

Python Numbers and 1-D Arrays of Numbers

Numbers

- Reading assignment: Chapter 3 from Zelle
- Numbers w/o fractional part: integers, or int for short
- Numbers w/ fractional part: floating-point numbers, or floats for short

Int or float

- Type does not have to be declared explicitly
- Python infers type based on
 - presence or absence of decimal point
 - operands
- Special Python function `type` returns the type float or int

```
>>> type(2)
```

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

```
>>> type(2.0)
```

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

```
>>> type(1+2)
```

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

```
>>> type(1 + 2.0)
```

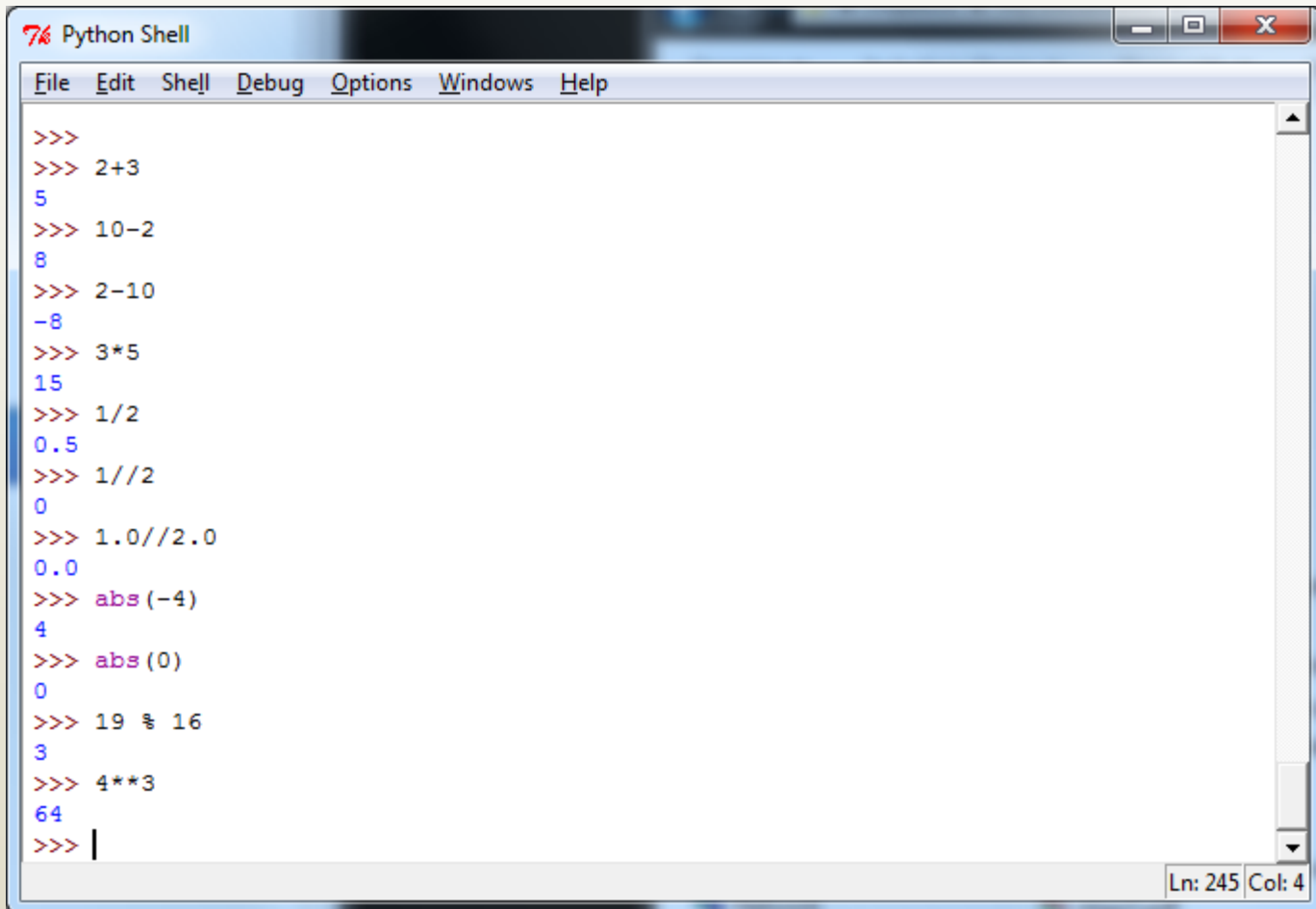
```
<class 'float'>
```

Built-in numeric operations

| operator | operation |
|----------|------------------|
| + | addition |
| - | subtraction |
| * | multiplication |
| / | float division |
| ** | exponentiation |
| abs() | absolute value |
| // | integer division |
| % | remainder |

Table 3.1: Python built-in numeric operations

Built-in numeric operations

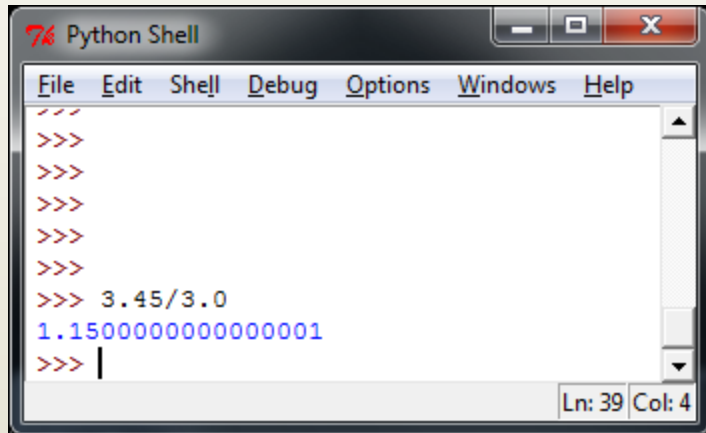


