python 121 Cheat Sheet

by luckytime via cheatography.com/214087/cs/46599/

Files

```
file = open("data.txt") # open file
contents = file.read() # whole file as string
lines = file.readlines() # all lines as list
line = file.readline() # one line
for line in file: # loop lines
line = line.strip() # remove \n and spaces
parts = line.split(",") # split by comma
file.close() # close file
```

Class

```
"""A program that defines a Student class and then tests it"""
class Student: # Hint: How do you declare a class?
"""Defines a Student class.
Data attributes:
student_id : integer
usercode : string
first_name, family_name : strings
courses : set of course names(-strings)
Methods:
get_email_address()
enrol_course()
is_enrolled_in()
"""
```

Class (cont)

```
def __init__(self, student_id, usercode, first_name, family-_name ):
"""Creates a new Student object."""
self.student_id = student_id
self.usercode = usercode
self.first_name = first_name
self.family_name = family_name
self.courses = set()
def __str__(self):
"""Returns a printable version of a student."""
return f"{self.student_id} {self.f-amily_name.upper()}, {self.first_-name.capitalize()}"
def get_email_address(self):
"""Returns the student's email address."""
return f"{self.usercode}@ucli-ve.ac.nz"
def enrol_course(self, course):
"""Update the set of courses a student is enrolled in."""
self.courses.add(course)
def is_enrolled_in(self, course):
```

Class (cont)

```
"""Test to see if the course exists in the set of courses taken by the student."""
return course in self.courses
def main():
"""Main function tests the Student class"""
astudent = Student(1234567, "-ffn127", "first NAMES", "Family name") # Hint: Create an Student instance
print(type(astudent)) # Print its type
print(astudent) # Print the student's id and name
astudent.enrol_course("COSC1-21") # Hint: Enrol the student in the course "COSC121"
print(astudent.is_enrolled_in("-COSC121")) #Should print True
print(astudent.is_enrolled_in("-COSC122")) #Should print False
print(astudent.get_email_ad-dress() ) #Should print Student's email address
main()
```

list slicing

nums = [10, 20, 30, 40, 50, 60]	nums[1:4] [20, 30, 40]
nums[:3]. [10, 20, 30]	nums[-2:] last 2 items
nums[:-1] all but last	nums[::2] every 2nd item
nums[::-1]. reverse	

If statment

if num % 2 == 0: #even	if num % 2 != 0: #odd
6 / 2 #3.0	7 // 2 #3
7 % 2 #1	3 ** 2 #9

matplotlib

```
import matplotlib.pyplot as plt
import numpy as np
# Data
x = np.linspace(0, 10, 100)
y = np.sin(x)
# Basic Plot
plt.plot(x, y, label='Sine wave')
plt.scatter(x, y, s=10, color='red')
plt.bar([1, 2, 3], [3, 5, 2])
plt.hist(np.random.randn(1000), bins=20, alpha=0.5)
plt.pie([25, 35, 40], labels=['A', 'B', 'C'])
# Labels and Title
plt.title("Main Title") # Regular title
```

matplotlib (cont)

```
plt.title(r"$y = x^2 + 3x + 500$")
# LaTeX math formatted title
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.legend()
plt.grid(True)
plt.xlim(0, 10)
plt.ylim(-1, 1)
# Axes Object (Manual)
ax = plt.axes()
ax.plot(x, y, label="sin(x)")
ax.set_title(r"Axes Title: $y = \sin(x)$")
ax.set_xlabel("X label")
ax.set_ylabel("Y label")
ax.legend()
ax.grid(True)
# Multiple Manual Axes (Optional Advanced Use)
fig = plt.figure()
ax1 = fig.add_axes([0.1, 0.55, 0.35, 0.35]) # [left, bottom, width, height]
ax1.plot(x, y)
ax2 = fig.add_axes([0.6, 0.1, 0.35, 0.35])
ax2.scatter(x, y, color='purple')
# Save and Show
plt.savefig("my_plot.png")
plt.show()
```

For Loops

```
for i in range(5): print(i) # 0 1 2 3 4
for i in range(2,6): print(i) # 2 3 4 5
for i in range(1,10,3): print(i) # 1 4 7
lst=['a','b','c']
for i in range(len(lst)): print(i,lst[i]) # 0 a 1 b 2 c
for item in lst: print(item) # a b c
for i,item in enumerate(lst): print(i,item) # 0 a 1 b 2 c
```

switching while loop to for loop

```
# For loops
for op1 in range(0, value1):
for op2 in range(value1, value2, -1):
print(f"{op1:3} x {op2:3} = {op1*op2:3}")
# Equivalent while loops
op1 = 0
while op1 < value1:
op2 = value1
while op2 > value2:
print(f"{op1:3} x {op2:3} = {op1*op2:3}")
op2 -= 1
op1 += 1
```

Sets and Dictionaries

```
# Sets from list and operations
lst=[1,2,2,3,4,4]
s=set(lst) # {1,2,3,4}
s.add(5) # {1,2,3,4,5}
s.remove(2) # {1,3,4,5}
s.discard(6) # {1,3,4,5} no error
print(3 in s) # True
s2={3,4,5,6}
print(s.union(s2)) # {1,3,4,5,6}
print(s.intersection(s2))# {3,4,5}
print(s.difference(s2)) # {1}
print(s.issubset(s2)) # False
for x in s: print(x) # 1 3 4 5 (any order)
# Dictionaries and methods
d={"a":1,"b":2}
print(d.keys()) # dict_keys(['a','b'])
print(d.values()) # dict_values([1,2])
print(d.items()) # dict_items([('a',1),('b',2)])
print(d.get("a")) # 1
print(d.get("z",0)) # 0 default if missing
d["c"]=3
```

Sets and Dictionaries (cont)

```
for k in d: print(k) # a b c
for v in d.values(): print(v) # 1 2 3
for k,v in d.items(): print(k,v) # a 1 b 2 c 3
```

Lists

fruits = ["apple", "banana", "cherry"]	fruits[0] #apple
fruits[-1] #cherry	len(fruits) #3
fruits.append("kiwi")	fruits.insert(1, "pear")
fruits.remove("banana")	fruits.pop() #removes last
fruits.sort()	sorted(fruits)