Loops and if-Statements

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #3

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction:

for-loops, while-loops, and if-statements are really useful

- This lecture covers how to use these with Python

Note: Python does not use end-statements

- Indentation indicated which lines are within a loop

In Python, carriage returns and intendations have meaning

- unlike C where they are decorative

```
for i in range(0,6):
    d1 = i
    for j in range(0,6):
        d2 = j
        y = d1 + d2
```

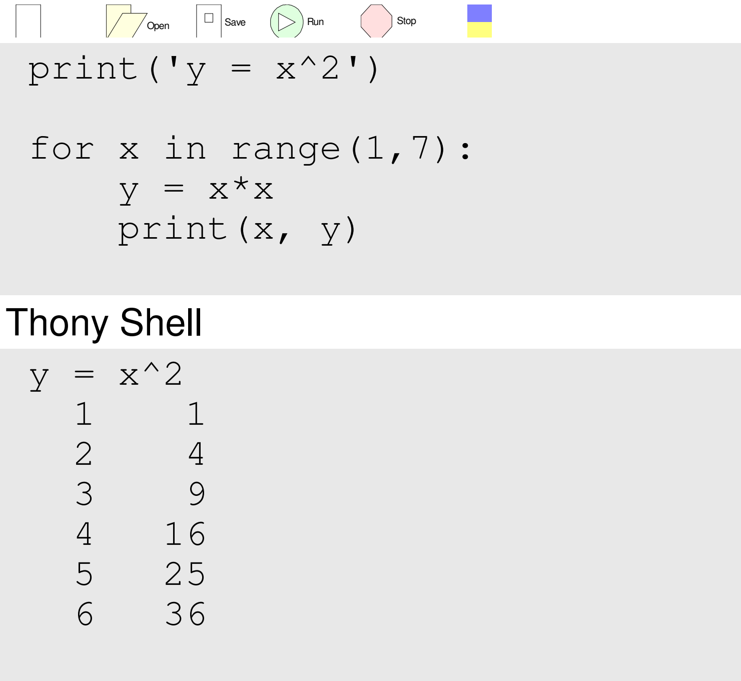
```
t = 0
dt = 0.01
while(t < 5):
    y = sin(t)
    t += dt
```

```
if(x < 3):
    y = 2*x + 4
elif(x < 5):
    y = 3 - 2*x
else:
    y = 0
```

For-Loops

Similar to Matlab:

- A variable is required for the loop
- The variable increments as you go through the loop
- The looping continues as long as you are less than the end
  - *different than Matlab & C*
  - *Matlab and C use less than or equal to*



```
print('y = x^2')

for x in range(1,7):
    y = x*x
    print(x, y)
```

Thony Shell

```
y = x^2
  1    1
  2    4
  3    9
  4   16
  5   25
  6   36
```

For-Loops Syntax

A colon is required

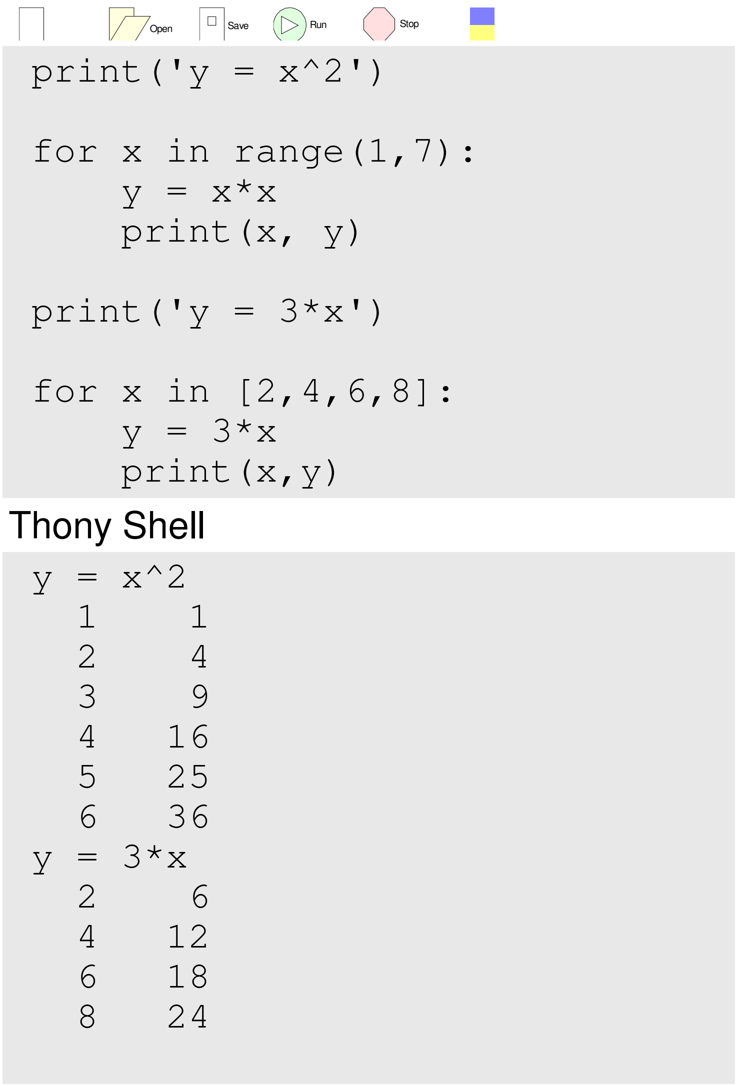
- This marks the start of the loop

Indentation is required

- This indicated instructions within the loop
- Four spaces are standard

There are no end statements

- Removing indentation indicated the end of the loop



```
print('y = x^2')

for x in range(1,7):
    y = x*x
    print(x, y)

print('y = 3*x')

for x in [2,4,6,8]:
    y = 3*x
    print(x,y)
```

Thony Shell

```
y = x^2
  1    1
  2    4
  3    9
  4   16
  5   25
  6   36
y = 3*x
  2    6
  4   12
  6   18
  8   24
```