A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main area shows a series of Python commands and their outputs. The commands are: `>>>`, `>>> 2+3`, `>>> 10-2`, `>>> 2-10`, `>>> 3*5`, `>>> 1/2`, `>>> 1//2`, `>>> 1.0//2.0`, `>>> abs(-4)`, `>>> abs(0)`, `>>> 19 % 16`, and `>>> 4**3`. The outputs are: `5`, `8`, `-8`, `15`, `0.5`, `0`, `0.0`, `4`, `0`, `3`, and `64`. The window also shows a status bar at the bottom right with "Ln: 245" and "Col: 4".

```
>>>
>>> 2+3
5
>>> 10-2
8
>>> 2-10
-8
>>> 3*5
15
>>> 1/2
0.5
>>> 1//2
0
>>> 1.0//2.0
0.0
>>> abs(-4)
4
>>> abs(0)
0
>>> 19 % 16
3
>>> 4**3
64
>>> |
```

Ln: 245 Col: 4

iClicker Question



The image shows a screenshot of a Python Shell window. The window has a title bar that says "Python Shell" and a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area contains the following text:

```
>>>  
>>>  
>>>  
>>>  
>>>  
>>> 3.45/3.0  
1.1500000000000001  
>>> |
```

The status bar at the bottom right shows "Ln: 39 Col: 4".

- Explain the result in the Python shell window
 - A. The result should be 1.15, you have discovered a Python bug
 - B. The Python shell window image was photoshop-ed to show the incorrect result
 - C. 3.45 cannot be represented precisely in base 2, so, since one operand is approximate, the division result is approximate
 - D. 3.45 divided by 3.0 is a little more than 1.15, there's nothing to explain
 - E. We should have used integer division `//` for the correct result

