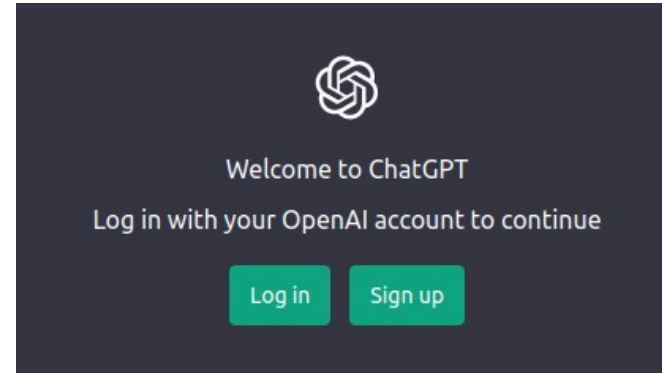
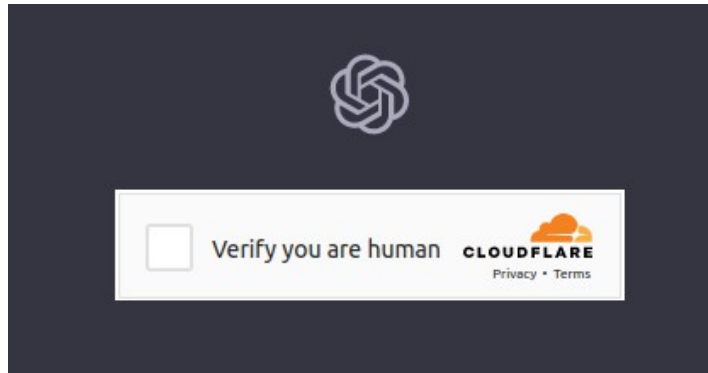


Introduction to Using ChatGPT to write Python code

Hamilton Python Users Group
Ian Stewart
13 February 2023

What is ChatGPT?

- **Chat** Generative **P**re-trained **T**ransformer = “chatbot”
- Chatbot = a software application used to conduct an online chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent.
- OpenAI Inc / OpenAI Ltd Partnership - Founded 2015.
- ChatGPT Launched by OpenAI in November 2022
- OpenAI website: <https://openai.com/>
- OpenAI ChatGPT website: <https://chat.openai.com/chat>
- Play ground: <https://platform.openai.com/playground>



ChatGPT Playground

<https://platform.openai.com/playground>

invent a name for a flavor of icecream

Cherry Chocolate Ripple Delight

invent a name for a flavor of green colored icecream

Minty Meadow Breeze

How high is Mount Everest

Mount Everest is the highest mountain on Earth, with an elevation of 8,848 meters (29,029 feet) above sea level.

Playground – Clock program 1st time - Python

<https://platform.openai.com/playground>

mples Playground

Playground

write a clock program for a computer

Submit



```
import time
```

```
while True:
```

```
    currentTime = time.strftime("%H:%M:%S")
```

```
    print(currentTime)
```

```
    time.sleep(1)
```

```
1 # ChatGPT response from:
2 # write a clock computer program
3
4 import time
5
6 while True:
7     currentTime = time.strftime("%H:%M:%S")
8     print(currentTime)
9     time.sleep(1)
```

```
$ python clock.py
```

```
18:38:47
```

```
18:38:48
```

```
18:38:49
```

```
18:38:50
```

Playground - Clock program 2nd time - C++

mples Playground

Playground

Not: write a clock program for a computer

write a clock computer program

```
#include <iostream>
#include <ctime>
#include <string>

using namespace std;

int main(){
    // Get the current time
    time_t now = time(0);
    tm* ltm = localtime(&now);

    // Output the current time
    cout << "The current time is " << ltm->tm_hour << ":";
    cout << ltm->tm_min << ":" << ltm->tm_sec << endl;

    return 0;
}
```

```
1 /* write a clock computer program */
2
3 #include <iostream>
4 #include <ctime>
5 #include <string>
6
7 using namespace std;
8
9 int main(){
10     // Get the current time
11     time_t now = time(0);
12     tm* ltm = localtime(&now);
13
14     // Output the current time
15     cout << "The current time is " << ltm->tm_hour << ":";
16     cout << ltm->tm_min << ":" << ltm->tm_sec << endl;
17
18     return 0;
19 }
```

\$ g++ clock.cpp -o clock

\$./clock

The current time is 18:30:36

\$./clock

The current time is 18:30:39

Playground – Requesting Python and GUI

<https://platform.openai.com/playground>



mples [Playground](#)

Playground

write a gui clock program in python

Submit



Playground – Python code with Tkinter GUI

```
import tkinter
```

```
import time
```

```
# Create a window
```

```
root = tkinter.Tk()
```

```
# Set window title
```

```
root.title("Clock")
```

```
# Set window size
```

```
root.geometry("350x200")
```

```
# Create a label for clock
```

```
clock = tkinter.Label(root, font = ('times', 20, 'bold'), bg = 'green')
```

```
# Place the label at a particular location
```

```
clock.place(relx = 0.5, rely = 0.5, anchor = tkinter.CENTER)
```

```
# Function used to update the time display as time changes
```

```
def update_time():
```

```
    current_time = time.strftime("%H:%M:%S")
```

```
    clock.configure(text = current_time)
```

```
    clock.after(200, update_time)
```

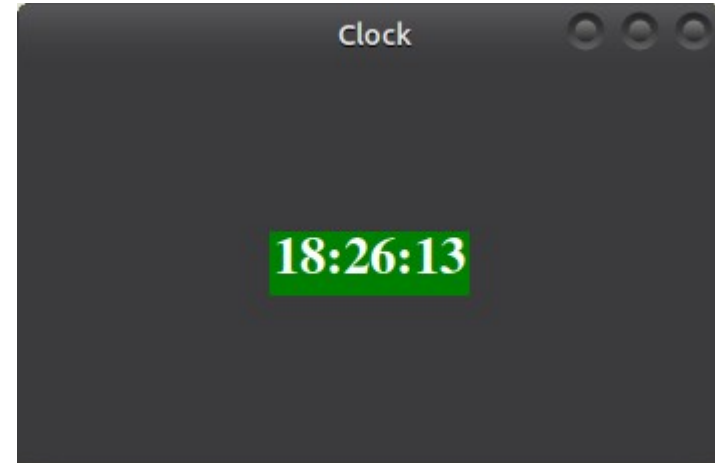
```
update_time()
```

```
# Run the main loop
```

```
root.mainloop()
```

Playground – Python code with Tkinter GUI

```
1 #write a gui clock program in python
2
3 import tkinter
4 import time
5 # Create a window
6 root = tkinter.Tk()
7 # Set window title
8 root.title("Clock")
9 # Set window size
10 root.geometry("350x200")
11 # Create a label for clock
12 clock = tkinter.Label(root, font = ('times', 20, 'bold'), bg = 'green')
13 # Place the label at a particular location
14 clock.place(relx = 0.5, rely = 0.5, anchor = tkinter.CENTER)
15 # Function used to update the time display as time changes
16 def update_time():
17     current_time = time.strftime("%H:%M:%S")
18     clock.configure(text = current_time)
19     clock.after(200, update_time)
20
21 update_time()
22 # Run the main loop
23 root.mainloop()
```