Nested Loops in Python

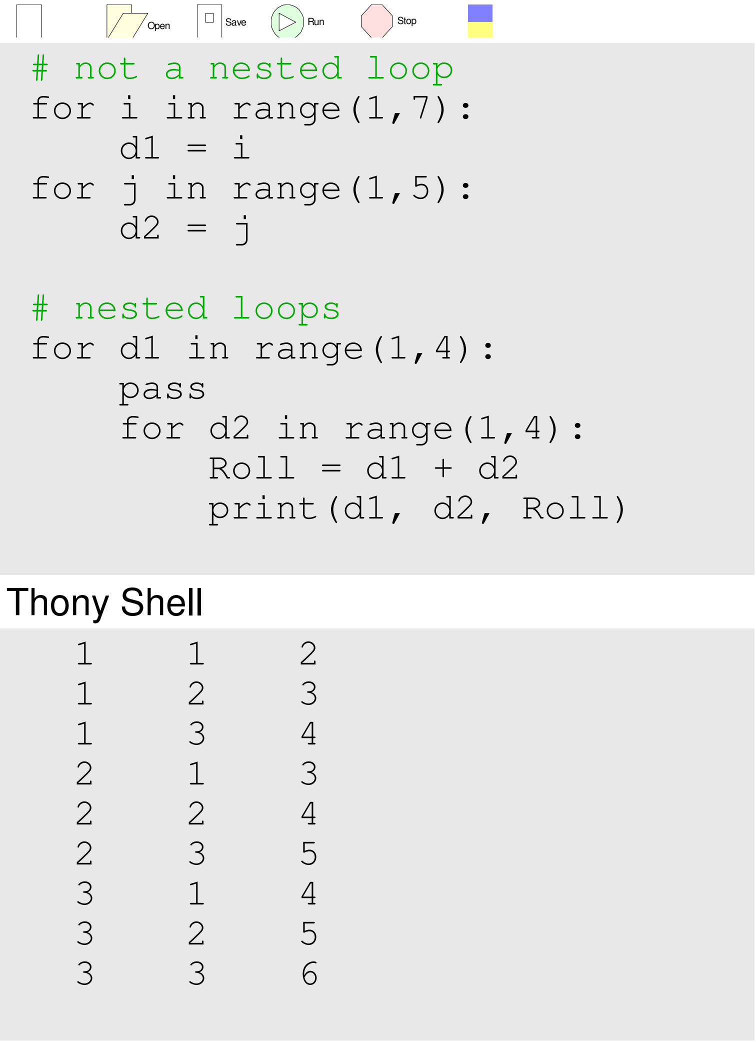
Nested loops are allowed

Indentation is important

- To be part of a loop, the indentation must be maintained
- Remove the indentation to end the loop

For nested loops:

- Add another level of indentation



```
# not a nested loop
for i in range(1,7):
    d1 = i
for j in range(1,5):
    d2 = j

# nested loops
for d1 in range(1,4):
    pass
    for d2 in range(1,4):
        Roll = d1 + d2
        print(d1, d2, Roll)
```

Thony Shell

```
  1    1    2
  1    2    3
  1    3    4
  2    1    3
  2    2    4
  2    3    5
  3    1    4
  3    2    5
  3    3    6
```

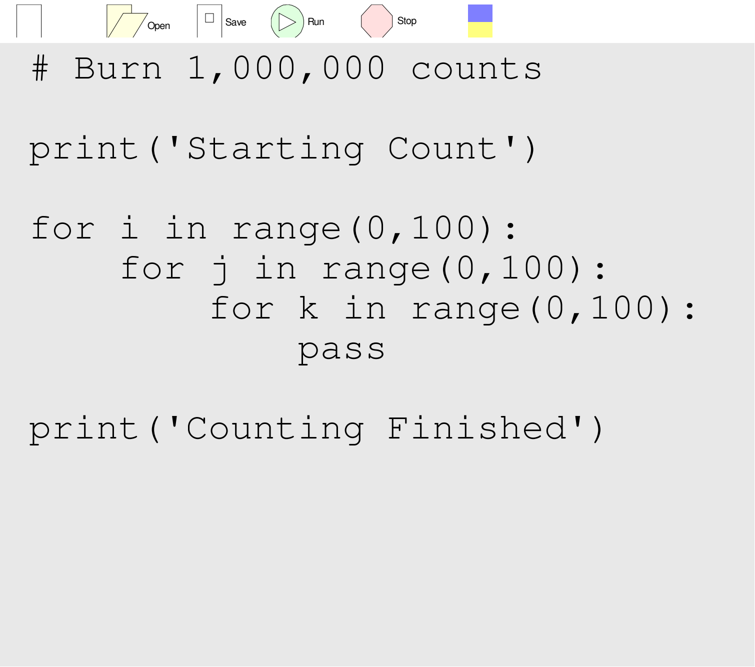
pass statement

Each loop *must* contain 1+ statements

- You can use a *pass* statement
- Behaves like a nop command

Example:

- Count to 1,000,000
- Wastes time
- (there are better ways to do this)



```
# Burn 1,000,000 counts

print('Starting Count')

for i in range(0,100):
    for j in range(0,100):
        for k in range(0,100):
            pass

print('Counting Finished')
```

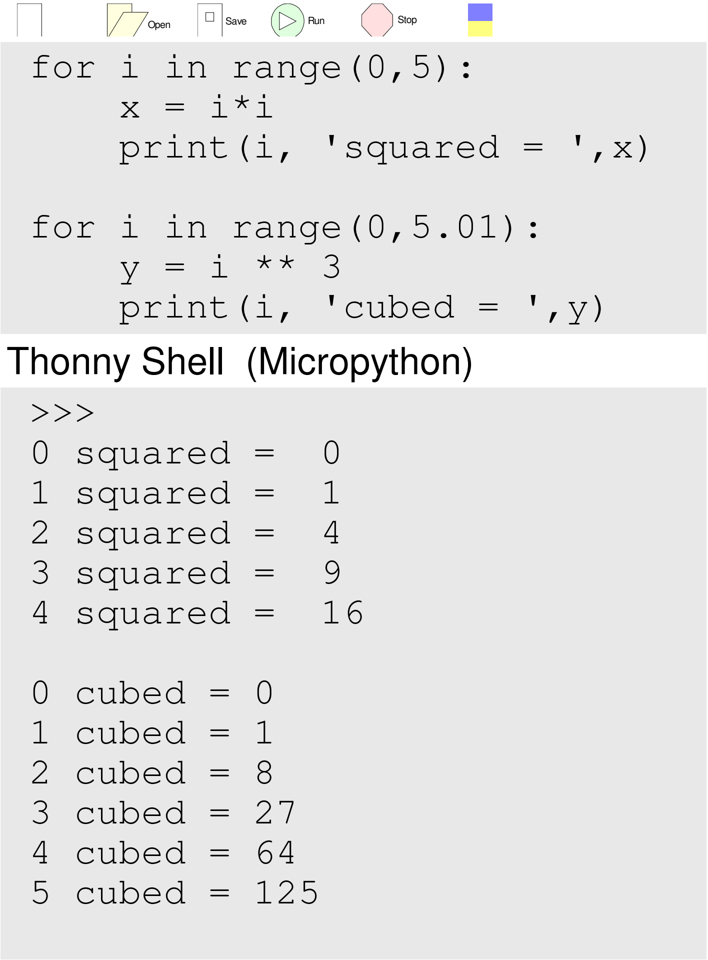
range() statement

Commonly used in for loops

for i in range(0,5):

- i starts at 0
  - *same as Matlab*
- Increments by one each loop
  - *same as Matlab*
- Loops while i < 5
  - *slightly different than Matlab*
  - *Matlab and C loop while i <= 5*

To make similar to Matlab, make the 2nd number 5.01



```
for i in range(0,5):
    x = i*i
    print(i, 'squared = ',x)

for i in range(0,5.01):
    y = i ** 3
    print(i, 'cubed = ',y)
```

Thonny Shell (Micropython)