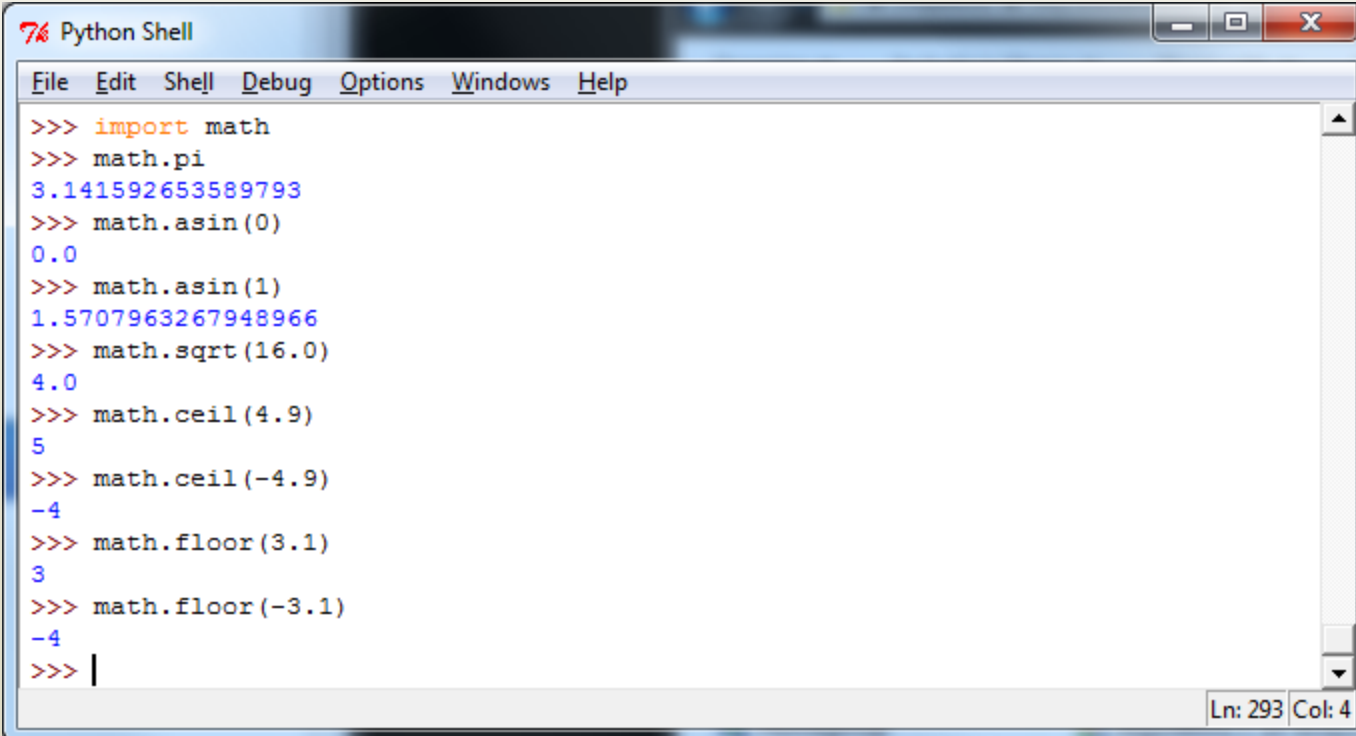
Some math library functions

| Python | Mathematics | English |
|----------|---------------------|--|
| pi | π | An approximation of pi. |
| e | e | An approximation of e . |
| sqrt(x) | \sqrt{x} | The square root of x. |
| sin(x) | $\sin x$ | The sine of x. |
| cos(x) | $\cos x$ | The cosine of x. |
| tan(x) | $\tan x$ | The tangent of x. |
| asin(x) | $\arcsin x$ | The inverse of sine x. |
| acos(x) | $\arccos x$ | The inverse of cosine x. |
| atan(x) | $\arctan x$ | The inverse of tangent x. |
| log(x) | $\ln x$ | The natural (base e) logarithm of x |
| log10(x) | $\log_{10} x$ | The common (base 10) logarithm of x. |
| exp(x) | e^x | The exponential of x. |
| ceil(x) | $\lceil x \rceil$ | The smallest whole number $\geq x$ |
| floor(x) | $\lfloor x \rfloor$ | The largest whole number $\leq x$ |

Table 3.2: Some math library functions

Some math library functions

- Using math library
 - import math
 - Prefix function name with math.



