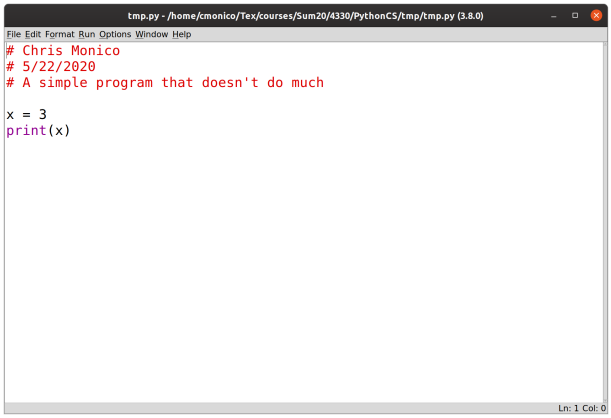
Python 'Cheat Sheet'
May 22, 2020, Chris Monico

This is just a very quick reference, with a couple of small examples illustrating each concept. Most of these are far more powerful than the examples illustrate, though!

IDLE shell window



IDLE editor Window

IDLE

Start a new program	in shell window, *File* → *New File*, or CTRL+N
Open existing program	in either window, *File* → *Open File*, or CTRL+O
Run program	in editor window, *Run* → *Run Module*, or F5

Comments

 # This comments out a single line
Or surround with triple-quotes, for multi-line comments.

Variables

Variable names are case-sensitive, and can contain upper and lower case letters, digits, and the underscore character. They may **not** start with a digit. The following are valid and different variable names:

```
N = 10
n = 2
my_str = 'Monty'
my_str2 = "Python"
x2 = 1.2917
```

Numeric Operators

For numbers, the basic arithmetic operators are exactly what you would expect: `+ − ∗ /`, and parentheses group expressions as you would expect. Two additional operators that are often useful are:

```
x = 3
y = 2
#(1) The modulus operator % to compute the remainder of x divided by y:
z = x % y
#(2) The exponentiation operator ** to compute x to the y power:
u = x**y
v = (x+y)**(0.5)
```

Printing

```
n=5
pi=3.141592653589
#Simple positional formatting:
#%d integer, %f float, %s string
print("n is %d and pi is about %1.5f"% (n,pi))
#The format method:
print("n is {0} and pi is about {1}".format(n,pi))
```

Input

```
#Prompt the user to enter a name
name = input("Enter a name: ")
#Prompt the user for an age, but convert to an int,
#in case we want to do arithmetic with it later.
age = int(input("Age: "))
print("Name: %s, Age: %d"%(name,age))
```

for loops

```
s=0
for n in range(4):
  s = s+n
print(s) #prints 6, since 0+1+2+3=6.
```

The above code is the same as:

```
print(sum(range(4)))
```

which is also the same as: `print(sum([0,1,2,3]))`

`while` loops

Suppose we want to find the least positive odd integer N for which $N^3 + 3N^2 > 1000$. We can check 1, 3, 5,... in order until we find one that works. A `for` loop is not an ideal choice, because we don't know exactly how many times we need to iterate.

```
#Find the smallest odd positive integer N for which
#N**3 + 3*N**2 > 1000,
N=1
#Note: this loop will terminate, because we know such an N exists.
while N**3 + 3*N**2 <= 1000:
  N += 2
print(N)
```