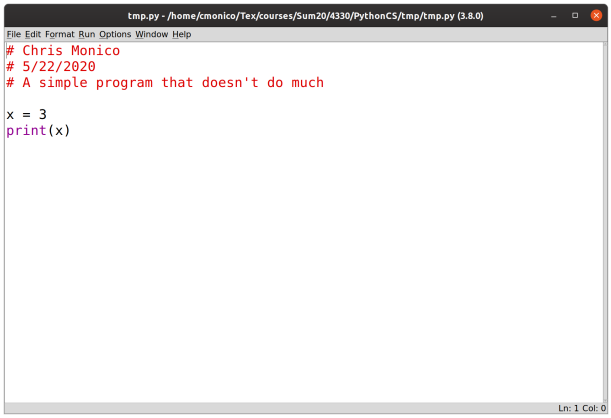
Python 'Cheat Sheet'
June 4, 2020, Chris Monico

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 #int
n = 2 #another int
my_str = 'Monty' #string
my_str2 = "Python" #string
x2 = 1.2917 #float
goodjob = True #bool
```

Numeric Operators

For numbers, the basic arithmetic operators are exactly what you would expect: `+ - * /`, and parentheses group expressions as you would expect. Three additional operators that are often useful are the modulus operator `%`, integer division operator `//`, and exponentiation operator `**`

```
x = 3
y = 2
#(1) The modulus operator % to compute the remainder of x divided by y:
r = x % y
#The integer division operator to return the integer quotient:
q = 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 `while` loop is a good 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)
```

Conditional statements

```
x = float(input("Enter a number: "))
if x>0:
  print("your number is positive")
elif x<0:
  print("your number is negative")
else:
  print("your number is zero.")
```

Note: since = is the assignmet operator, there is a different operator for testing equality:

```
if x == 0:
  print("x is zero.")
```

Functions

```
def sumofdigits(n):
    #Given a positive integer n,
    #return the sum of its digits.
    s = 0
    remaining = n
    while remaining > 0:
        s += (remaining%10) #Add the last digit to s
        remaining = remaining // 10 #Remove the last digit
    return s

k = 3915
#The function sumofdigits will be called with the argument 3915,
#and the value it returns will be substituted in place:
res = sumofdigits(k)
print("sum of the digits of {0} is {1}.".format(k, res))
#try to print(remaining) here and see what happens.
#The variable no longer exists, so it would be an error.
#This is an example of 'scope'.
```

Lists

```
my_list=[3,1,4,1,5,9,2,6]
print(my_list[0]) #prints the number 3
print(len(my_list)) #prints 8.
#range creates a list of integers in a given range.
print(range(4)) #prints [0,1,2,3]
print(range(1,4)) #prints [1,2,3]
```

List comprehension is like mathematical set-builder notation. Compare the following with the set $\{x^2 \ : \ x \in \mathbb{Z} \cap [0, 10)\}$:

```
squares = [x**2 for x in range(10)]
10 in squares #evaluates to False
16 in squares #evaluates to True
```

Other useful list methods include `insert`, `append`, `index`, `pop`.

Classes

Classes provide a way to create new data types and functions (methods) that can be applied to them. This class has 4 methods, and each object of the class will have two attributes, `l` and `w`.

```
class Rectangle:
  def __init__(self, length, width):
    # This function is called on a new object when it's created.
    self.l = length
    self.w = width
  def area(self):
    return self.l * self.w
  def perimeter(self):
    return 2*(self.l + self.w)
  def change_dimensions(self, length, width):
    self.l = length
    self.w = width

# Create, or instantiate, a Rectangle object with dimensions 3x5.
R = Rectangle(3, 5)
print("R has area {0}".format(R.area()))
print("R has perimeter {0}".format(R.perimeter()))
# Change it to be 5x2:
R.change_dimensions(5, 2)
print("R now has area {0}".format(R.area()))
```

Dictionaries

Like a list, but indexed by a key instead of the integers 0,1,2,...

```
plateowner={"ABC123":"Jack",
  "1FG80A":"Jill"}
print(plateowner["1FG80A"])
```

`max` and `min`

```
a=10
b=3.14159**2
c=2.71828**2
print(max(a,b,c))
print(min(a,b,c))
```

Using Libraries

```
import math
print(math.sqrt(7))
print(math.cos(3.1415926/2)) #Radians!
print(math.cos(math.pi/2))
```

Or, if we need only a single function or two:

```
from math import sqrt
print(sqrt(7))
```

To find out what's contained in a library,

```
dir(math)
```

or

```
help(math)
```

File I/O

```
filehandle = open("afile.txt", "r")
```

Second argument is `"r"` (read), `"w"` (write), or `"a"` (append).

```
wholefile = filehandle.read()
tenbytes = filehandle.read(10)
```

Or, read just one line:

```
thisline = filehandle.readline()
```

Or, read a list of lines:

```
theselines = filehandle.readlines()
```

Write a string to file with:

`filehandle.write("Hello file")`!

Or write a list of lines with the `writelines()` method.

IMPORTANT: When you're done with the file, call the `close()` method!

`filehandle.close()` Other methods: `tell()`, `seek()`