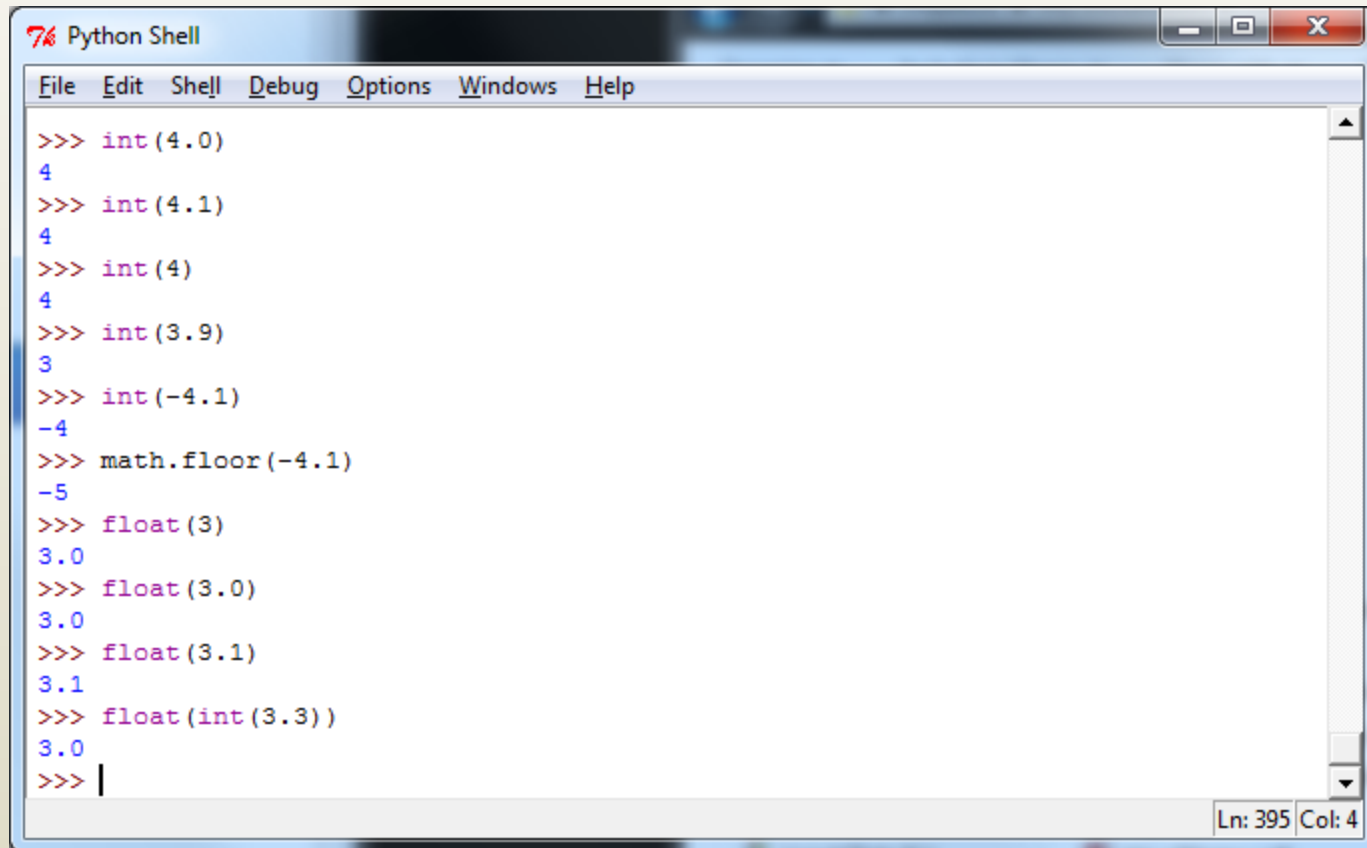
A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area shows the following code and output:

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.asin(0)
0.0
>>> math.asin(1)
1.5707963267948966
>>> math.sqrt(16.0)
4.0
>>> math.ceil(4.9)
5
>>> math.ceil(-4.9)
-4
>>> math.floor(3.1)
3
>>> math.floor(-3.1)
-4
>>> |
```

The status bar at the bottom right shows "Ln: 293 Col: 4".

Explicit type conversion

- Force int to float or float to int

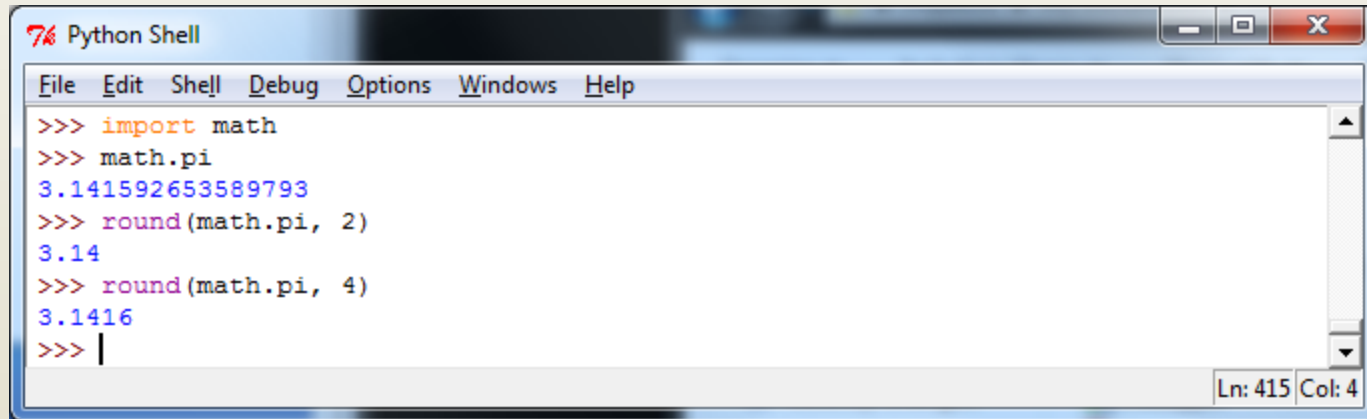


