

# Beginners Guide To Python

...

# Why Python

## Upsides

- No semicolons
- No setting variable types

## Downsides

- Not compiled language
- Whitespace matters

## Topics

- Input data
- Output data
- Variables
- Variable operations
- String concatenation or string format
- If statements
- For loops
  - `in range()`
- While loops

<https://docs.python.org/3/tutorial/index.html>

# Integrated development environment (IDE)

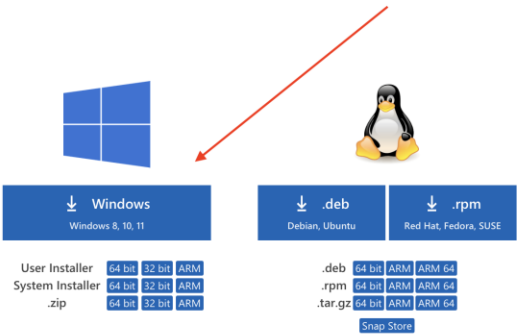
- Allows programmers to quickly program files
  - Built in error checking
  - Compiles and Runs the files for us
- 
- We will be using Visual Studio Code

# Installing VSCode

# Go to <https://code.visualstudio.com/download>

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



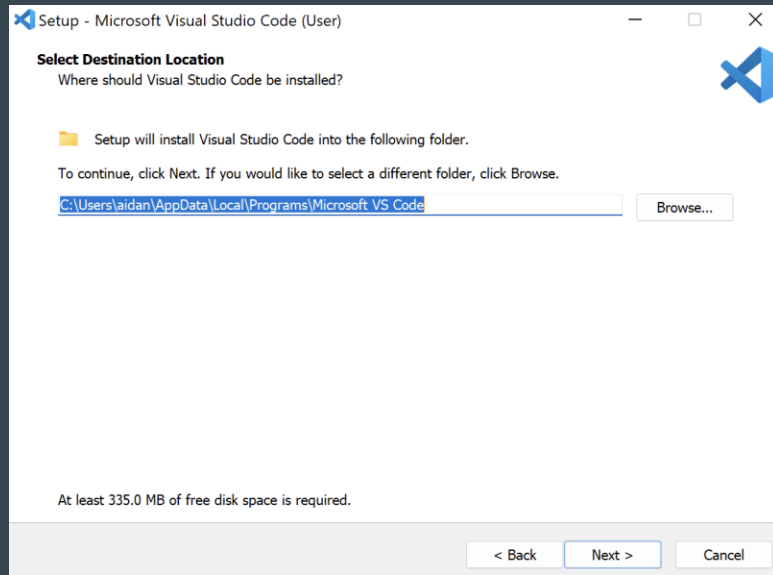
**Windows**  
Windows 8, 10, 11

User Installer 64 bit 32 bit ARM  
System Installer 64 bit 32 bit ARM  
.zip 64 bit 32 bit ARM

**Linux**  
Debian, Ubuntu .deb 64 bit ARM ARM 64  
Red Hat, Fedora, SUSE .rpm 64 bit ARM ARM 64  
.tar.gz 64 bit ARM ARM 64  
Snap Store

