Introduction to Python

Getting started with the basics plus a few advanced concepts

P. Moggridge

July 14, 2023

Outline

Learning Objectives

The goal of this session is to give you enough knowledge to enable you to build simple programs so that you can continue to learn independently.

Knowledge:

- Knowledge of the basic concept of programming.
- Comprehend the syntax and patterns in Python.

Skills:

- Be able to code and execute a small Python program.
- Be able fix bugs.

Why learn to code

1. **Coding is fun!** Learning to code can be a challenging but rewarding experience.
2. **Coding is very important for computer science students.** It is one of the gateway skills that unlocks deeper understanding across the field.
3. **Coding teaches problem-solving skills.** Coding requires you to break down problems into smaller, more manageable steps.
4. **Coding teaches critical thinking skills.** Coding requires you to think logically and to come up with creative solutions to problems.
5. **Coding unlocks a new world of creativity.** You can create games, visualisations, music, useful tools and solve problems which might seem intractable to people without coding ability.
6. **Coding only requires a computer and electricity.** (*really any old computer will do, a common misconception that you need a powerful machine for coding*)

Why learn Python

1. **Python is easy to learn.** Python has a simple syntax that is easy to understand, even for beginners. This makes it a great language for students who are new to programming.
2. **Python is versatile.** Python can be used for a wide variety of tasks, including web development, data science, machine learning, and artificial intelligence. This makes it a valuable skill for computer science students to have.
3. **Python is in demand.** Python is one of the most popular programming languages in the world, and the demand for Python developers is growing. This means that students who learn Python will be well-positioned for jobs in the tech industry.
4. **Python is open source.** Python is an open-source language, which means that it is free to use and modify. This makes it a great language for students who want to learn how to contribute to open-source projects.
5. **Python has a large community.** Python has a large and active community of users and developers. This means that students who learn Python will have access to a wealth of resources, including tutorials, documentation, and support forums.

Installing Python and Visual Studio Code (Windows 8, 10 & 11)

Type “python” into the Microsoft Store app. Click install.

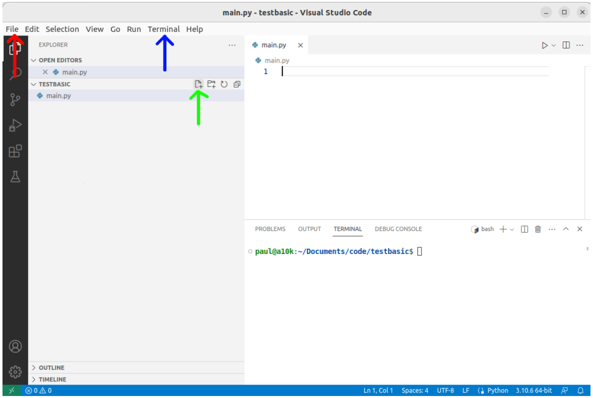
Type “visual studio code” into the Microsoft Store app. Click install.

Installing Visual Studio Code (Ubuntu Linux)

Type “visual studio code” into Ubuntu Software. Click install.

Ubuntu comes with Python3 out-of-the-box, no need to install it.

Creating a Basic Project with Visual Studio Project


main.py - testbasic - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
main.py
TESTBASIC
main.py
OUTLINE
TIMELINE
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
bash
Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit

```
paul@a10k:~/Documents/code/testbasic$
```

File > Open Folder (New Folder). New file icon, type "main.py". Terminal > New Terminal.

Hello World (Command Line Output)

Into the the code area:

```
print("Hello World")
```

After saving, into the terminal area type *python3 main.py* OR *python main.py*

Output

```
Hello World
```

Hello World (Command Line Input)

```
name = input('what is your name?')
print('Hello world, ' + name)
```

Output

what is your name > *Paul*
Hello world, Paul

Variables

```
target_planet = "Venus"
arrived = False
shuttle_fuel = 251.1
passengers = 10

# print out all the variables
print(target_planet)
print(arrived)
print(shuttle_fuel)
print(passengers)

print(type(target_planet)) # printing the data type of target_planet
print(type(arrived))
print(type(shuttle_fuel))
print(type(passengers))
```