Playground – Slight variation – Still Python/Tkinter

write a gui clock program for a computer

Not: write a gui clock program in python

```
import tkinter
```

```
from time import strftime
```

```
#Create a window
```

```
window = tkinter.Tk()
```

```
window.title("Digital Clock")
```

```
window.geometry("400x100")
```

```
#Create a label to display time
```

```
clock = tkinter.Label(window, font=("times", 30, "bold"))
```

```
clock.grid(row=1, column=1, padx=5, pady=5)
```

```
def tick():
```

```
    time2 = strftime('%H:%M:%S %p')
```

```
    clock.config(text=time2)
```

```
    clock.after(200, tick)
```

```
tick()
```

```
window.mainloop()
```

```
1 # write a gui clock program for a computer
```

```
3 import tkinter
```

```
4 from time import strftime
```

```
5
```

```
6 #Create a window
```

```
7 window = tkinter.Tk()
```

```
8 window.title("Digital Clock")
```

```
9 window.geometry("400x100")
```

```
10
```

```
11 #Create a label to display time
```

```
12 clock = tkinter.Label(window, font=("times", 30, "bold"), bg="green")
```

```
13 clock.grid(row=1, column=1, padx=5, pady=5)
```

```
14
```

```
15 def tick():
```

```
16     time2 = strftime('%H:%M:%S %p')
```

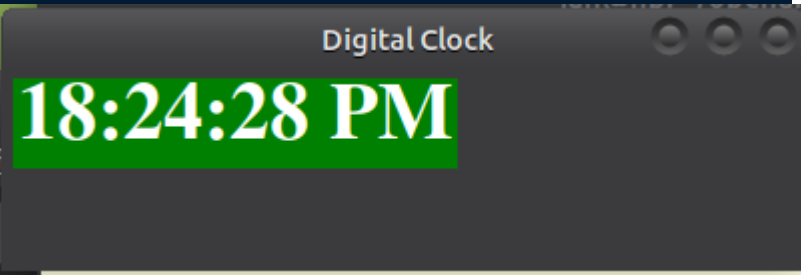
```
17     clock.config(text=time2)
```

```
18     clock.after(200, tick)
```

```
19
```

```
20 tick()
```

```
21 window.mainloop()
```



Gtk Clock – Truncated output.

write a gtk gui clock proram in python

```
#!/usr/bin/env python
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk
import time

class ClockApp(Gtk.Window):
    def __init__(self):
        Gtk.Window.__init__(self, title="GTK Clock")

        self.set_border_width(10)
        self.set_size_request(200, 200)

        vbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        self.add(vbox)

        self.label = Gtk.Label()
        self.label.set_markup("<span font_desc='Sans Bold 60'>00:00:00</span>")
        vbox.pack_start(self.label, True, True, 0)

        button = Gtk.Button(label="Quit")
        button.connect("clicked", self.on_clicked_quit)
        vbox.pack_start(button, True, True, 0)
```

The maximum number of tokens to **generate**. Requests can use up to 2,048 or 4,000 tokens shared between prompt and completion. The exact limit varies by model. (One token is roughly 4 characters for normal English text)

Stops! Huh?

Model

text-davinci-003

Temperature

0.7

Maximum length

256

Increase Maximum length

Model

text-davinci-003

Temperature

0.7

Maximum length

1000

Gtk Clock. With increased “Maximum Length”

```
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk, Gdk
import time

class ClockWindow(Gtk.Window):
    def __init__(self):
        Gtk.Window.__init__(self, title="GTK Clock")
        self.set_border_width(10)
        self.set_default_size(300, 200)
        self.set_position(Gtk.WindowPosition.CENTER)

        self.label = Gtk.Label("Time will appear here")
        self.add(self.label)

        # Create timeout to update the clock every second
        GObject.timeout_add(1000, self.update_clock)

    def update_clock(self):
        # Get the current time
        current_time = time.strftime("%H:%M:%S")
        # Update the label with the current time
        self.label.set_label(current_time)
        # Return true so the timeout will run again
        return True

window = ClockWindow()
window.connect("destroy", Gtk.main_quit)
window.show_all()
Gtk.main()
```

Was Truncated

Gtk Clock – Issues with the code

```
1 # write a gtk gui clock proram in python
2 # Original - Fails to run
```

Gtk now at version 4, but version 3 still supported

```
3
4 import gi
5 gi.require_version('Gtk', '3.0')
6 from gi.repository import Gtk, Gdk
7 import time
```

No GObject, but deprecated need GLib. Gdk not used

```
8
9 class ClockWindow(Gtk.Window):
10     def __init__(self):
11         Gtk.Window.__init__(self, title="GTK Clock")
12         self.set_border_width(10)
13         self.set_default_size(300, 200)
14         self.set_position(Gtk.WindowPosition.CENTER)
15
16         self.label = Gtk.Label("Time will appear here")
17         self.add(self.label)
18
19         # Create timeout to update the clock every second
20         GObject.timeout_add(1000, self.update_clock)
21
22     def update_clock(self):
23         # Get the current time
24         current_time = time.strftime("%H:%M:%S")
25         # Update the label with the current time
26         self.label.set_label(current_time)
27         # Return true so the timeout will run again
28         return True
```

Deprecated. Use: label="Time will appear here"

Deprecated. Use Glib not GObject

```
29
30 window = ClockWindow()
31 window.connect("destroy", Gtk.main_quit)
32 window.show_all()
33 Gtk.main()
```


Before and After

```
1 # write a gtk gui clock proram in python
2 # Original - Fails to run
3
4 import gi
5 gi.require_version('Gtk', '3.0')
6 from gi.repository import Gtk, Gdk
7 import time
8
9 class ClockWindow(Gtk.Window):
10     def __init__(self):
11         Gtk.Window.__init__(self, title="GTK Clock")
12         self.set_border_width(10)
13         self.set_default_size(300, 200)
14         self.set_position(Gtk.WindowPosition.CENTER)
15
16         self.label = Gtk.Label("Time will appear here")
17         self.add(self.label)
18
19         # Create timeout to update the clock every second
20         GObject.timeout_add(1000, self.update_clock)
21
22     def update_clock(self):
23         # Get the current time
24         current_time = time.strftime("%H:%M:%S")
25         # Update the label with the current time
26         self.label.set_label(current_time)
27         # Return true so the timeout will run again
28         return True
29
30 window = ClockWindow()
31 window.connect("destroy", Gtk.main_quit)
32 window.show_all()
33 Gtk.main()
```

```
1 # write a gtk gui clock proram in python
2 # Updated
3
4 import gi
5 gi.require_version('Gtk', '3.0')
6 from gi.repository import Gtk, GLib
7 import time
8
9 class ClockWindow(Gtk.Window):
10     def __init__(self):
11         Gtk.Window.__init__(self, title="GTK Clock")
12         self.set_border_width(10)
13         self.set_default_size(300, 200)
14         self.set_position(Gtk.WindowPosition.CENTER)
15
16         self.label = Gtk.Label(label="Time will appear here")
17         self.add(self.label)
18
19         # Create timeout to update the clock every second
20         GLib.timeout_add(1000, self.update_clock)
21
22     def update_clock(self):
23         # Get the current time
24         current_time = time.strftime("%H:%M:%S")
25         # Update the label with the current time
26         self.label.set_label(current_time)
27         # Return true so the timeout will run again
28         return True
29
30 window = ClockWindow()
31 window.connect("destroy", Gtk.main_quit)
32 window.show_all()
33 Gtk.main()
```