```
Python Shell
File Edit Shell Debug Options Windows Help
>>> int(4.0)
4
>>> int(4.1)
4
>>> int(4)
4
>>> int(3.9)
3
>>> int(-4.1)
-4
>>> math.floor(-4.1)
-5
>>> float(3)
3.0
>>> float(3.0)
3.0
>>> float(3.1)
3.1
>>> float(int(3.3))
3.0
>>> |
```

Ln: 395 Col: 4

Rounding

- Allows specifying the number of decimal places



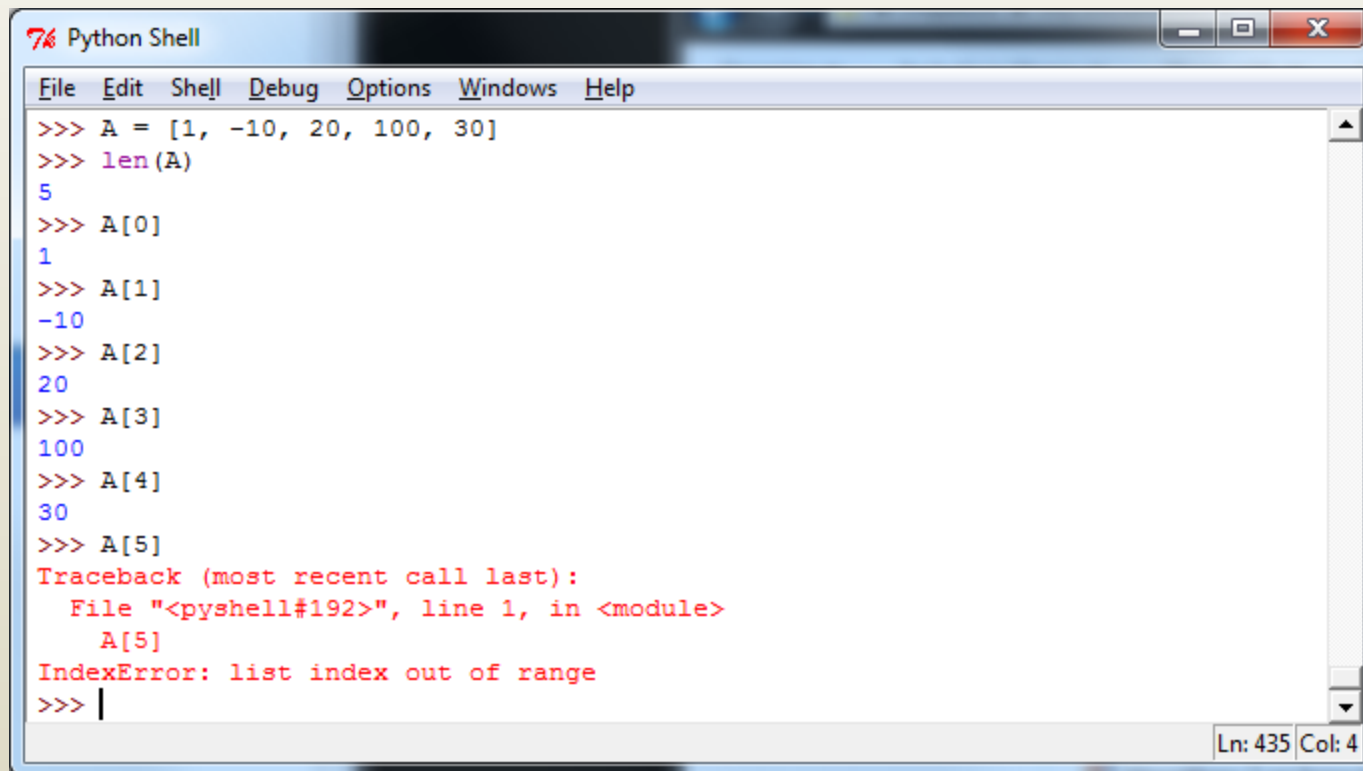
```
Python Shell
File Edit Shell Debug Options Windows Help
>>> import math
>>> math.pi
3.141592653589793
>>> round(math.pi, 2)
3.14
>>> round(math.pi, 4)
3.1416
>>> |
```

Ln: 415 Col: 4

1-D array of numbers

[element₀, element₁, ..., element_{n-1}]

- 0-based indices
- len function returns length of array, i.e. the number of elements (“n”)

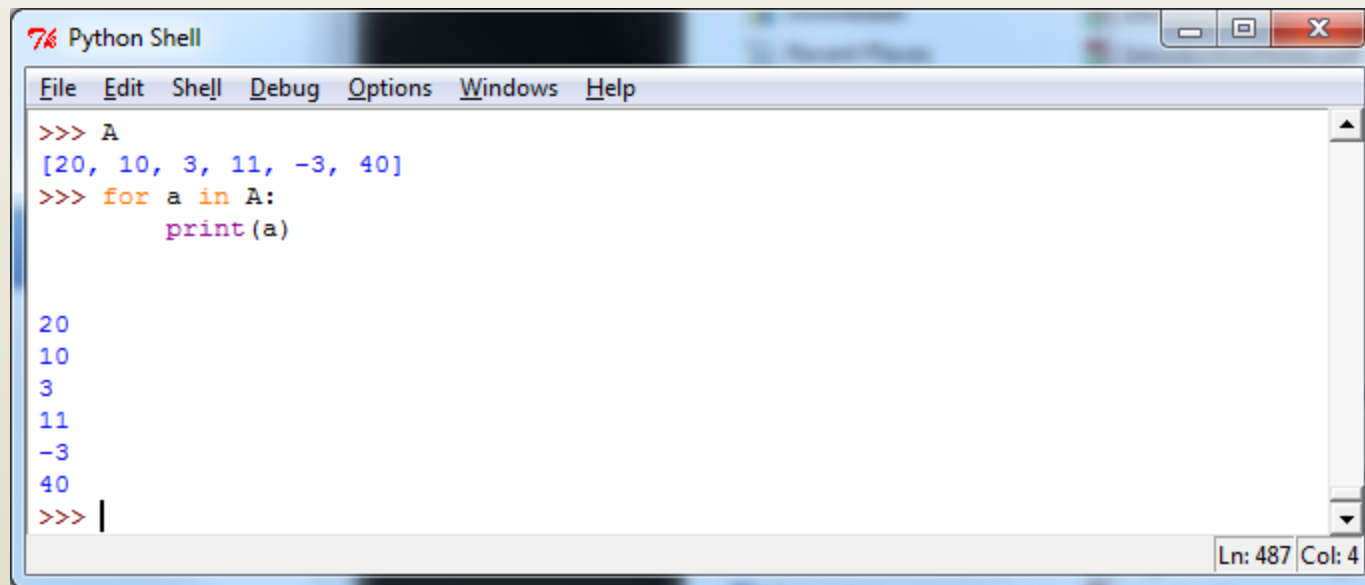