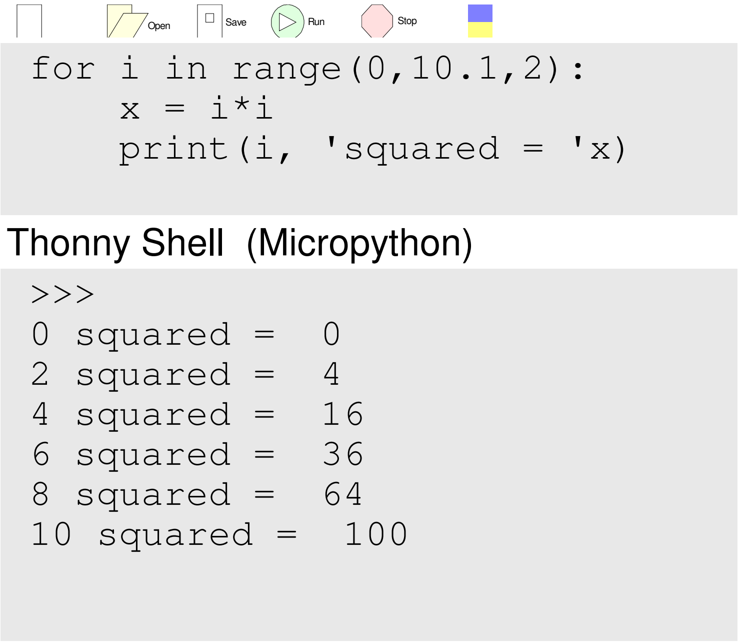
```
>>>
0 squared =  0
1 squared =  1
2 squared =  4
3 squared =  9
4 squared =  16

0 cubed = 0
1 cubed = 1
2 cubed = 8
3 cubed = 27
4 cubed = 64
5 cubed = 125
```

Range statement (cont'd)

Add a 3rd number to set the step size

- Go from 0
- to 10.1
- step size 2



```
for i in range(0,10.1,2):
    x = i*i
    print(i, 'squared = 'x)
```

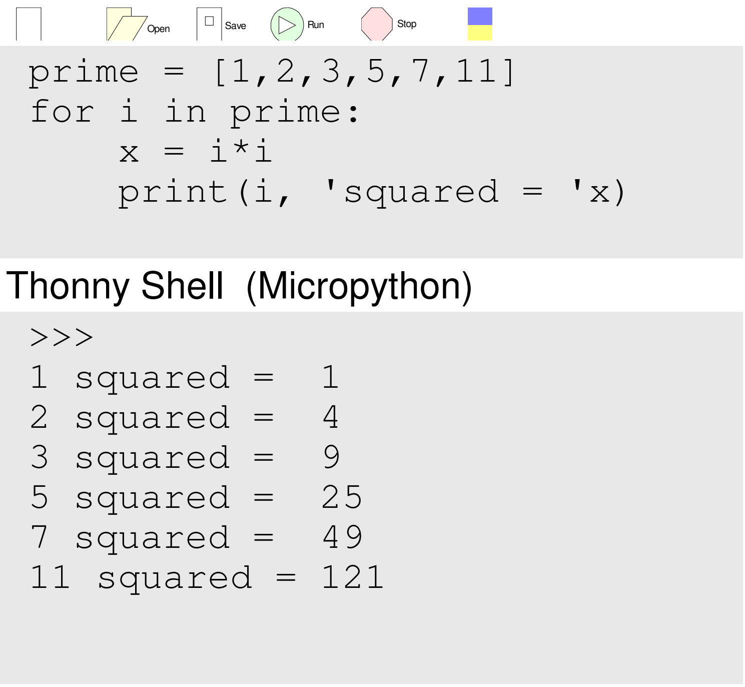
Thonny Shell (Micropython)

```
>>>
0 squared =  0
2 squared =  4
4 squared =  16
6 squared =  36
8 squared =  64
10 squared =  100
```

Stepping Through an Array

You can also step through an array.

Example: Squares of prime numbers



```
prime = [1,2,3,5,7,11]
for i in prime:
    x = i*i
    print(i, 'squared = 'x)
```

Thonny Shell (Micropython)

```
>>>
1 squared =  1
2 squared =  4
3 squared =  9
5 squared =  25
7 squared =  49
11 squared = 121
```

For-Loop Example: Timer2 Interrupts

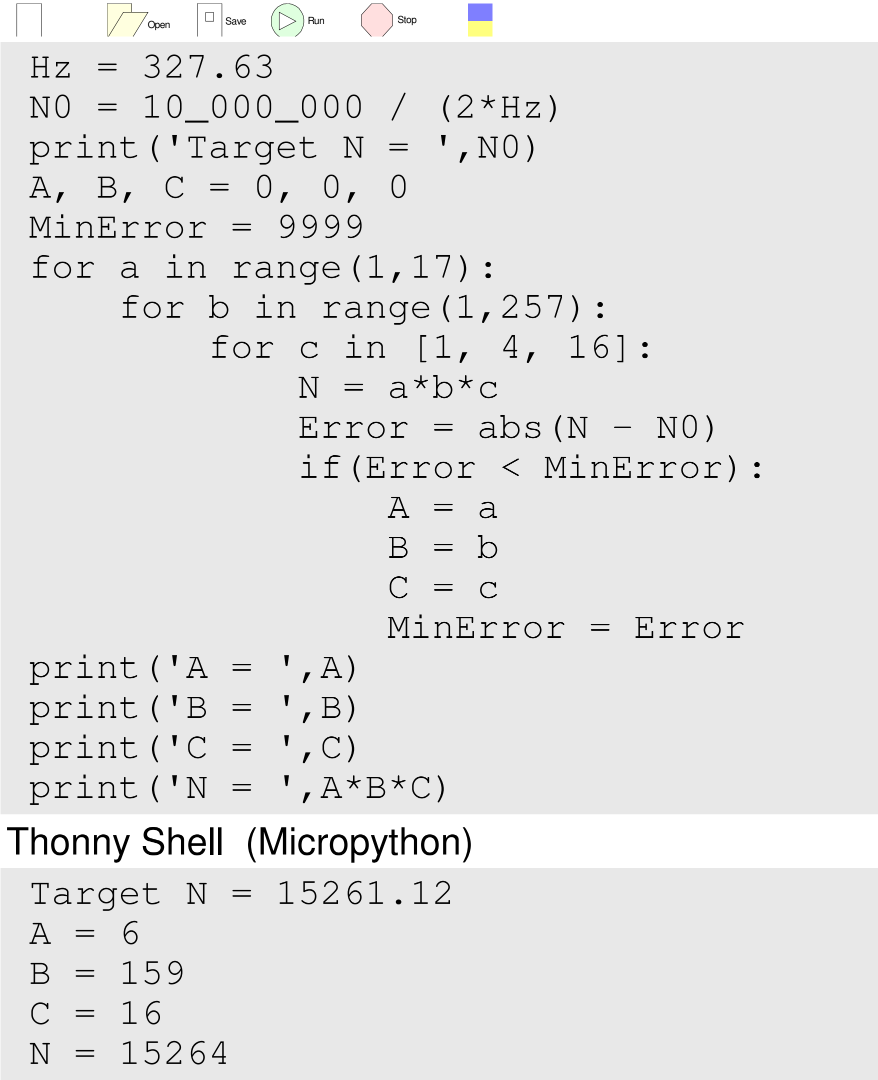
Recall from ECE 376.....