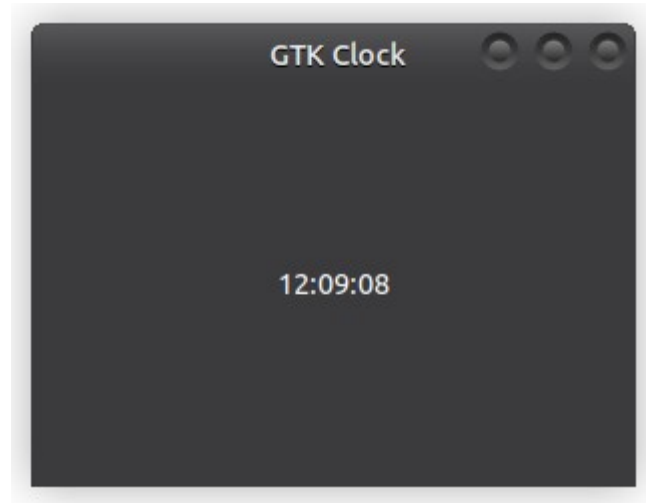
```

1 # write a gtk gui clock program in python
2 # Updated
3
4 import gi
5 gi.require_version('Gtk', '3.0')
6 from gi.repository import Gtk, GLib
7 import time
8
9 class ClockWindow(Gtk.Window):
10     def __init__(self):
11         Gtk.Window.__init__(self, title="GTK Clock")
12         self.set_border_width(10)
13         self.set_default_size(300, 200)
14         self.set_position(Gtk.WindowPosition.CENTER)
15
16         self.label = Gtk.Label(label="Time will appear here")
17         self.add(self.label)
18
19         # Create timeout to update the clock every second
20         GLib.timeout_add(1000, self.update_clock)
21
22     def update_clock(self):
23         # Get the current time
24         current_time = time.strftime("%H:%M:%S")
25         # Update the label with the current time
26         self.label.set_label(current_time)
27         # Return true so the timeout will run again
28         return True
29
30 window = ClockWindow()
31 window.connect("destroy", Gtk.main_quit)
32 window.show_all()
33 Gtk.main()

```

Running the updated Gtk clock

\$ python clock_gtk.py



Qt GUI Clock

Preference: List the Widgets

```
import QWidget, QLabel, QApplication
```

Add: , QTime

```
5 from PyQt5.QtCore import QTimer, QTime
```

Increase size of string:
("Time: ", self)

```
("Time: ", self)
```

QTime

Qt GUI Clock

Time: 16:11:53

write a qt gui clock program in python

```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtCore import QTimer

class Window(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Qt GUI Clock")
        self.setGeometry(50,50,500,300)
        self.UI()

    def UI(self):
        self.timer = QTimer()
        self.timer.timeout.connect(self.time)
        self.timer.start(1000)
        self.labelTime = QLabel("Time: ", self)
        self.labelTime.move(200,50)
        self.show()

    def time(self):
        time = QTime.currentTime().toString()
        self.labelTime.setText("Time: " + time)

def main():
    App = QApplication(sys.argv)
    window = Window()
    sys.exit(App.exec_())

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

```
1 # write a qt gui clock program in python
2
3 import sys
4 from PyQt5.QtWidgets import *
5 from PyQt5.QtCore import QTimer
6
7 class Window(QWidget):
8     def __init__(self):
9         super().__init__()
10        self.setWindowTitle("Qt GUI Clock")
11        self.setGeometry(50,50,500,300)
12        self.UI()
13
14    def UI(self):
15        self.timer = QTimer()
16        self.timer.timeout.connect(self.time)
17        self.timer.start(1000)
18        self.labelTime = QLabel("Time: ", self)
19        self.labelTime.move(200,50)
20        self.show()
21
22    def time(self):
23        time = QTime.currentTime().toString()
24        self.labelTime.setText("Time: " + time)
25
26
27 def main():
28     App = QApplication(sys.argv)
29     window = Window()
30     sys.exit(App.exec_())
31
32 if __name__ == '__main__':
33     main()
```


Square root of x

write a python program to find square root of X

```
# Python program to find the square root of a number
```

```
# importing math library for sqrt()
```

```
import math
```

```
# Input a number
```

```
num = float(input("Enter a number: "))
```

```
# Calculate the square root of the number
```

```
num_sqrt = math.sqrt(num)
```

```
# Print the number
```

```
print('The square root of %0.3f is %0.3f'%(num ,num_sqrt))
```

```
1 #write a python program to find square root of X
2
3 # Python program to find the square root of a number
4
5 # importing math library for sqrt()
6 import math
7
8 # Input a number
9 num = float(input("Enter a number: "))
10
11 # Calculate the square root of the number
12 num_sqrt = math.sqrt(num)
13
14 # Print the number
15 print('The square root of %0.3f is %0.3f'%(num ,num_sqrt))
```

```
$ python square_root.py
```

```
Enter a number: 98.01
```

```
The square root of 98.010 is 9.900
```

```
$ python square_root.py
```

```
Enter a number: 169
```

```
The square root of 169.000 is 13.000
```


Gtk GUI Square root of x

write a gtk qui python program to find square root of X

```
#import required modules
```

```
import gtk
```

from gi.repository import Gtk

```
import math
```

```
#create window
```

Change all gtk. to Gtk.

