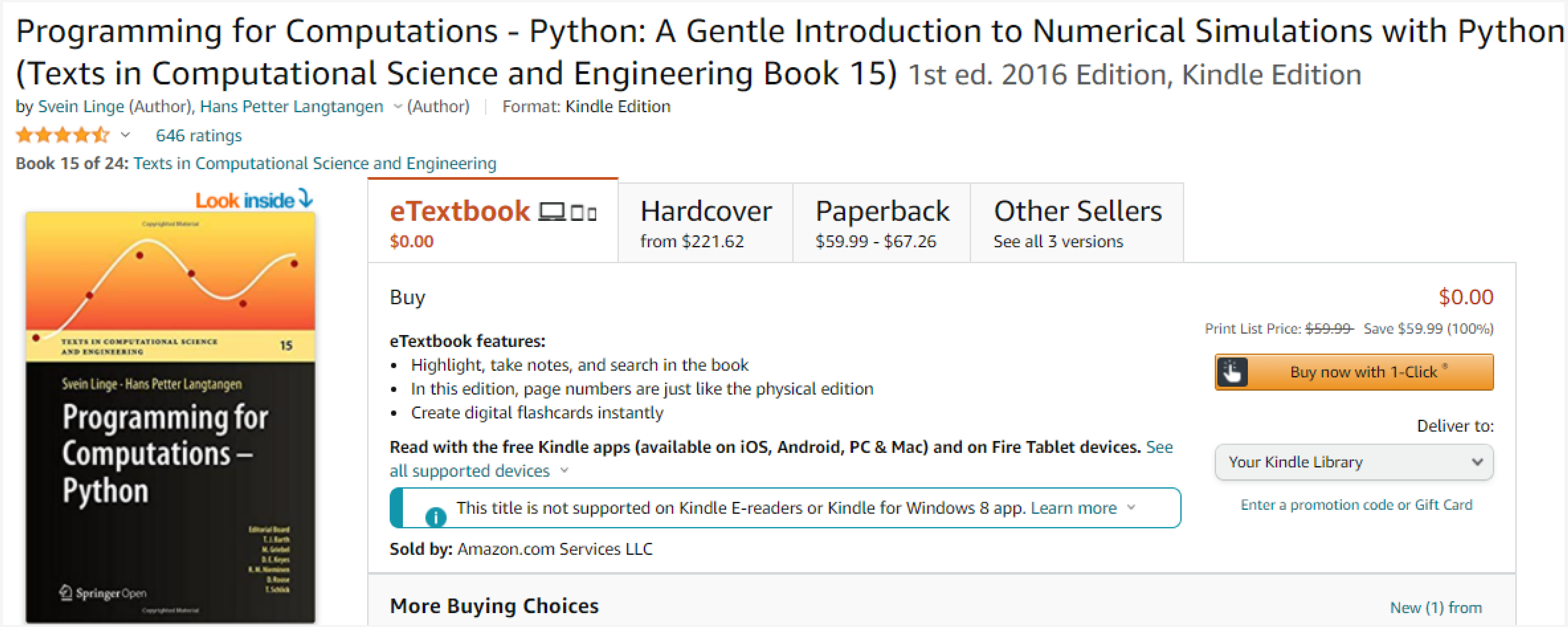
Learn Python as fast and as much as possible

- Free Python course on https://www.datacamp.com/
- There is a chapter in Python in the textbook
- DiveIntoPython.pdf & cheatsheets from course resources on Blackboard
  - https://www.datacamp.com/community/data-science-cheatsheets
  - https://ehmatthes.github.io/pcc/cheatsheets/README.html
- Great free book specifically for numerical computation
  - https://www.amazon.com/Programming-Computations-Introduction-Simulations-Computational-ebook-dp-B078YGVNSF/dp/B078YGVNSF/ref=mt_other?_encoding=UTF8&me=&qid=

Specific Python book for numerical computations



Programming for Computations - Python: A Gentle Introduction to Numerical Simulations with Python (Texts in Computational Science and Engineering Book 15) 1st ed. 2016 Edition, Kindle Edition

by Svein Linge (Author), Hans Petter Langtangen (Author) | Format: Kindle Edition