- Using Timer2 interrupts:
- Find A*B*C to produce 327.63Hz
- A = 1..16
- B = 1..256
- C = 1, 4, or 16

What combination is best?

Solution:

- Go through every combination
- Keep the solution which is closest



```
Hz = 327.63
N0 = 10_000_000 / (2*Hz)
print('Target N = ',N0)
A, B, C = 0, 0, 0
MinError = 9999
for a in range(1,17):
    for b in range(1,257):
        for c in [1, 4, 16]:
            N = a*b*c
            Error = abs(N - N0)
            if(Error < MinError):
                A = a
                B = b
                C = c
                MinError = Error
print('A = ',A)
print('B = ',B)
print('C = ',C)
print('N = ',A*B*C)
```

Thonny Shell (Micropython)

```
Target N = 15261.12
A = 6
B = 159
C = 16
N = 15264
```

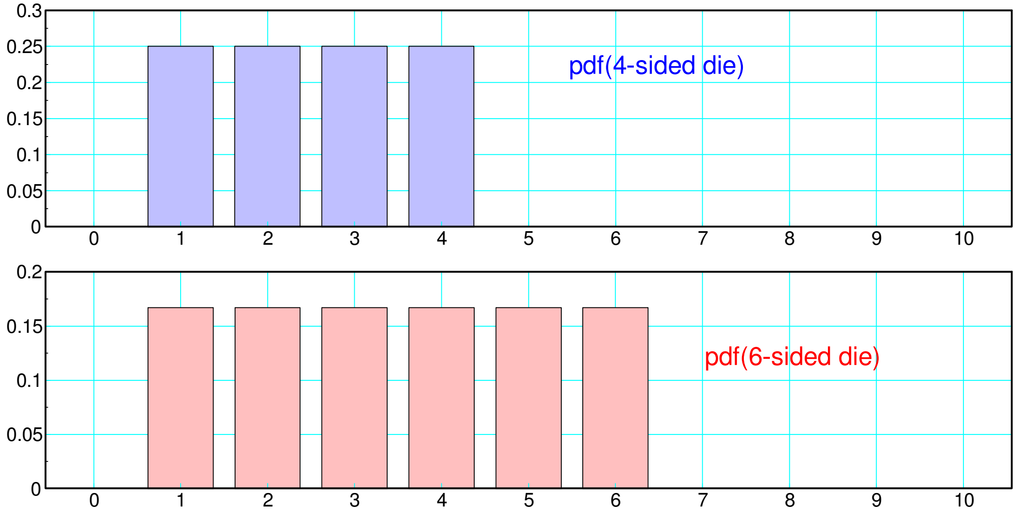
For-Loop Example: Creating Arrays

As an example of using for-loops, create an array which indicated the probability of getting the numbers 0..10 when rolling

- A 4-sided die, and a 6-sided die

The array should like the following:

k (die roll)	0	1	2	3	4	5	6	7	8	9	10
d4	0	1/4	1/4	1/4	1/4	0	0	0	0	0	0
d6	0	1/6	1/6	1/6	1/6	1/6	1/6	0	0	0	0



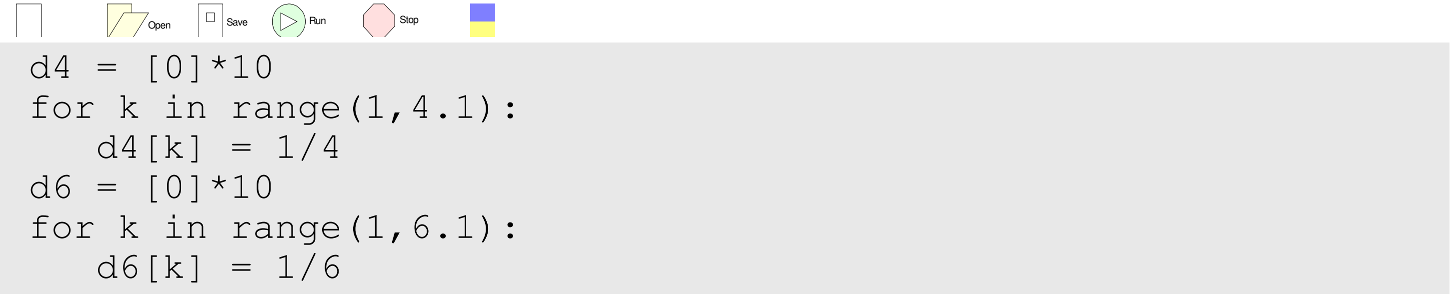
pdf for a 4-sided and 6-sided die

In Micropython, there are a couple of ways of doing this:

Option #: No Finesse

```
d4 = [0,1/4,1/4,1/4,1/4,0,0,0,0,0,0]
d6 = [0,1/6,1/6,1/6/1/6,1/6,1/6,0,0,0,0]
```