```
window = gtk.Window(title="Find Square Root")
```

```
window.set_position(gtk.WIN_POS_CENTER)
```

Obsolete

```
window.set_size_request(150, 100)
```

```
#create entry for input
```

```
entry = gtk.Entry()
```

```
entry.set_max_length(50)
```

Call before function defined

```
entry.connect("activate", calculate_sqrt)
```

```
entry.set_text("Enter a number")
```

```
entry.set_size_request(120, 30)
```

```
#create label for output
```

```
label = gtk.Label()
```

```
#method to calculate square root
```

```
def calculate_sqrt(entry):
```

```
    try:
```

Position function above call

```
        #convert input to float
```

```
        num = float(entry.get_text())
```

```
        #calculate square root
```

```
        result = math.sqrt(num)
```

```
        #set result in label
```

```
        label.set_label(str(result))
```

```
    except ValueError:
```

```
        label.set_label("Invalid input.")
```

```
#create a vertical box
```

```
vbox = gtk.VBox()
```

```
vbox.pack_start(entry, True, True, 0)
```

```
vbox.pack_start(label, True, True, 0)
```

```
#add box to window
```

```
window.add(vbox)
```

```
#show window
```

```
window.show_all()
```

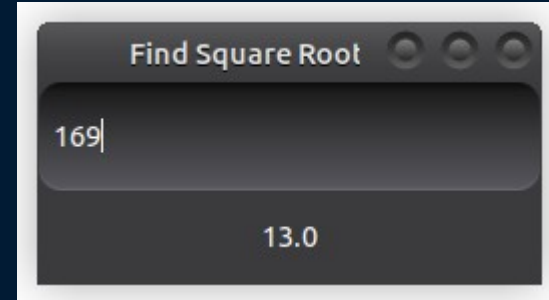
```
#main loop
```

```
gtk.main()
```

Gtk GUI Square root of x

```
1 #write a gtk gui python program to find square root of X
2
3 #import required modules
4 import gi
5 gi.require_version('Gtk', '3.0')
6 from gi.repository import Gtk
7
8 import math
9
10 #method to calculate square root
11 def calculate_sqrt(entry):
12     try:
13         #convert input to float
14         num = float(entry.get_text())
15         #calculate square root
16         result = math.sqrt(num)
17         #set result in label
18         label.set_label(str(result))
19     except ValueError:
20         label.set_label("Invalid input.")
21
22
```

```
23 #create window
24 window = Gtk.Window(title="Find Square Root")
25 #window.set_position(Gtk.WIN_POS_CENTER)
26 window.set_size_request(250, 100)
27
28 #create entry for input
29 entry = Gtk.Entry()
30 entry.set_max_length(50)
31 entry.connect("activate", calculate_sqrt)
32 entry.set_text("Enter a number")
33 entry.set_size_request(120, 30)
34
35 #create label for output
36 label = Gtk.Label()
37
38 #create a vertical box
39 vbox = Gtk.VBox()
40 vbox.pack_start(entry, True, True, 0)
41 vbox.pack_start(label, True, True, 0)
42
43 #add box to window
44 window.add(vbox)
45
46 #show window
47 window.show_all()
48
49 #main loop
50 Gtk.main()
```



Remove duplicates from a list

remove duplicates from a python list

```
def remove_duplicates(lst):  
    final_list = []  
    for num in lst:  
        if num not in final_list:  
            final_list.append(num)  
    return final_list
```

Driver Code

```
lst = [10, 20, 30, 20, 10, 50, 60, 40, 80, 50, 40]  
print(remove_duplicates(lst))
```

remove duplicate strings from a python list

```
def remove_duplicates(list):  
    new_list = list(dict.fromkeys(list))  
    return new_list
```

```
list = [1,2,3,1,2,3,4,5,6,7,7,8,9,10]  
print(remove_duplicates(list))
```

```
10 # ChatGPT: remove duplicates from a python list  
11 def remove_duplicates(lst):  
12     final_list = []  
13     for num in lst:  
14         if num not in final_list:  
15             final_list.append(num)  
16     return final_list  
17  
18 # Driver Code  
19 lst = [10, 20, 30, 20, 10, 50, 60, 40, 80, 50, 40]  
20 print(remove_duplicates(lst))  
21  
22 # ChatGPT: remove duplicate strings from a python list  
23 def remove_duplicates(listx):  
24     new_list = list(dict.fromkeys(listx))  
25     return new_list  
26  
27 listx = [1,2,3,1,2,3,4,5,6,7,7,8,9,10]  
28 print(remove_duplicates(listx))
```

```
$ python remove_duplicates.py  
[10, 20, 30, 50, 60, 40, 80]  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Example Applications: <https://platform.openai.com/examples>

Examples

Explore what's possible with some example applications

All categories



Q&A

Answer questions based on existing knowledge.



Grammar correction

Corrects sentences into standard English.



Summarize for a 2nd grader

Translates difficult text into simpler concepts.



Natural language to OpenAI API

Create code to call to the OpenAI API using a natural language instruction.



Text to command

Translate text into programmatic commands.



English to other languages

Translates English text into French, Spanish and Japanese.



Natural language to Stripe API

Create code to call the Stripe API using natural language.



SQL translate

Translate natural language to SQL queries.



Parse unstructured data

Create tables from long form text



Classification

Classify items into categories via example.



Python to natural language

Explain a piece of Python code in human understandable language.



Movie to Emoji

Convert movie titles into emoji.



Calculate Time Complexity

Find the time complexity of a function.



Translate programming languages

Translate from one programming language to another



Text to command

Translate text into programmatic commands.



Python to natural language

Explain a piece of Python code in human understandable language.



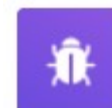
Translate programming languages

Translate from one programming language to another



Explain code

Explain a complicated piece of code.



Python bug fixer

Find and fix bugs in source code.



JavaScript to Python

Convert simple JavaScript expressions into Python.



Write a Python docstring

An example of how to create a docstring for a given Python function. We specify the Python

Natural Language to Python Application – random_names.py

Load preset

Natural language to python



```
"""
```

1. Create a list of first names
2. Create a list of last names
3. Combine them randomly into a list of 100 full names

```
"""
```

Submit



209

Natural Language to Python Application – random_names.py

```
import random
```

```
#1
```

```
first_names = ['John', 'Dave', 'Steve', 'Michael', 'Alex', 'Amy', 'Mary', 'Mandy', 'Jane', 'Claire']
```

```
#2
```

```
last_names = ['Smith', 'Wilson', 'Johnson', 'Williams', 'Black', 'White', 'Davis', 'Jones', 'Thompson', 'Taylor']
```

```
#3
```

```
full_names = []
```

```
for i in range(100):
```

```
    first_name = random.choice(first_names)
```

```
    last_name = random.choice(last_names)
```

```
    full_name = first_name + ' ' + last_name
```

```
    full_names.append(full_name)
```

```
print(full_names)
```


Natural Language to Python Application – random_names.py

```
1 """
2 1. Create a list of first names
3 2. Create a list of last names
4 3. Combine them randomly into a list of 100 full names
5 """
6
7 import random
8
9 #1
10 first_names = ['John', 'Dave', 'Steve', 'Michael', 'Alex', 'Amy', 'Mary', 'Mandy', 'Jane', 'Claire']
11
12 #2
13 last_names = ['Smith', 'Wilson', 'Johnson', 'Williams', 'Black', 'White', 'Davis', 'Jones', 'Thompson', 'Taylor']
14
15 #3
16 full_names = []
17
18 for i in range(100):
19     first_name = random.choice(first_names)
20     last_name = random.choice(last_names)
21     full_name = first_name + ' ' + last_name
22     full_names.append(full_name)
23
24 print(full_names)
```