646 ratings

Book 15 of 24: Texts in Computational Science and Engineering

Look inside
TEXTS IN COMPUTATIONAL SCIENCE AND ENGINEERING
15
Svein Linge · Hans Petter Langtangen
Programming for Computations – Python
Springer Open

eTextbook $0.00 | Hardcover from $221.62 | Paperback $59.99 - $67.26 | Other Sellers See all 3 versions

Buy

$0.00

Print List Price: ~~$59.99~~ Save $59.99 (100%)

eTextbook features:

- Highlight, take notes, and search in the book
- In this edition, page numbers are just like the physical edition
- Create digital flashcards instantly

Read with the free Kindle apps (available on iOS, Android, PC & Mac) and on Fire Tablet devices. See all supported devices

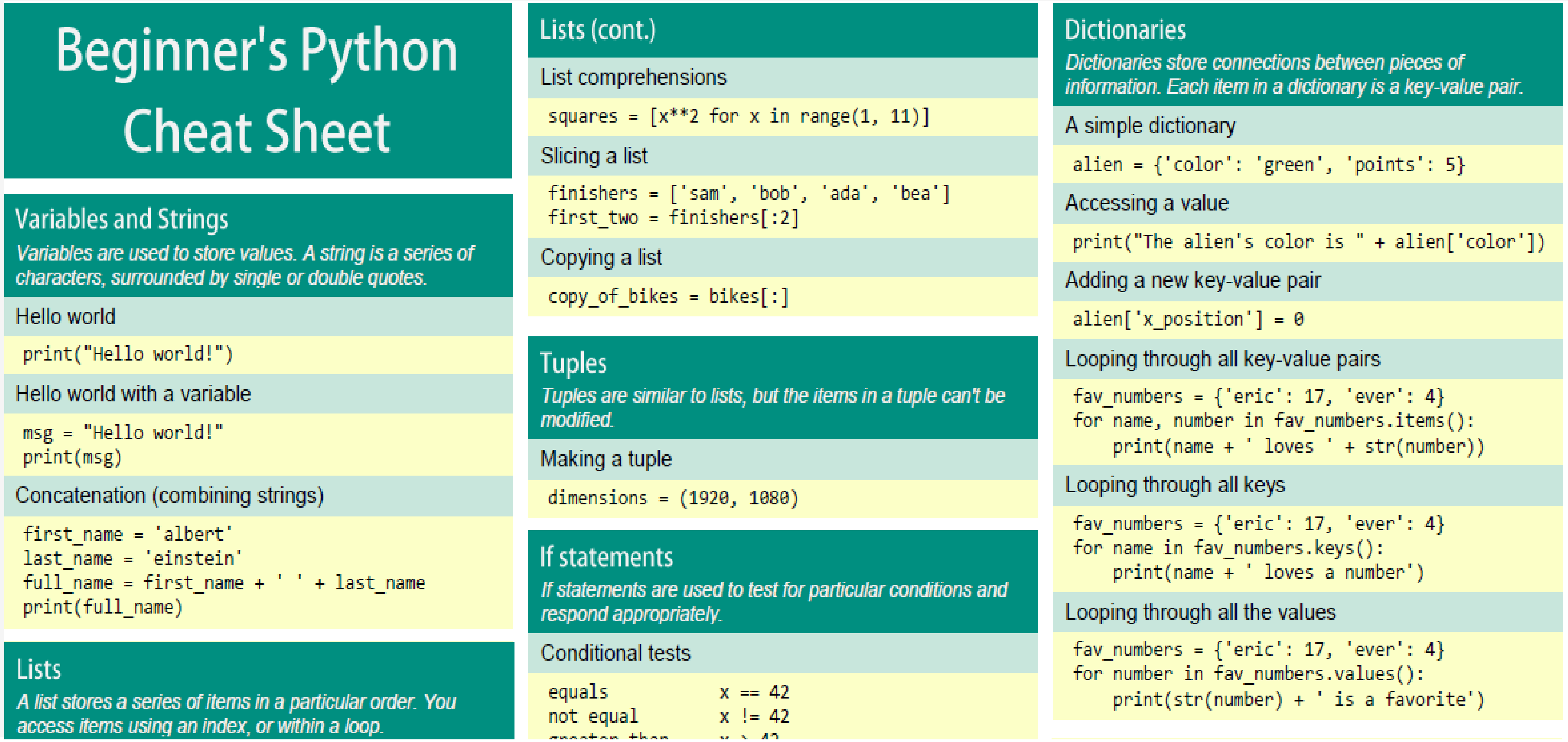
This title is not supported on Kindle E-readers or Kindle for Windows 8 app. Learn more

