:  param=-n  result=foo(param) | param=n if n>=0 else -n  result=foo(param)  result=foo(n if n>=0 else -n) |
| squares=[]  for k in range(1, n+1):  squares.append(k\*k) | squares=[k\*k for k in range(1, n+1) ] |

|  |
| --- |
| factor=[k for k in range(1, n+1) if n%k==0] list |
| factor={k for k in range(1, n+1) if n%k==0} set |
| factor=(k for k in range(1, n+1) if n%k==0) generator(eff when no need for saving results) |
| factor={k: k\*k for k in range(1, n+1) if n%k==0} dictionary |

return x,y => single obj that is tuple (x,y)

a,b,c,d=range(7,11) => a=7, b=8 etc

quotient, remainder=divmode(a,b)

for x,y in [(7,2), (5,8),(6,4)]