Natural Language to Python Application – random_names.py

```
$ python random_names.py
['Jane Johnson', 'Claire Jones', 'Mandy Thompson', 'Claire Williams', 'John
Smith', 'Claire Williams', 'Dave Williams', 'John Thompson', 'Claire Taylor',
'Steve Johnson', 'Michael Black', 'Mandy Smith', 'Dave Black', 'Mandy Davis', 'Amy
White', 'Steve Black', 'Jane Black', 'Steve White', 'Jane Davis', 'Dave Smith',
'Alex Jones', 'Steve Smith', 'Mandy Davis', 'John Johnson', 'Michael Taylor',
'Dave Williams', 'Mary Smith', 'Jane Jones', 'Dave Johnson', 'Dave Wilson', 'Amy
Taylor', 'Alex Johnson', 'Jane White', 'Mary Wilson', 'Dave Williams', 'Michael
Jones', 'Mandy Johnson', 'Michael Taylor', 'Amy Williams', 'Alex Wilson', 'John
Smith', 'Steve Taylor', 'John White', 'Dave Johnson', 'John Wilson', 'Mary White',
'Alex Davis', 'Jane White', 'Dave Davis', 'Dave Wilson', 'Alex Taylor', 'John
Black', 'Amy Johnson', 'Claire Johnson', 'Mary Johnson', 'Mary Taylor', 'Alex
Jones', 'Michael Jones', 'Alex Johnson', 'Claire Wilson', 'Alex Davis', 'Mary
Thompson', 'Jane Black', 'Michael White', 'Jane Thompson', 'Jane Taylor', 'Jane
Johnson', 'Alex Williams', 'John Williams', 'Steve Johnson', 'Claire Thompson',
'Alex Davis', 'Jane Thompson', 'Jane White', 'Mary Johnson', 'John Thompson',
'Alex Black', 'Michael Black', 'Alex Black', 'Claire Davis', 'Mandy Smith', 'Steve
Williams', 'John Jones', 'Amy Johnson', 'John Taylor', 'Dave Taylor', 'Mary
White', 'Jane White', 'Claire Black', 'Amy Thompson', 'Amy Smith', 'Steve Smith',
'Alex Davis', 'Dave Wilson', 'Jane Black', 'Amy Johnson', 'John Wilson', 'John
Thompson', 'Dave Jones', 'Amy Smith']
```


Natural Language to Python Application – sphere.py

```
"""
```

1. prompt for value of the radius
2. Use radius to calculate volume of a sphere
3. Use radius to calculate surface area of a sphere

```
"""
```

```
# Prompt the user for the radius
```

```
radius = float(input("Please enter the radius of the sphere: "))
```

```
# Calculate the volume of the sphere
```

```
volume = (4/3) * 3.14 * (radius**3)
```

```
# Calculate the surface area of the sphere
```

```
surface_area = 4 * 3.14 * (radius**2)
```

```
# Print the result
```

```
print(f"The volume of the sphere is {volume:.2f} and the surface area is {surface_area:.2f}.")
```

```
1 """
2 1. prompt for value of the radius
3 2. Use radius to calculate volume of a sphere
4 3. Use radius to calculate surface area of a sphere
5 """
6
7 # Prompt the user for the radius
8 radius = float(input("Please enter the radius of the sphere: "))
9
10 # Calculate the volume of the sphere
11 volume = (4/3) * 3.14 * (radius**3)
12
13 # Calculate the surface area of the sphere
14 surface_area = 4 * 3.14 * (radius**2)
15
16 # Print the result
17 print(f"The volume of the sphere is {volume:.2f} and the surface
    area is {surface_area:.2f}.")
```

```
$ python sphere.py
```

```
Please enter the radius of the sphere: 10
```

```
The volume of the sphere is 4186.67 and the surface area is 1256.00.
```

Natural Language to Python Application – sphere_v2.py

.....

1. prompt for value of the radius
2. Use math.pi
3. Use radius to calculate volume of a sphere
4. Use radius to calculate surface area of a sphere

.....

```
import math
```

```
# prompt for value of the radius
```

```
radius = float(input("Enter the radius of the sphere: "))
```

```
# Use radius to calculate volume of a sphere
```

```
volume = (4/3) * math.pi * radius**3
```

```
# Use radius to calculate surface area of a sphere
```

```
surface_area = 4 * math.pi * radius**2
```

```
# Display results
```

```
print("The volume of the sphere is {:.2f}".format(volume))
```

```
print("The surface area of the sphere is {:.2f}".format(surface_area))
```

Use math.pi, not 3.14

```
8 import math
9
10 # prompt for value of the radius
11 radius = float(input("Enter the radius of the sphere: "))
12
13 # Use radius to calculate volume of a sphere
14 volume = (4/3) * math.pi * radius**3
15
16 # Use radius to calculate surface area of a sphere
17 surface_area = 4 * math.pi * radius**2
18
19 # Display results
20 print("The volume of the sphere is {:.2f}".format(volume))
21 print("The surface area of the sphere is {:.2f}".format(surface_area))
```

```
$ python sphere_v2.py
```

```
Enter the radius of the sphere: 10
```

```
The volume of the sphere is 4188.79
```

```
The surface area of the sphere is 1256.64
```

Python to Natural Language Application



Python to natural language

[Code](#)[Translation](#)[Open in Playground](#)

Explain a piece of Python code in human understandable language.

Prompt

```
# Python 3
def remove_common_prefix(x, prefix, ws_prefix):
    x["completion"] = x["completion"].str[len(prefix) :]
    if ws_prefix:
        # keep the single whitespace as prefix
        x["completion"] = " " + x["completion"]
    return x

# Explanation of what the code does

#
```

Settings

Engine	code-davinci-002
Max tokens	64
Temperature	0
Top p	1.0
Frequency	0.0
penalty	
Presence penalty	0.0
Stop sequence	

Sample response

The code above is a function that takes a dataframe and a prefix as input and returns a dataframe with the prefix removed from the completion column.

Python to Natural Language Application

Playground

Load preset

Explain code

Save

```
2 gi.require_version('Gtk', '3.0')
3 from gi.repository import Gtk, GLib
4 import time
5
6 class ClockWindow(Gtk.Window):
7     def __init__(self):
8         Gtk.Window.__init__(self, title="GTK Clock")
9         self.set_border_width(10)
10        self.set_default_size(300, 200)
11        self.set_position(Gtk.WindowPosition.CENTER)
12
13        self.label = Gtk.Label(label="Time will appear here")
14        self.add(self.label)
15
16        # Create timeout to update the clock every second
17        GLib.timeout_add(1000, self.update_clock)
18
19    def update_clock(self):
20        # Get the current time
21        current_time = time.strftime("%H:%M:%S")
22        # Update the label with the current time
23        self.label.set_label(current_time)
24        # Return true so the timeout will run again
25        return True
26
27 window = ClockWindow()
28 window.connect("destroy", Gtk.main_quit)
29 window.show_all()
30 Gtk.main()
```