```
Python Shell
File Edit Shell Debug Options Windows Help
>>> A = [1, -10, 20, 100, 30]
>>> len(A)
5
>>> A[0]
1
>>> A[1]
-10
>>> A[2]
20
>>> A[3]
100
>>> A[4]
30
>>> A[5]
Traceback (most recent call last):
  File "<pyshell#192>", line 1, in <module>
    A[5]
IndexError: list index out of range
>>> |
```

Ln: 435 Col: 4

Traversal of 1-D array, version 1

- Using for loop
 - Directly on elements

A screenshot of a Python Shell window. The window has a title bar with a Python logo and the text "Python Shell". Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main area of the window contains the following text:

```
>>> A
[20, 10, 3, 11, -3, 40]
>>> for a in A:
    print(a)
```

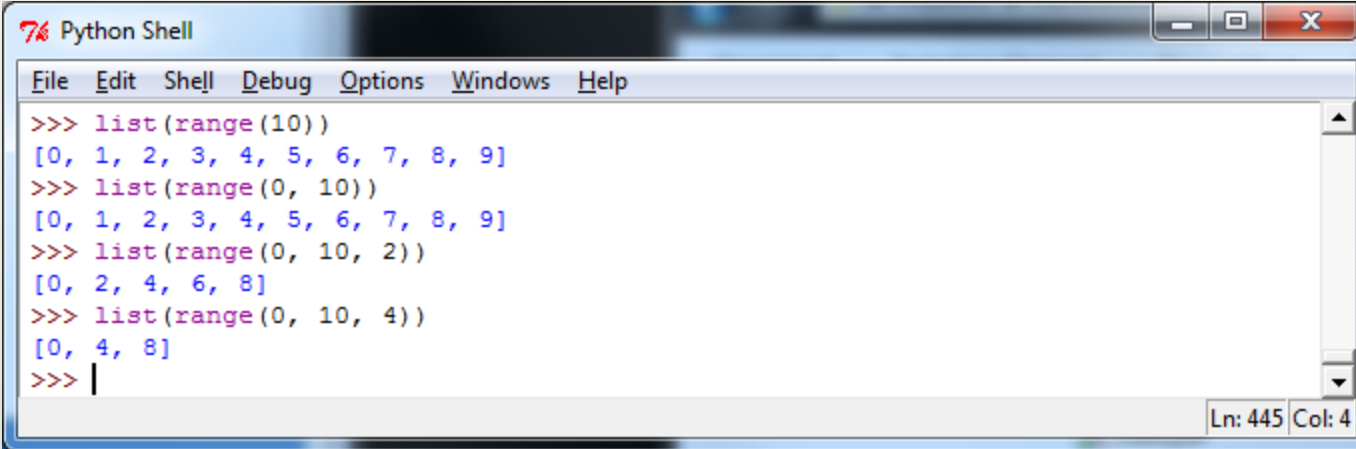
Below the code, the output of the program is displayed:

```
20
10
3
11
-3
40
>>> |
```

The status bar at the bottom right of the window shows "Ln: 487 Col: 4".

Traversal of 1-D array, version 2

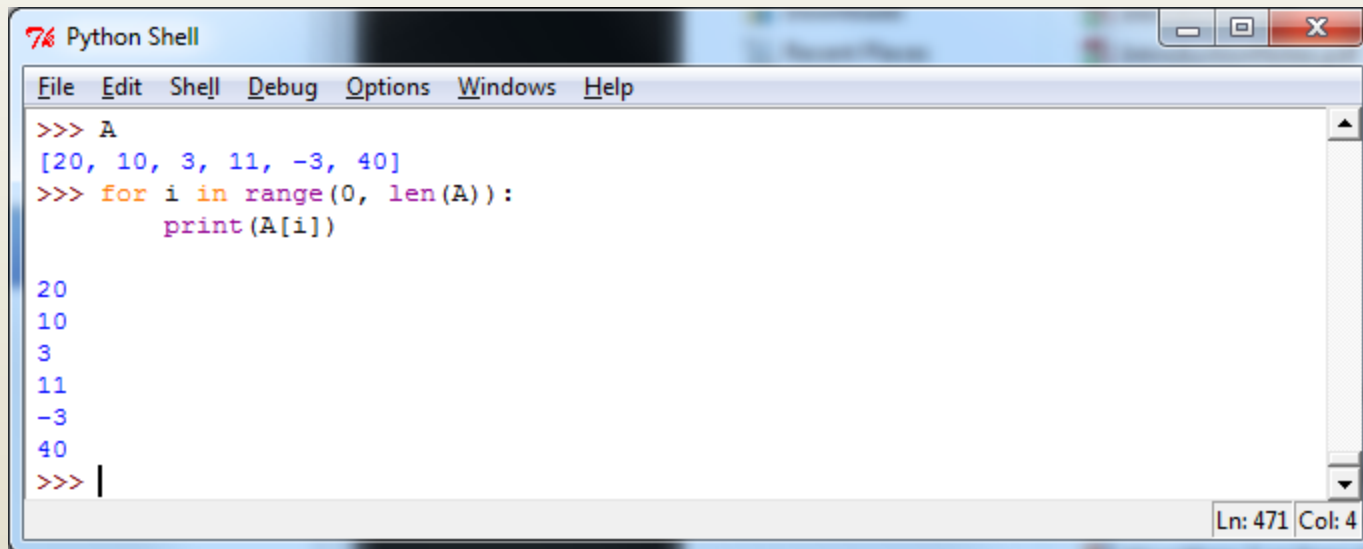
- Using range function ...

A screenshot of a Python Shell window. The window has a title bar with a Python logo and the text "Python Shell". Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Windows, and Help. The main area of the window contains a series of Python commands and their outputs. The commands are: list(range(10)), list(range(0, 10)), list(range(0, 10, 2)), and list(range(0, 10, 4)). The outputs are: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [0, 2, 4, 6, 8], and [0, 4, 8]. The window also has a status bar at the bottom right showing "Ln: 445 Col: 4".

```
Python Shell
File Edit Shell Debug Options Windows Help
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(0, 10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(0, 10, 2))
[0, 2, 4, 6, 8]
>>> list(range(0, 10, 4))
[0, 4, 8]
>>> |
Ln: 445 Col: 4
```


Traversal of 1-D array , version 2

- Using range function ... and for loop on index

A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area contains the following code:

```
>>> A
[20, 10, 3, 11, -3, 40]
>>> for i in range(0, len(A)):
    print(A[i])
```

The output of the code is displayed below the input lines:

```
20
10
3
11
-3
40
```

The prompt ">>>" is followed by a vertical bar "|". The status bar at the bottom right shows "Ln: 471 Col: 4".

SortMin in Python

```
SortMin.py - C:\Python32\CS177\Intro\SortMin.py
File Edit Format Run Options Windows Help
# function (i.e. sub-algorithm) for finding index of minimum element
# starting at index i0
def MinIndex(A, i0):
    n = len(A)
    imin = i0
    for i in range(i0+1, n):
        if A[i] < A[imin]:
            imin = i
    return imin

# function for sorting array by finding minimum repeatedly
def SortMin(A):
    n = len(A)
    for i in range(0, n-1):

        #find index of minimum, starting at i
        imin = MinIndex(A, i)

        # swap A[i] and A[imin]
        tmp = A[i]
        A[i] = A[imin]
        A[imin] = tmp

    return A
Ln: 5 Col: 13
```

```
Python Shell
File Edit Shell Debug Options Windows Help
>>> SortMin([-10, 20, -100, -100, -9, 17, 20, 80, -1000])
[-1000, -100, -100, -10, -9, 17, 20, 20, 80]
>>> |
Ln: 489 Col: 4
```