Option 2: Use a for-loop

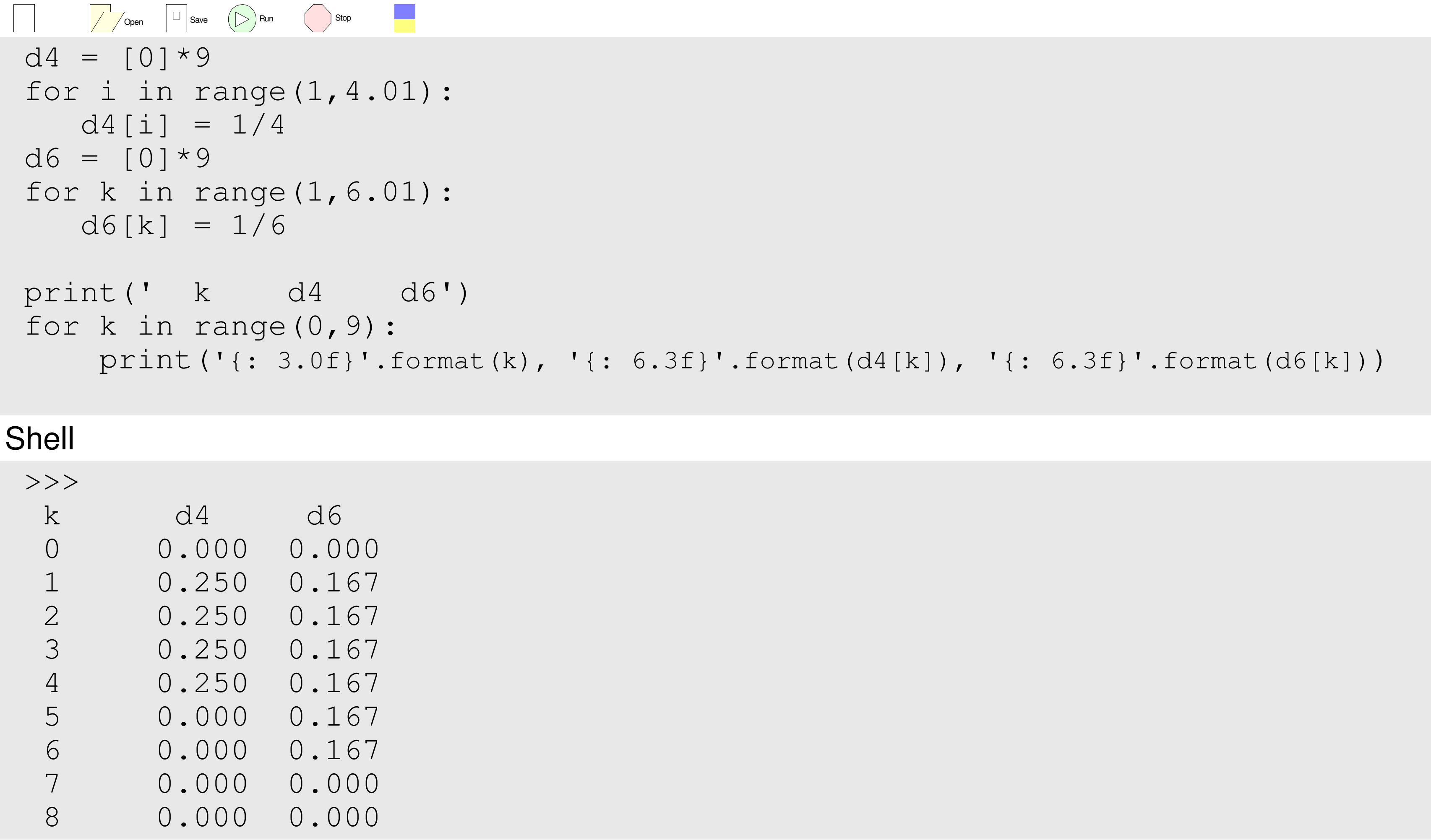


```
d4 = [0]*10
for k in range(1,4.1):
   d4[k] = 1/4
d6 = [0]*10
for k in range(1,6.1):
   d6[k] = 1/6
```

Option #3: Use a subroutine

something we'll cover shortly

You can also format the output:



```
d4 = [0]*9
for i in range(1,4.01):
   d4[i] = 1/4
d6 = [0]*9
for k in range(1,6.01):
   d6[k] = 1/6

print('  k    d4    d6')
for k in range(0,9):
    print('{: 3.0f}'.format(k), '{: 6.3f}'.format(d4[k]), '{: 6.3f}'.format(d6[k]))
```

```
>>>
 k      d4     d6
 0     0.000  0.000
 1     0.250  0.167
 2     0.250  0.167
 3     0.250  0.167
 4     0.250  0.167
 5     0.000  0.167
 6     0.000  0.167
 7     0.000  0.000
 8     0.000  0.000
```

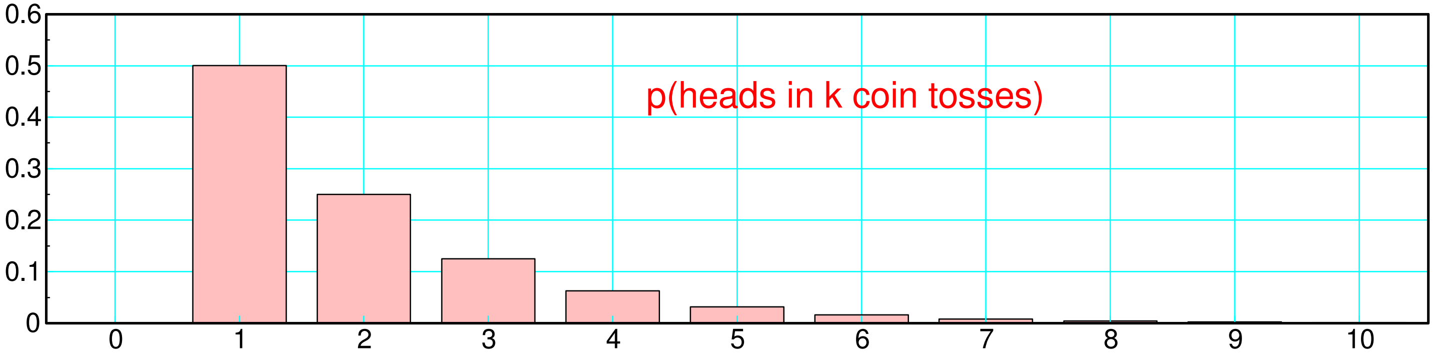
While-Loops

A while loop keeps going

- As long as a condition holds, or
- Until you encounter a *break* statement

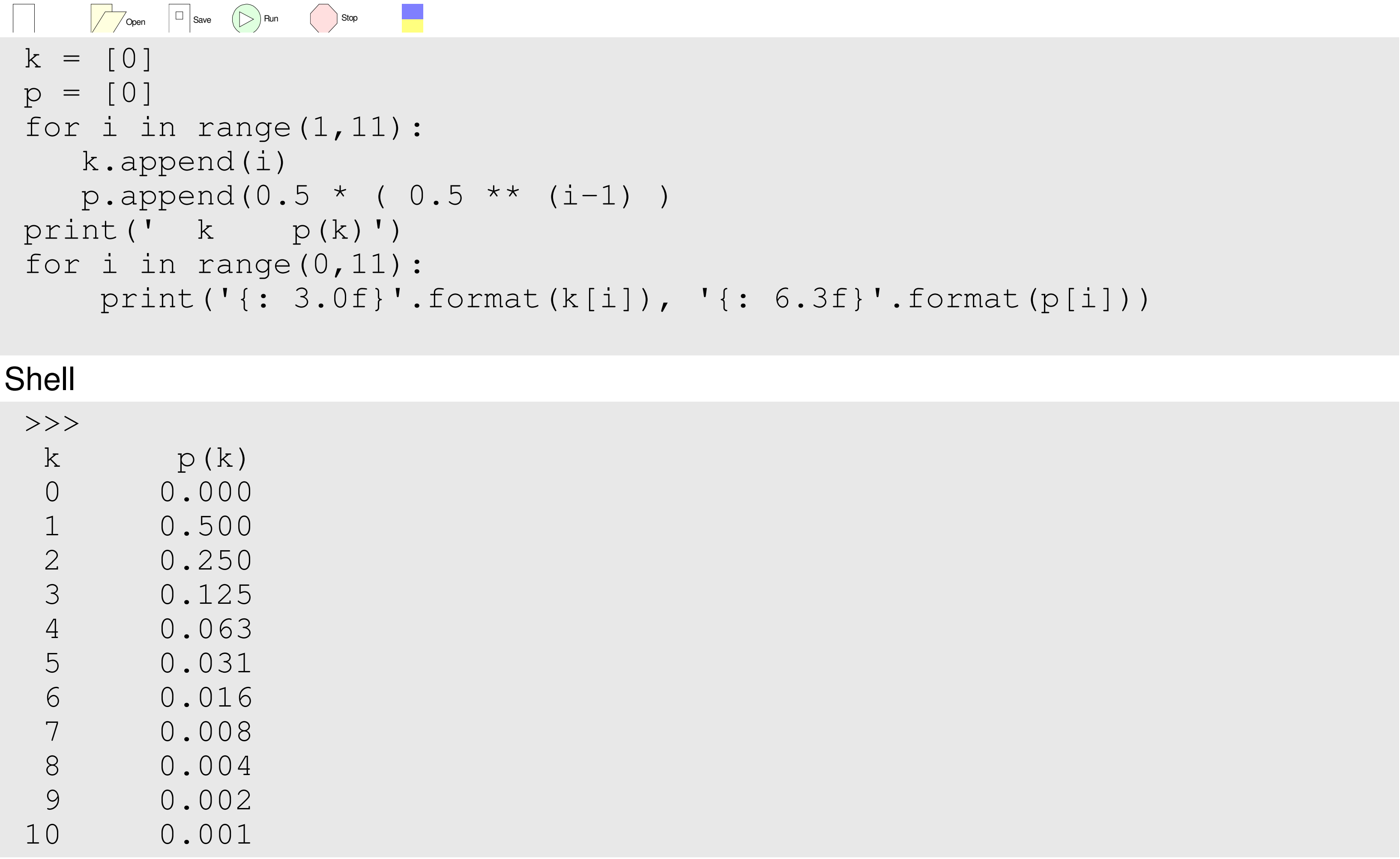
For example, the probability of flipping a coin k times before you get a heads (exponential distribution) is:

$$p(k) = \left(\frac{1}{2}\right)\left(\frac{1}{2}\right)^{k-1} u(k-1)$$



This series goes out to infinity

- Truncate the series using a for-loop

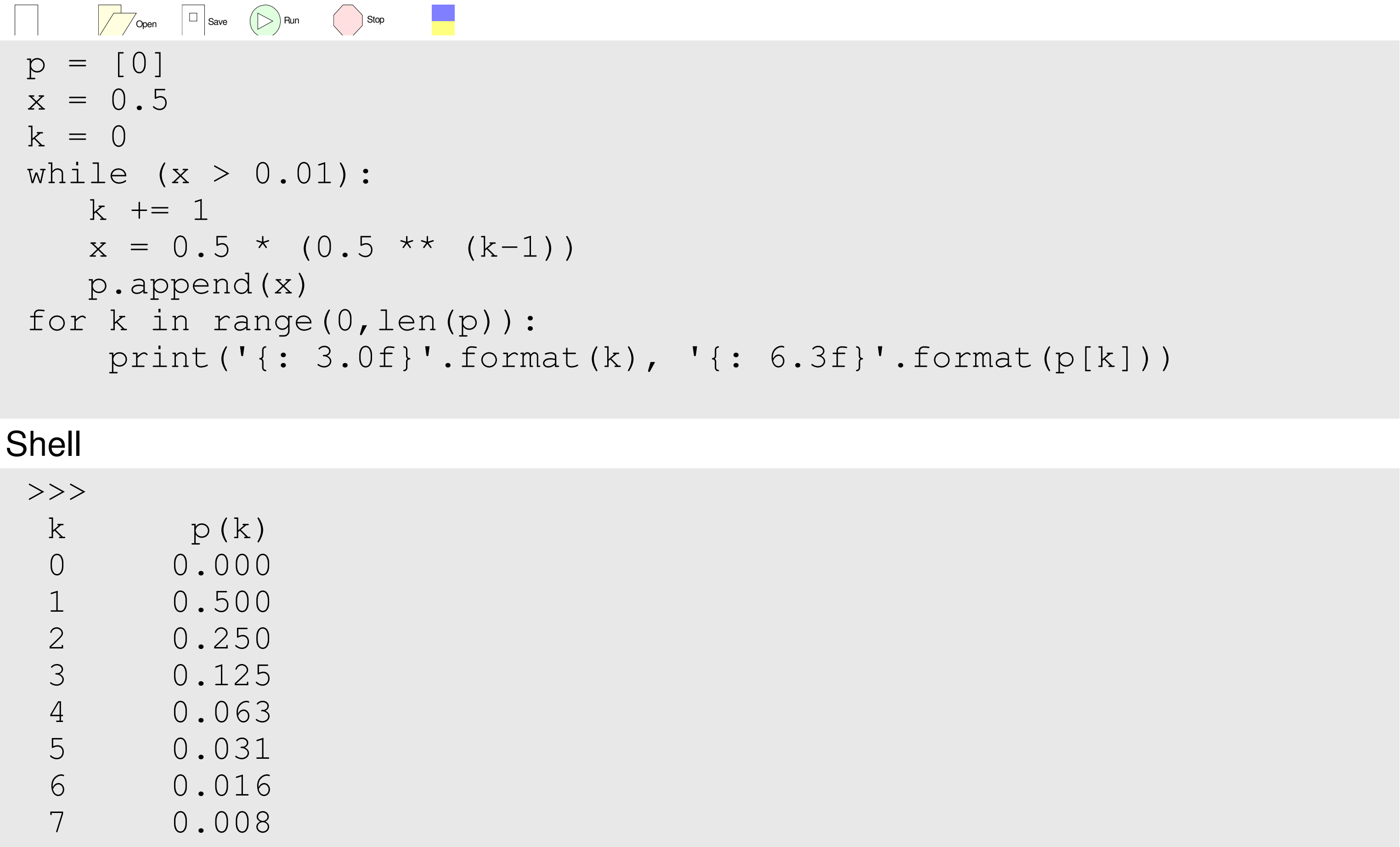


```
k = [0]
p = [0]
for i in range(1,11):
   k.append(i)
   p.append(0.5 * ( 0.5 ** (i-1) )
print('  k    p(k)')
for i in range(0,11):
    print('{: 3.0f}'.format(k[i]), '{: 6.3f}'.format(p[i]))
```

Shell

```
>>>
 k      p(k)
 0     0.000
 1     0.500
 2     0.250
 3     0.125
 4     0.063
 5     0.031
 6     0.016
 7     0.008
 8     0.004
 9     0.002
10     0.001
```