Explain code

EXAMPLES

Grammatical Standard English

Summarize for a 2nd grader

Text to command

Q&A

English to other languages

Parse unstructured data

Classification

Natural language to python

Explain code

Python to Natural Language Application

Playground

Explain code

Save

```
2 gi.require_version('Gtk', '3.0')
3 from gi.repository import Gtk, GLib
4 import time
5
6 class ClockWindow(Gtk.Window):
7     def __init__(self):
8         Gtk.Window.__init__(self, title="GTK Clock")
9         self.set_border_width(10)
10        self.set_default_size(300, 200)
11        self.set_position(Gtk.WindowPosition.CENTER)
12
13        self.label = Gtk.Label(label="Time will appear here")
14        self.add(self.label)
15
16        # Create timeout to update the clock every second
17        GLib.timeout_add(1000, self.update_clock)
18
19    def update_clock(self):
20        # Get the current time
21        current_time = time.strftime("%H:%M:%S")
22        # Update the label with the current time
23        self.label.set_label(current_time)
24        # Return true so the timeout will run again
25        return True
26
27 window = ClockWindow()
28 window.connect("destroy", Gtk.main_quit)
29 window.show_all()
30 Gtk.main()
```

"""

Here's what the above class is doing:

1. The `__init__` method creates a label and adds it to the window.
2. The `update_clock` method gets the current time and updates the label.
3. The timeout is created with a call to `GLib.timeout_add`. The first argument is the number of milliseconds to wait before calling the function. The second argument is the function to call.
4. The `update_clock` method returns `True` so that the timeout will run again.

Paste this at the end of the python code...

"""

Here's what the above class is doing:
1.

Increase:
Maximum
length

Submit



287

Plain text

Controls: Model / Temperature / Length.

write a gtk gui clock program in python

The model which will generate the completion. Some models are suitable for natural language tasks, others specialize in code. [Learn more.](#)

Controls randomness: Lowering results in less random completions. As the temperature approaches zero, the model will become deterministic and repetitive.

The maximum number of tokens to **generate**. Requests can use up to 2,048 or 4,000 tokens shared between prompt and completion. The exact limit varies by model. (One token is roughly 4 characters for normal English text)

Model

text-davinci-003



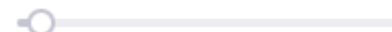
Temperature

0.7



Maximum length

256



9 tokens in prompt
Up to 256 tokens in response
[Learn more about pricing](#)

Stopwatch Examples – Using “Chat”

write a stopwatch program in python

```
import time
```

```
# define the start time
```

```
start = time.time()
```

```
# keep looping until user presses 'q'
```

```
while True:
```

```
    # print the elapsed time
```

```
    print('Elapsed Time: {:.2f}s'.format(time.time() - start))
```

```
    # prompt user to quit
```

```
    user_input = input('Press "q" to quit: ')
```

```
    if user_input == 'q':
```

```
        break
```

```
2 # write a stopwatch program in python
3
4 import time
5
6 # define the start time
7 start = time.time()
8
9 # keep looping until user presses 'q'
10 while True:
11     # print the elapsed time
12     print('Elapsed Time: {:.2f}s'.format(time.time() - start))
13
14     # prompt user to quit
15     user_input = input('Press "q" to quit: ')
16
17     if user_input == 'q':
18         break
```

```
$ python stopwatch.py
```

```
Elapsed Time: 0.00s
```

```
Press "q" to quit: #<--- Pressed Return key
```

```
Elapsed Time: 1.43s
```

```
Press "q" to quit:
```

```
Elapsed Time: 3.53s
```

```
Press "q" to quit: q
```

Stopwatch Examples – Using “Natural Language to python”

1. Create a stopwatch class.
2. In the class provide the methods: start and stop.
3. provide a menu to select start and stop commands to call class
4. Display elapsed time.

```
1 """
2 1. Create a stopwatch class.
3 2. In the class provide the methods: start and stop.
4 3. provide a menu to select start and stop comands to call class
5 4. Display elapsed time.
6 """
7
8 import time
9
10 # Create a Stopwatch class
11 class Stopwatch:
12     # Define the start method
13     def start(self):
14         self.start_time = time.time()
15
16     # Define the stop method
17     def stop(self):
18         self.stop_time = time.time()
19         self.elapsed_time = self.stop_time - self.start_time
20     # Define the elapsed time
21     self.elapsed_time = round(self.elapsed_time,2)
22
```

```
23 # Create a menu
24 def menu():
25     print("\nWhat would you like to do?")
26     print("1. Start Timer")
27     print("2. Stop Timer")
28     print("3. Quit")
29
30 # Create a main function
31 def main():
32     # Make an instance of the Stopwatch class
33     timer = Stopwatch()
34     choice = 0
35     while choice != 3:
36         menu()
37         choice = int(input("\nPlease enter your choice: "))
38         if choice == 1:
39             timer.start()
40         elif choice == 2:
41             timer.stop()
42             print("\nElapsed time: ", timer.elapsed_time)
43         elif choice == 3:
44             print("\nGoodbye!")
45
46     else:
47         print("\nThat is not a valid choice. Please try again!")
48
49 # Call the main function
50 main()
```


Stopwatch Examples – Using “Natural Language to python”

```
$ python stopwatch_class.py
```

```
What would you like to do?
```

1. Start Timer
2. Stop Timer
3. Quit

```
Please enter your choice: 1
```

```
What would you like to do?
```

1. Start Timer
2. Stop Timer
3. Quit

```
Please enter your choice: 2
```

```
Elapsed time: 4.27
```

```
What would you like to do?
```

1. Start Timer
2. Stop Timer
3. Quit

```
Please enter your choice: 3
```

```
Goodbye!
```

Stopwatch Examples – Using “Explain code” – Temperature 0.0

Here's what the above class is doing:

1. The start method records the current time.
2. The stop method records the current time and then calculates the elapsed time.
3. The elapsed time is stored in the elapsed_time attribute.
4. The elapsed time is rounded to two decimal places.

1. Create a stopwatch class.
2. In the class provide the methods: start and stop.
3. provide a menu to select start and stop commands to call class
4. Display elapsed time.

Stopwatch Examples – Using “Explain code” – Temperature 0.7

Here's what the above class is doing:

1. The Stopwatch class defines the start, stop and elapsed time methods.
2. In the main function, an instance of the Stopwatch class is made.
3. In the main function, a while loop is created to run the main function.

The while loop is set to run until the user enters in a value of 3.

4. The menu function is called in the main function.

This function creates a menu and prompts the user to enter in a value.

5. The user's input is saved to a variable called choice.
6. An if statement is created to test the value of choice.

If choice is equal to 1, then the start method from the Stopwatch class is called.

7. An elif statement is created to test the value of choice.

If choice is equal to 2, then the stop method from the Stopwatch class is called.

Then, the elapsed time method from the Stopwatch class is called and saved to the elapsed_time variable.

The elapsed time is printed to the screen.

8. An elif statement is created to test the value of choice.

If choice is equal to 3, then a message is printed to the screen.