Output

```
Venus
False
251.1
10
<class 'str'>
<class 'bool'>
<class 'float'>
<class 'int'>
```

Mathematical Operators

```python
print(10 + 10)
print(10 - 10)
print(10 * 10)
print(87 / 10)
print(87 // 10)
print(87 % 10)
print(10 ** 10)
print(abs(-10))
print(round(10.51))
```

Output

```
20
0
100
8.7
8
7
10000000000
10
11
```

Conditional Statements

```
print(1 > 1)
print(1 < 2)
print(1 <= 1)
print(2 >= 10)
print(2 == 2)
print(target_planet == "Mars")
```

Output

```
False
True
True
False
True
False
```

If elif else

```
if passengers == 10:
    print("All passengers present")

if target_planet == "Mars":
    print("Mission is proceding as planned")
else:
    print("Mission has deviated for the plan")

if shuttle_fuel > 200:
    print("Fuel level OK")
elif shuttle_fuel > 0:
    print("Fuel level LOW!")
else:
    print("Fuel deplated")
```

Output

```
All passengers present
Mission has deviated for the plan
Fuel level OK
```

Logical Operators

```
print("AND")
print(False and False)
print(False and True)
print(True and False)
print(True and True)
print("OR")
print(False or False)
print(False or True)
print(True or False)
print(True or True)
print("NOT")
print(not False)
print(not True)
```

Output

```
AND
False
False
False
True
OR
False
True
True
True
NOT
True
False
```

Loops & Ctrl-C

```python
i = 0
while i < 4:
    print("Pressure level " + str(i))
    i = i + 1
print("Airlock pressurised")

for item in ['ship', 'lander', 'rover', 'boosters']:
    print(item)

for i in range(3):
    print(i)
```

Output

```
Pressure level 0
Pressure level 1
Pressure level 2
Pressure level 3
Airlock pressurised
ship
lander
rover
boosters
0
1
2
```

Importing Modules

```
import random

print(random.randint(3, 9))
print(random.random())
print(random.random() < 0.7)
```

Output

```
3
0.2373448449931269
True
```

List

```python
space_tools = ["spanner", "pouch", "laser", "hammer", "laptop", "notepad"]

print(space_tools[3])

# slicing
print(space_tools[:3]) # everything before position 3
print(space_tools[3:]) # 3 and everything after
print(space_tools[:-1]) # everything before position -1 (loops around)

for tool in space_tools:
    print("Space " + tool)
```

Output

```
hammer
['spanner', 'pouch', 'laser' ]
['hammer', 'laptop', 'notepad' ]
['spanner', 'pouch', 'laser', 'hammer', 'laptop']
Space spanner
Space pouch
Space laser
Space hammer
Space laptop
Space notepad
```

Strings

```
captains_log = "Day 30 on the martian surface, the water is running low"

print(len(captains_log))
print(captains_log[1])
print(captains_log + 'est ever')
print(captains_log * 2)

print(captains_log.split(" "))
print(captains_log.split(" ")[0]) # the split function returns a list

print(captains_log.replace("the","silly"))
```

Output

```
55
a
Day 30 on the martian surface, the water is running lowest ever
Day 30 on the martian surface, the water is running lowDay 30 on the martian surface, the water is
running low
['Day', '30', 'on', 'the', 'martian', 'surface,', 'the', 'water', 'is', 'running', 'low']
Day
Day 30 on silly martian surface, silly water is running low
```

Dictionary

```
water_tanks = { "drinking":44.54, "coolant":13.5, "waste":120.2}
water_tanks["aquaculture"] = 58.5
print(water_tanks["waste"])
print(water_tanks["aquaculture"])
```

Output

```
120.2
58.5
```

File Handling (Reading)

File "mars_samples.csv"

```
Site,Day,Weight(g)
A,3,34
B,3,40
A,3,34
C,3,50
D,5,150
```

```python
for line in open("mars_samples.csv","r"):
    cells = line.strip("\n").split(",")
    print(cells[0] + '\t' + cells[2])
```

Output

```
Site Weight(g)
A 34
B 40
A 34
C 50
D 150
```

File Handling (Appending)

```
samples_file = open("mars_samples.csv","a")
samples_file.write("A,6,45\n")
samples_file.write("B,6,110\n")
samples_file.write("B,7,41\n")
samples_file.close()
```