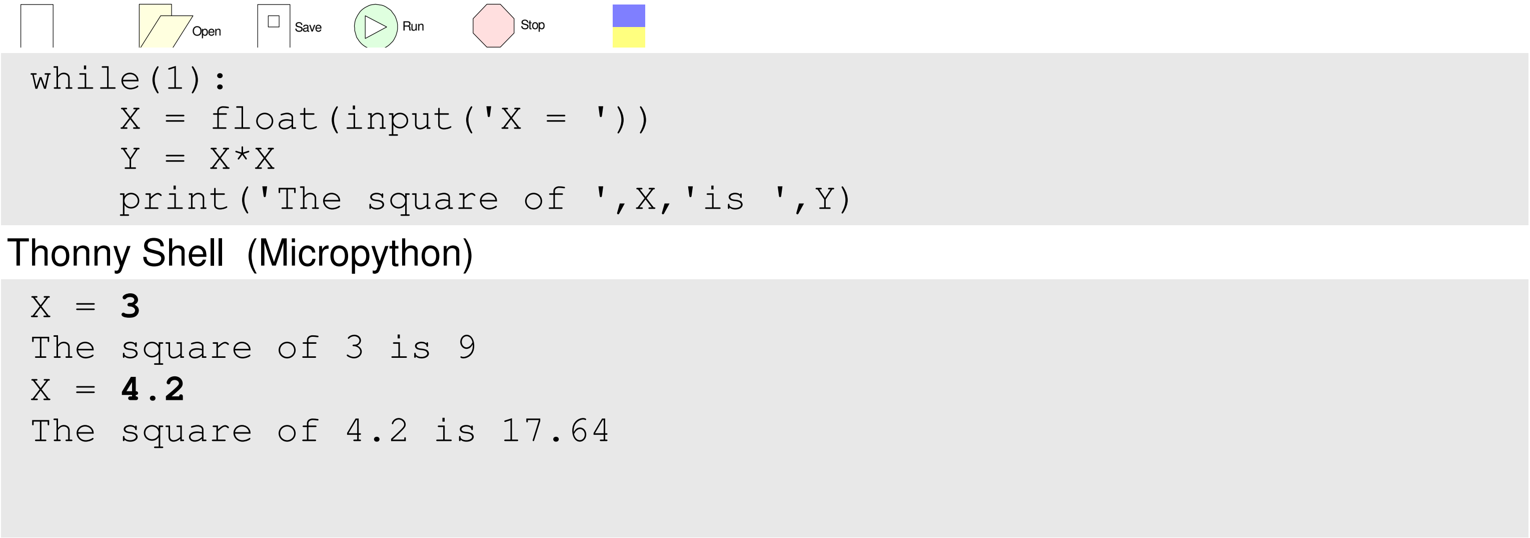
If you use a while loop, you can stop as soon as p(k) < 0.01



```
p = [0]
x = 0.5
k = 0
while (x > 0.01):
   k += 1
   x = 0.5 * (0.5 ** (k-1))
   p.append(x)
for k in range(0,len(p)):
    print('{: 3.0f}'.format(k), '{: 6.3f}'.format(p[k]))
```

```
>>>
 k      p(k)
 0     0.000
 1     0.500
 2     0.250
 3     0.125
 4     0.063
 5     0.031
 6     0.016
 7     0.008
```

Another common use of while statements is to set up an infinite loop



```
while(1):
    X = float(input('X = '))
    Y = X*X
    print('The square of ',X,'is ',Y)
```

Thonny Shell (Micropython)

```
X = 3
The square of 3 is 9
X = 4.2
The square of 4.2 is 17.64
```

Press the Stop symbol to break out of an infinite loop

If Statements

With if-statements

- If the condition is true, the indented section is executed one time,
- Otherwise it is skipped.

Conditional statements are:

```
X > Y     X is greater than Y
X < Y     X is less than Y
X >= Y    X is greater than or equal to Y
X == Y    X is equal to Y
X != Y    X is not equal to Y
&         logical and
|         logical or
^         logical xor
```

Indentation indicates the statements that are within the for loop.

```
if(x>y):
    print('x is greater than y')
if(x<y):
    print('x is less than y')
if(x==y):
    print('x is equal to y')
```

else, elif statements:

else indicates instructions to execute if the if-statement is false

```
if(x>y):
    print('x is greater than y')
else:
    print('x is less than or equal to y')
```

elif is an else-if statement

```
if(x>y):
    print('x is greater than y')
elif(x<y):
    print('x is less than y')
else:
    print('x is equal to y')
```

One place where else-if is useful is when you have different bands. For example, the following code is equivalent:

```
# Option 1
if(T>40):
    print('Really hot: T > 40')
if( (T>30)&(T<=40)):
    print('Hot: 30<T<40)')
if( (T>20)&(T<=30)):
    print('Comfortable: 20<T<30')
if( (T>10)&(T<=20)):
    print('Cool: 10<T<20')
```

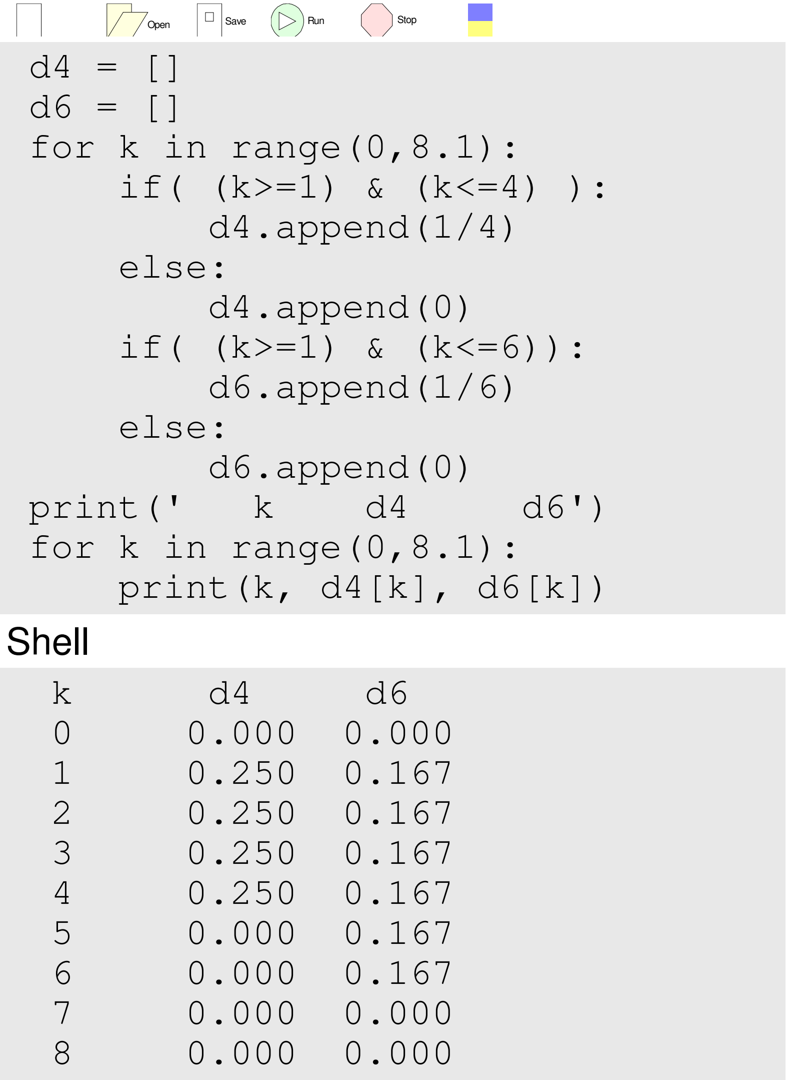
or using else-statements

```
# Option 2
if(T>40):
    print('Really hot: T > 40')
elif( T>30):
    print('Hot: 30<T<40)')
elif( T>20):
    print('Comfortable: 20<T<30')
elif( T>10):
    print('Cool: 10<T<20')
else:
    print('Chilly: T < 10')
```