Sold by: Amazon.com Services LLC

More Buying Choices

New (1) from

Python CheatSheet



Beginner's Python Cheat Sheet

Variables and Strings

Variables are used to store values. A string is a series of characters, surrounded by single or double quotes.

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"
print(msg)
```

Concatenation (combining strings)

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

Lists

A list stores a series of items in a particular order. You access items using an index, or within a loop.

Lists (cont.)

List comprehensions

```
squares = [x**2 for x in range(1, 11)]
```

Slicing a list

```
finishers = ['sam', 'bob', 'ada', 'bea']
first_two = finishers[:2]
```

Copying a list

```
copy_of_bikes = bikes[:]
```

Tuples

Tuples are similar to lists, but the items in a tuple can't be modified.

Making a tuple

```
dimensions = (1920, 1080)
```

If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

```
equals           x == 42
not equal        x != 42
```

Dictionaries

Dictionaries store connections between pieces of information. Each item in a dictionary is a key-value pair.

A simple dictionary

```
alien = {'color': 'green', 'points': 5}
```

Accessing a value

```
print("The alien's color is " + alien['color'])
```

Adding a new key-value pair

```
alien['x_position'] = 0
```

Looping through all key-value pairs

```
fav_numbers = {'eric': 17, 'ever': 4}
for name, number in fav_numbers.items():
    print(name + ' loves ' + str(number))
```

Looping through all keys

```
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(name + ' loves a number')
```

Looping through all the values

```
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(str(number) + ' is a favorite')
```

Python CheatSheet

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

Lists

A list stores a series of items in a particular order. You access items using an index, or within a loop.

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item in a list

```
first_bike = bikes[0]
```

Get the last item in a list

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:
    print(bike)
```

Adding items to a list

```
bikes = []
bikes.append('trek')
bikes.append('redline')
bikes.append('giant')
```

Making numerical lists

```
squares = []
for x in range(1, 11):
    squares.append(x**2)
```

If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

```
equals              x == 42
not equal           x != 42
greater than        x > 42
  or equal to       x >= 42
less than           x < 42
  or equal to       x <= 42
```

Conditional test with lists

```
'trek' in bikes
'surly' not in bikes
```

Assigning boolean values

```
game_active = True
can_edit = False
```

A simple if test

```
if age >= 18:
    print("You can vote!")
```

If-elif-else statements

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15
```

```
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(name + ' loves a number')
```

Looping through all the values

```
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(str(number) + ' is a favorite')
```

User input

Your programs can prompt the user for input. All input is stored as a string.

Prompting for a value