Output

```
(none)
```

File “mars_samples.csv”

```
Site,Day,Weight(g)
A,3,34
B,3,40
A,3,34
C,3,50
D,5,150
A,6,45
B,6,110
B,7,41
```

File Handling (Appending)

```
samples_file = open("mars_samples.csv","a")
samples_file.write("A,6,45\n")
samples_file.write("B,6,110\n")
samples_file.write("B,7,41\n")
samples_file.close()
```

Output

(none)

File “mars_samples.csv”

```
Site,Day,Weight(g)
A,3,34
B,3,40
A,3,34
C,3,50
D,5,150
A,6,45
B,6,110
B,7,41
```

Exceptions

```
astronauts = 10
rations = 0
escape_pod = ["alpha", "beta", "charlie"]

try:
    print(astronauts / rations)
except:
    print("Something went wrong!")

try:
    print(astronauts / rations)
    print(escape_pod[3])
except(ZeroDivisionError):
    print("Oh no, a ZeroDivisionError!")
except(IndexError):
    print("Oh no, a IndexError")

try:
    if astronauts > 5:
        raise Exception("Maximum number of logins exceeded!", astronauts)
except Exception as ex:
    print(ex.args)
```

Output

```
Something went wrong!
Oh no, a ZeroDivisionError!
('Maximum number of logins exceeded!', 10)
```

Functions

```python
# Approximates pi using the Madhava Leibniz formula
def aprox_pi(accuracy):
    pi = 0
    for i in range(accuracy):
        denominator = 1 + (i * 2) # denominator in Madhava Leibniz series 1,3,5,7...
        if i % 2 == 0: # Alternate between addition and subtraction
            pi = pi + (1 / denominator)
        else:
            pi = pi - (1 / denominator)
    return pi * 4

print(aprox_pi(1))
print(aprox_pi(10))
print(aprox_pi(100))
print(aprox_pi(1000))
print(aprox_pi(10000))
print(aprox_pi(100000))
```

Output

```
4.0
3.0418396189294032
3.1315929035585537
3.140592653839794
3.1414926535900345
3.1415826535897198
```

Classes (Definition)

```
class Ship:
    # Constructor - creates a new instance of ship
    def __init__(self, make, model, payload):
        self.make = make
        self.model = model
        self.payload = payload
        self.fuel = 100

    def get_info(self):
        return self.make + ' ' + self.model + ' - carrying ' + self.payload

    def take_off(self):
        self.fuel = 0
        self.payload = 'nothing!'

    def has_fuel(self):
        return self.fuel > 0
```

Classes (Usage)

```
my_ship = Ship('ESA','Vega','Pathfinder Satellite')
print(my_ship.get_info())
print(my_ship.has_fuel())
my_ship.take_off()
print(my_ship.get_info())
print(my_ship.has_fuel())
```

Output

```
ESA Vega - carrying Pathfinder Satellite
True
ESA Vega - carrying nothing!
False
```

Modules (Creation)

```
# mathslib.py
name = "MathsLib"
version = "1.0.0.1 alpha"
def factorial(n):
    ...
```

```
# picode.py
import math
def archemides_pi(sides):
    ...
def leibniz_pi(terms):
    ...
def python_pi():
    return math.pi
```

```
# __init__.py
__all__ = ["mathslib", "picode"]
```

File Structure

```
.
├── main.py
└── MathsLib
    ├── __init__.py
    ├── mathslib.py
    └── picode.py
```

Modules (Usage)

```python
from MathsLib import *

def main():
    print(mathslib.name + " " + mathslib.version)
    print(mathslib.factorial(8))
    print(picode.archemides_pi(100))
    print(picode.leibniz_pi(100))
    print(picode.python_pi())

if __name__ == '__main__':
    main()
```

Output

```
MathsLib 1.0.0.1 alpha
40320
3.141075907812829
3.121594652591011
3.141592653589793
```

Threads

```
import time, random, threading

def some_function(index):
    print('Hello from ' + str(index) + '!')
    time.sleep(random.randint(3,5))

my_threads = [None] * 5
for i in range(5):
    my_threads[i] = threading.Thread(target=some_function, args=[i])
    my_threads[i].start()
print('All threads have been started...')

for i in range(5):
    my_threads[i].join()
    print('Thread ' + str(i) + ' finished')
```

Output

```
Hello from 0!
Hello from 1!
Hello from 2!
Hello from 3!
Hello from 4!
All threads have been started...
Thread 0 finished
Thread 1 finished
Thread 2 finished
Thread 3 finished
Thread 4 finished
```

Practice makes perfect...