If-Statements and Probability Density Functions

A more efficient way to create the pdf for a 4-sided and 6-side die:

- Use if-statments
- Along with append() statements



```
d4 = []
d6 = []
for k in range(0,8.1):
    if( (k>=1) & (k<=4) ):
        d4.append(1/4)
    else:
        d4.append(0)
    if( (k>=1) & (k<=6)):
        d6.append(1/6)
    else:
        d6.append(0)
print('    k     d4      d6')
for k in range(0,8.1):
    print(k, d4[k], d6[k])
```

```
 k      d4     d6
 0     0.000  0.000
 1     0.250  0.167
 2     0.250  0.167
 3     0.250  0.167
 4     0.250  0.167
 5     0.000  0.167
 6     0.000  0.167
 7     0.000  0.000
 8     0.000  0.000
```

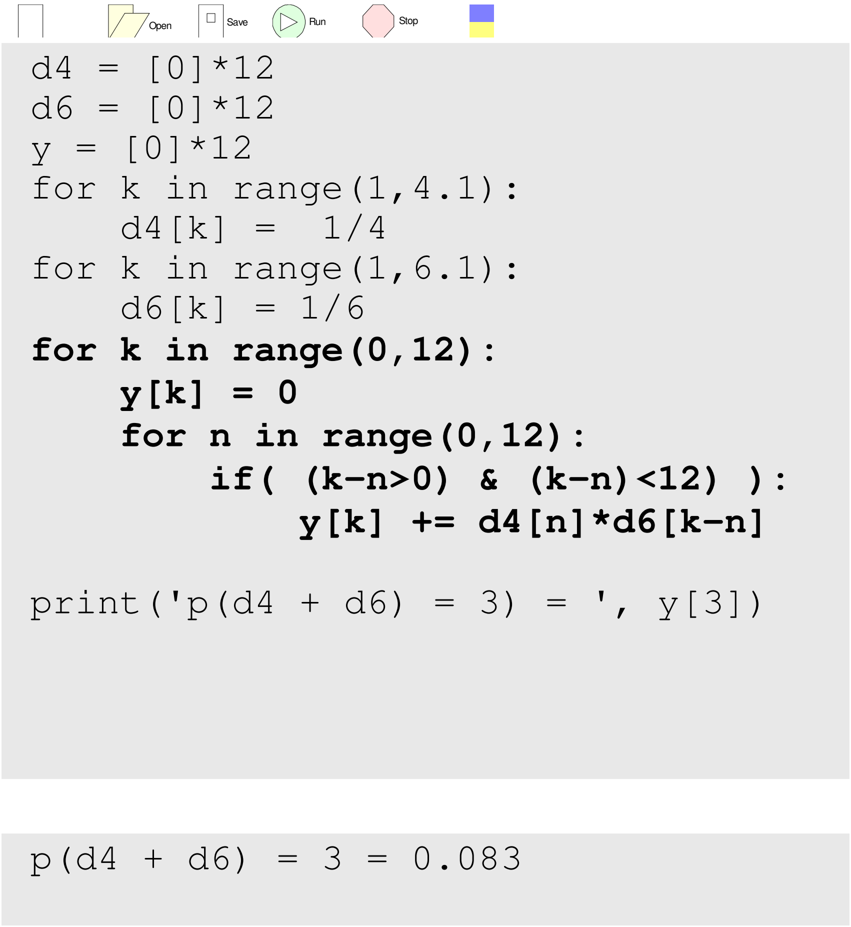
If-Statements & Convolution

Y = d4 + d6

When you add dice,

- You convolve the pdf's
- y[k] = sum(d4[n] * d6[k-n])

Convolution can be done with for-loops

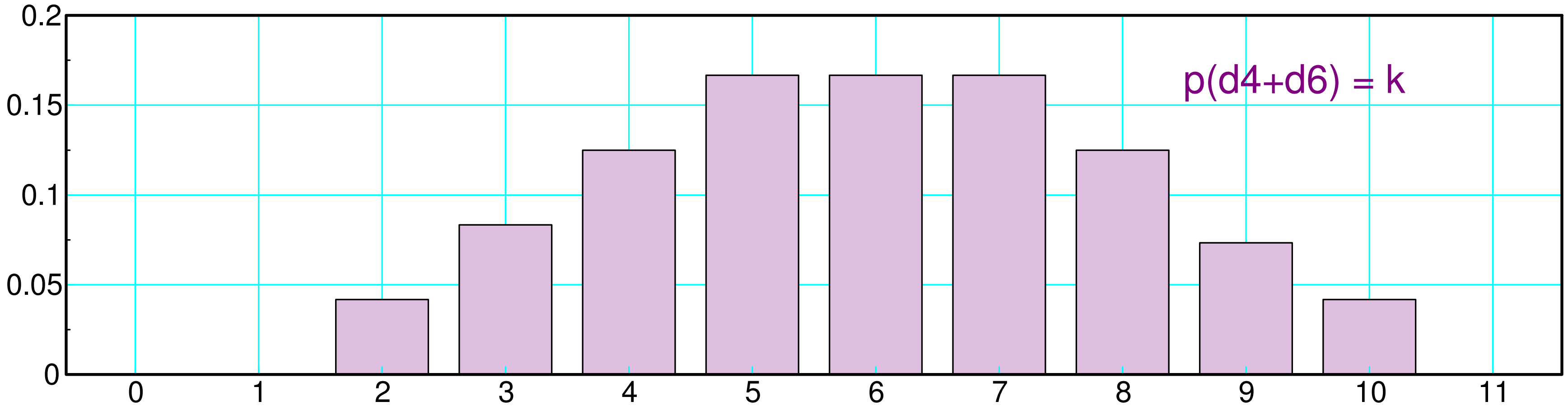


```
d4 = [0]*12
d6 = [0]*12
y = [0]*12
for k in range(1,4.1):
    d4[k] =  1/4
for k in range(1,6.1):
    d6[k] = 1/6
for k in range(0,12):
    y[k] = 0
    for n in range(0,12):
        if( (k-n>0) & (k-n)<12) ):
            y[k] += d4[n]*d6[k-n]

print('p(d4 + d6) = 3) = ', y[3])
```

```
p(d4 + d6) = 3 = 0.083
```

The probability of the sum of a d4 and d6 is 3 is 0.083



Summary

MicroPython is similar to Matlab

- MicroPython has for-loops
- It has while-loops
- It has if-statements

The syntax is slightly different

- MicroPython does not have *end* statements
- Instead, it uses indentation

Indentation is important

- It indicates which statements are part of a loop
- It tells you where the loop ends