9. An else statement is created to test the value of choice.

If choice is not equal to 1, 2, or 3, then a message is printed to the screen.

1. Create a stopwatch class.
2. In the class provide the methods: start and stop.
3. provide a menu to select start and stop commands to call class
4. Display elapsed time.

WebServer – Using “Natural language to Python” – Temperature 0.7

1. Create a webserver engine
2. Add a webpage that is green

```
"""
```

1. Create a webserver engine
2. Add a webpage that is green

```
"""
```

```
# Answer
```

```
#1. Create a webserver engine
```

```
from http.server import HTTPServer, BaseHTTPRequestHandler
```

```
class GreenWebPage(BaseHTTPRequestHandler):
```

```
    def do_GET(self):
```

```
        self.send_response(200)
```

```
        self.end_headers()
```

```
        self.wfile.write(b'<html><head><title>Green Webpage</title></head><body bgcolor="green"></body></html>')
```

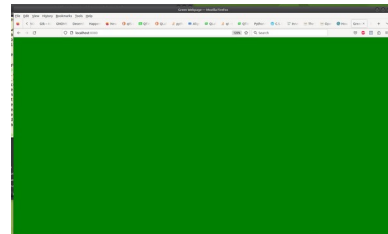
```
httpd = HTTPServer(('localhost', 8080), GreenWebPage)
```

```
httpd.serve_forever()
```

Suggestion: Add at start of python code...

print("View Webpage on a browser: <http://localhost:8080/>")

```
1 """
2 1. Create a webserver engine
3 2. Add a webpage that is green
4 """
5
6 # Answer
7 #1. Create a webserver engine
8
9 from http.server import HTTPServer, BaseHTTPRequestHandler
10
11 class GreenWebPage(BaseHTTPRequestHandler):
12
13     def do_GET(self):
14         self.send_response(200)
15         self.end_headers()
16         self.wfile.write(b'<html><head><title>Green
17 Webpage</title></head><body bgcolor="green"></body></
18 html>')
19
20 httpd = HTTPServer(('localhost', 8080), GreenWebPage)
21 httpd.serve_forever()
```



WebServer – Using “Natural language to Python” – Temperature 0.7

1. Create a HTTP webserver engine
2. Add a webpage.
3. Add to the webpage a form
4. The form has: first_name, surname, age fields
5. The form has a submit button

Suggestion: Add at start of python code...
`print("View Webpage on a browser: http://localhost:8000/")`

```
#Creating a HTTP webserver engine
from http.server import HTTPServer, BaseHTTPRequestHandler

#Creating a webpage
class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b'<html><head><title>Welcome!</title></head>')
        self.wfile.write(b'<body>')
        self.wfile.write(b'<h1>Welcome to my site!</h1>')

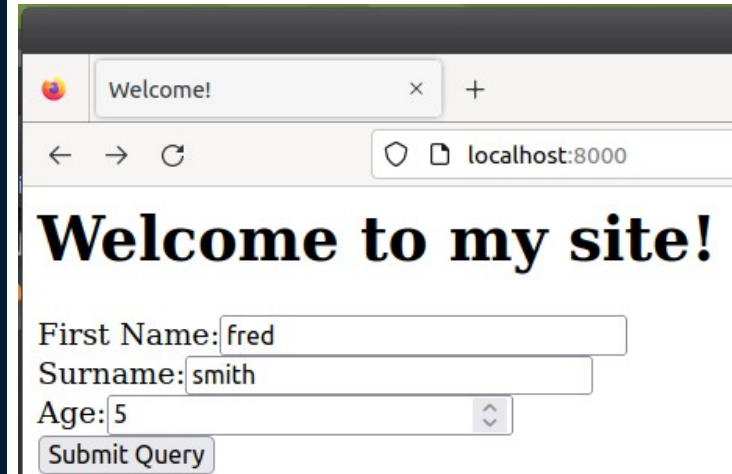
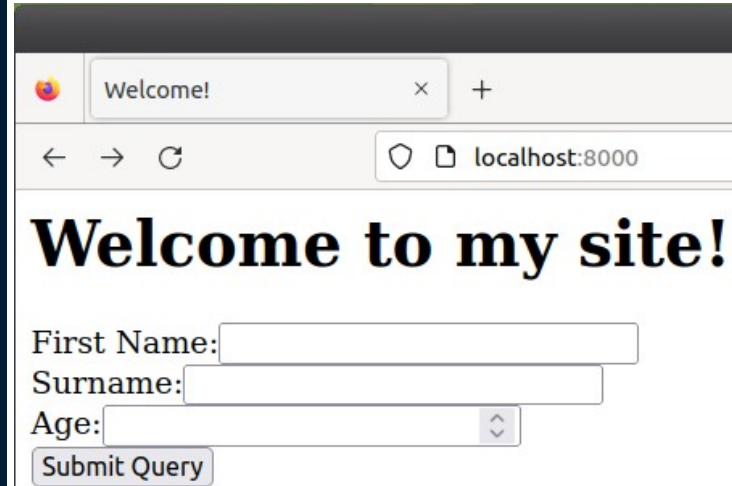
        #Adding a form to the webpage
        self.wfile.write(b'<form method="POST" action="/form-results">')
        self.wfile.write(b'<label>First Name:</label>')
        self.wfile.write(b'<input type="text" name="first_name"/><br>')
        self.wfile.write(b'<label>Surname:</label>')
        self.wfile.write(b'<input type="text" name="surname"/><br>')
        self.wfile.write(b'<label>Age:</label>')
        self.wfile.write(b'<input type="number" name="age"/><br>')
        self.wfile.write(b'<input type="submit"/>')
        self.wfile.write(b'</form>')
        self.wfile.write(b'</body></html>')

httpd = HTTPServer(('localhost', 8000), SimpleHTTPRequestHandler)
httpd.serve_forever()
```

WebServer – Using “Natural language to Python” – Temperature 0.7

```
9 #Creating a HTTP webserver engine
10 from http.server import HTTPServer, BaseHTTPRequestHandler
11
12 #Creating a webpage
13 class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
14
15     def do_GET(self):
16         self.send_response(200)
17         self.end_headers()
18         self.wfile.write(b'<html><head><title>Welcome!</title></head>')
19         self.wfile.write(b'<body>')
20         self.wfile.write(b'<h1>Welcome to my site!</h1>')
21
22     #Adding a form to the webpage
23     self.wfile.write(b'<form method="POST" action="/form-results">')
24     self.wfile.write(b'<label>First Name:</label>')
25     self.wfile.write(b'<input type="text" name="first_name"/><br>')
26     self.wfile.write(b'<label>Surname:</label>')
27     self.wfile.write(b'<input type="text" name="surname"/><br>')
28     self.wfile.write(b'<label>Age:</label>')
29     self.wfile.write(b'<input type="number" name="age"/><br>')
30     self.wfile.write(b'<input type="submit"/>')
31     self.wfile.write(b'</form>')
32     self.wfile.write(b'</body></html>')
33
34 httpd = HTTPServer(('localhost', 8000), SimpleHTTPRequestHandler)
35 httpd.serve_forever()
```