This is just a start of your journey! Now that you have the basic syntax, you can combine what you have learnt today in different ways and start making small programs.

You can learn so much by practising your skills undertaking small projects that are useful/interesting to you.

A this stage try building a text-based game or a utility (to automate a simple task on your computer such as transforming a file or calculating something for you.)

Coding Challenge

Can you write a program which...

1. prints "– Silly Sentence Generator –"
2. asks the user to input a sensible sentence
3. if the sentence is longer than 50 characters, output a well done message.
4. asks the user for a word unrelated to their sentence
5. creates an empty variable to accumulate the new sentence in
6. loops over the words in their sentence and...
7. ... with a random chance, adds the original word or the silly word to the new sentence.
8. prints the new sentence

Example Output

```
– Silly Sentence Generator –
Please input a sensible sentence > I need to go to work
Please input a word unrelated to your sentence > chickens
I need to go to chickens
```

Bug Fix Challenge

There are 3 bugs in this code, can you find and fix them?

```python
import math
def archemides_pi(sides):
    segment_angle = 360 / 0
    # We set the "radius" of our regular polgon
    inner_sides = 0.5
    # Use trigonmetry to find the length of the side we do not know...
    missing_outer_side = 2 * inner_sides * math.sin(math.radians(inner_sides * seg_angle))
    # Multiply to find the "circumference" of a our regular polygon
    circumference = missing_outer_side * sides
    diameter = (inner_sides * 2)
return circumference / diameter

print(archemides_pi(100))
```

Try to run the code, Python's error messages will help you.

Learning Objectives

The goal of this session is to give you enough knowledge to enable you to build simple programs so that you can continue to learn independently.

Knowledge:

- Knowledge of the basic concept of programming.
- Comprehend the syntax and patterns in Python.

Skills:

- Be able to code and execute a small Python program.
- Be able fix bugs.

Thank You

- Hopefully, you have enjoyed this brief introduction to python programming and find it useful in future! Programming is such an important skill to a computer scientist be sure to practice and enjoy it. Thank you for your attention!

Presentation created using LaTeX beamer.

What to learn next...

- Environments
- Graphical User Interfaces (Tk is good place to start)
- Multiprocessing
- Assertions and Unit Testing

Further Reading

📕 Cody Jackson
Learn Programming in Python with Cody Jackson: Grasp the Basics of Programming and Python Syntax While Building Real-World Applications
Packt Publishing Ltd, 2018

📕 Kent D. Lee
Python Programming Fundamentals
Springer, 2014

Coding Challenge Solution

```python
import random
print("-- Silly Sentence Generator --")
sentence = input("Please input a sensible sentence > ")
if len(sentence) > 50:
    print("well done that is a long sentence")
silly_word = input("Please input a word unrelated to your sentence > ")
new_sentence = ""
for word in sentence.split(" "):
    if random.random() < 0.3:
        new_sentence = new_sentence + silly_word + " "
    else:
        new_sentence = new_sentence + word + " "
print(new_sentence)
```

Bug Fix Challenge Solution

```python
import math
def archemides_pi(sides):
    segment_angle = 360 / sides
    # We set the "radius" of our regular polgon
    inner_sides = 0.5
    # Use trigonmetry to find the length of the side we do not know...
    missing_outer_side = 2 * inner_sides * math.sin(math.radians(inner_sides * segment_angle))
    # Multiply to find the "circumference" of a our regular polygon
    circumference = missing_outer_side * sides
    diameter = (inner_sides * 2)
    return circumference / diameter

print(archemides_pi(100))
```

Output (once fixed)

```
3.141075907812829
```