```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

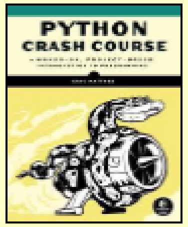
Prompting for numerical input

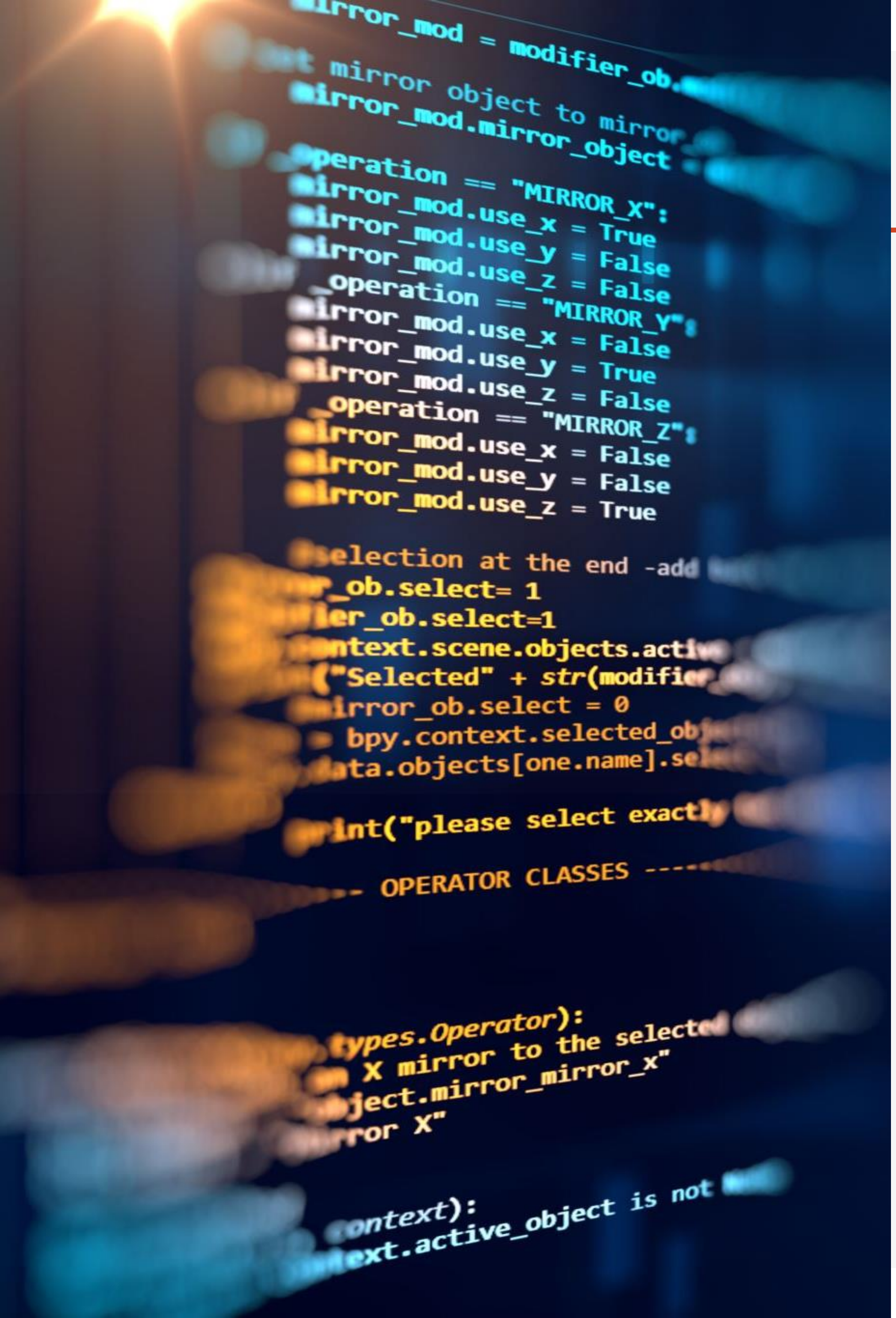
```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```

Python Crash Course

Covers Python 3 and Python 2

nostarchpress.com/pythoncrashcourse





```
irror_mod = modifier_ob.
 mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
 _operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
 _operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

election at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.activ
("Selected" + str(modifie
irror_ob.select = 0
 bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactl

- OPERATOR CLASSES ----

types.Operator):
 X mirror to the selected
ject.mirror_mirror_x"
rror X"

context):
ext.active_object is not
```

Python core concepts checklist

- How to create a class
- Distinguish between class methods versus instance method
- Familiar with the various built-in data type, know how to parse a date string into a date object
- Know what is a list, and how to loop through each element in the list, know list comprehension
- …

.

Learning by doing