\$ python webserver_v2.py
View Webpage on a browser: http://localhost:8000/
127.0.0.1 - - [10/Feb/2023 18:33:04] "GET / HTTP/1.1" 200 -



WebServer Stopwatch – Using “Natural language to Python” – T = 0.7

1. Create a HTTP webserver engine.
2. Add a stopwatch webpage.
3. Add to the webpage a start and stop button.
4. Display elapsed time.

```
import http.server
```

```
# 2. Add a stopwatch webpage.
```

```
# Create a simple HTML page
```

```
html_page = """
<html>
  <head>
    <title>Stopwatch</title>
  </head>
  <body>
    <h1>Stopwatch</h1>
    <p>Elapsed time: <span id="elapsed"></span></p>
    <button id="start">Start</button>
    <button id="stop">Stop</button>
    <script>
      // 3. Add to the webpage a start and stop button.
      let startButton = document.getElementById("start");
      let stopButton = document.getElementById("stop");
      let elapsedTime = document.getElementById("elapsed");
```

Suggestion: Add at start of python code...

```
print("View Webpage on a browser: http://localhost:8000/")
```

```
    let startTime = null;
    let interval = null;
    let elapsed = 0;

    startButton.addEventListener("click", () => {
      startTime = Date.now();
      interval = setInterval(() => {
        elapsed = Date.now() - startTime;
        // 4. Display elapsed time.
        elapsedTime.innerHTML = elapsed / 1000;
      }, 1000);
    });
```

```
    stopButton.addEventListener("click", () => {
      clearInterval(interval);
    });
  </script>
</body>
</html>
"""
```

```
# Create a handler for the page
```

```
class StopwatchRequestHandler(http.server.BaseHTTPRequestHandler):
```

```
    def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
```

```
        self.wfile.write(html_page.encode())
```

```
# Create the server and start it
```

```
server_address = ('', 8000)
```

```
httpd = http.server.HTTPServer(server_address, StopwatchRequestHandler)
```

```
httpd.serve_forever()
```

```

11 # 1. Create a HTTP webserver engine.
12 import http.server
13
14 # 2. Add a stopwatch webpage.
15 # Create a simple HTML page
16
17 html_page = """
18 <html>
19   <head>
20     <title>Stopwatch</title>
21   </head>
22   <body>
23     <h1>Stopwatch</h1>
24     <p>Elapsed time: <span id="elapsed"></span></p>
25     <button id="start">Start</button>
26     <button id="stop">Stop</button>
27     <script>
28       // 3. Add to the webpage a start and stop button.
29       let startButton = document.getElementById("start");
30       let stopButton = document.getElementById("stop");
31       let elapsedTime = document.getElementById("elapsed");
32
33       let startTime = null;
34       let interval = null;
35       let elapsed = 0;
36
37       startButton.addEventListener("click", () => {
38         startTime = Date.now();
39         interval = setInterval(() => {
40           elapsed = Date.now() - startTime;
41           // 4. Display elapsed time.
42           elapsedTime.innerHTML = elapsed / 1000;
43         }, 1000);
44       });
45
46       stopButton.addEventListener("click", () => {
47         clearInterval(interval);
48       });
49     </script>
50   </body>
51 </html>
52 """

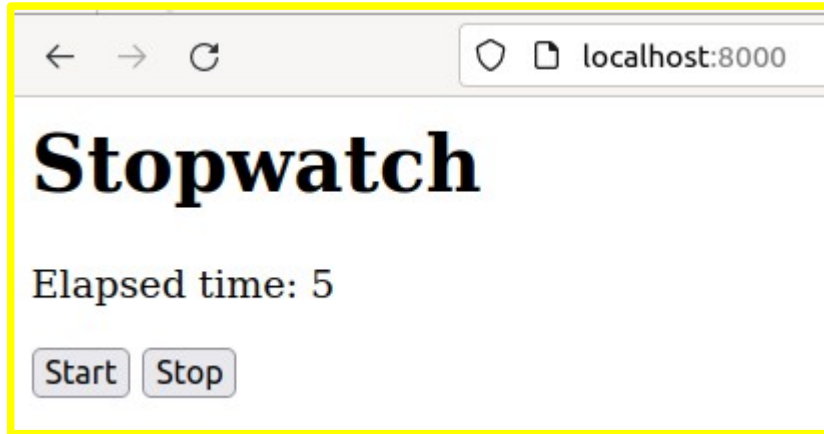
```

WebServer Stopwatch

```

53
54 # Create a handler for the page
55 class StopwatchRequestHandler(http.server.BaseHTTPRequestHandler):
56     def do_GET(self):
57         self.send_response(200)
58         self.send_header('Content-type', 'text/html')
59         self.end_headers()
60
61         self.wfile.write(html_page.encode())
62
63 # Create the server and start it
64 server_address = ('', 8000)
65 httpd = http.server.HTTPServer(server_address, StopwatchRequestHandler)
66 httpd.serve_forever()

```



Suggestion: Add at start of python code...
 print("View Webpage on a browser: http://localhost:8000/")

ChatGPT Account

When applying for a ChatGPT account at <https://chat.openai.com/chat>:

Initial requirements are to provide:

- Email address - (which is verified)
- Mobile phone number - (which is sent a token/pin)

Intention:

- Don't always play around in the Playground. You have applications that use OpenAI. OpenAI collect revenue depending on the traffic.

Pricing

Base models

Ada Fastest

\$0.0004 / 1K tokens

Babbage

\$0.0005 / 1K tokens

Curie

\$0.0020 / 1K tokens

Davinci Most powerful

\$0.0200 / 1K tokens

Multiple models, each with different capabilities and price points.

Ada is the fastest model, while **Davinci** is the most powerful.

Prices are per 1,000 tokens. You can think of tokens as pieces of words, where 1,000 tokens is about 750 words. This paragraph is 35 tokens.

🚩 Start for free

Start experimenting with \$18 in free credit that can be used during your first 3 months.

Usage

Below you'll find a summary of API usage for your organization. All dates and times are UTC-based, and data may be delayed up to 5 minutes.



Free trial usage



GRANT #	CREDIT GRANTED	EXPIRES (UTC)
Grant 1	\$18.00	1 June 2023

Daily usage breakdown (UTC)

8 February 2023 ▾

All org members ▾

Model usage	36 requests	▾
Fine-tune training	0 requests	▾

Account

Demo: ChatGPT to write Python code

- Log into: <https://chat.openai.com/chat>
- Use Playground: <https://platform.openai.com/playground>
- Chat: <https://chat.openai.com/chat>
- Playground: <https://platform.openai.com/playground>
- Playground Python:
<https://platform.openai.com/playground/p/default-english-to-python?model=text-davinci-003>
- Playground Explain: <https://platform.openai.com/playground/p/default-explain-code>
-
- Load Preset: “Chat”, “Natural Language to Python”, or “Explain Code”