**Mac**  
macOS 10.11+

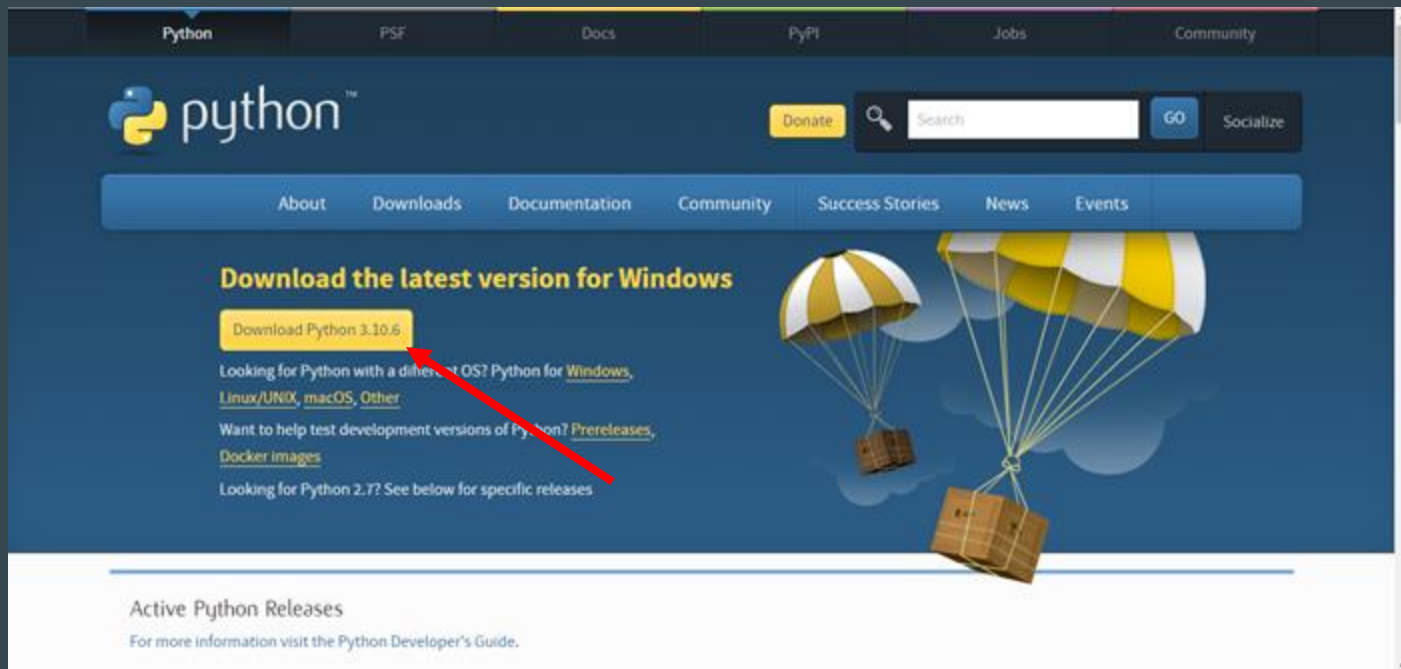
.zip Universal Intel Chip Apple Silicon



# Installing Python

# Downloading Python

[python.org/downloads/](https://python.org/downloads/)



The screenshot shows the Python.org website with a dark blue header and a lighter blue main content area. The header includes navigation links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a search bar with a 'Donate' button and a 'Socialize' button. The main content area features a large illustration of two parachutes with cargo boxes, one of which is labeled 'Python'. A red arrow points to the 'Download Python 3.10.6' button. Below this button are links for other operating systems and development versions.

**Download the latest version for Windows**

[Download Python 3.10.6](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

**Active Python Releases**

For more information visit the [Python Developer's Guide](#).

# Downloading Python

The screenshot shows the Python.org website with a dark blue header. The main navigation bar includes links for Python, PSF, Docs, PyPI, and Jobs. Below this is a secondary navigation bar with links for About, Downloads, Documentation, Community, Success Stories, and News. The main content area features a large yellow and white striped parachute with a crate hanging from it, and a smaller yellow parachute with a crate. The text 'Download the latest version for Windows' is prominently displayed. Below this, there is a button labeled 'Download Python 3.10.6'. The text 'Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)' is visible. Below that, it says 'Want to help test development versions of Python? [Prereleases](#), [Docker images](#)'. At the bottom, it says 'Looking for Python 2.7? See below for specific releases'. A red arrow points from the 'Downloads' link in the navigation bar to the 'python-3.10.6-amd64.exe' file in the 'Active Python Releases' section.

Python

PSF

Docs

PyPI

Jobs

python™

Donate

Search

About Downloads Documentation Community Success Stories News

**Download the latest version for Windows**

Download Python 3.10.6

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

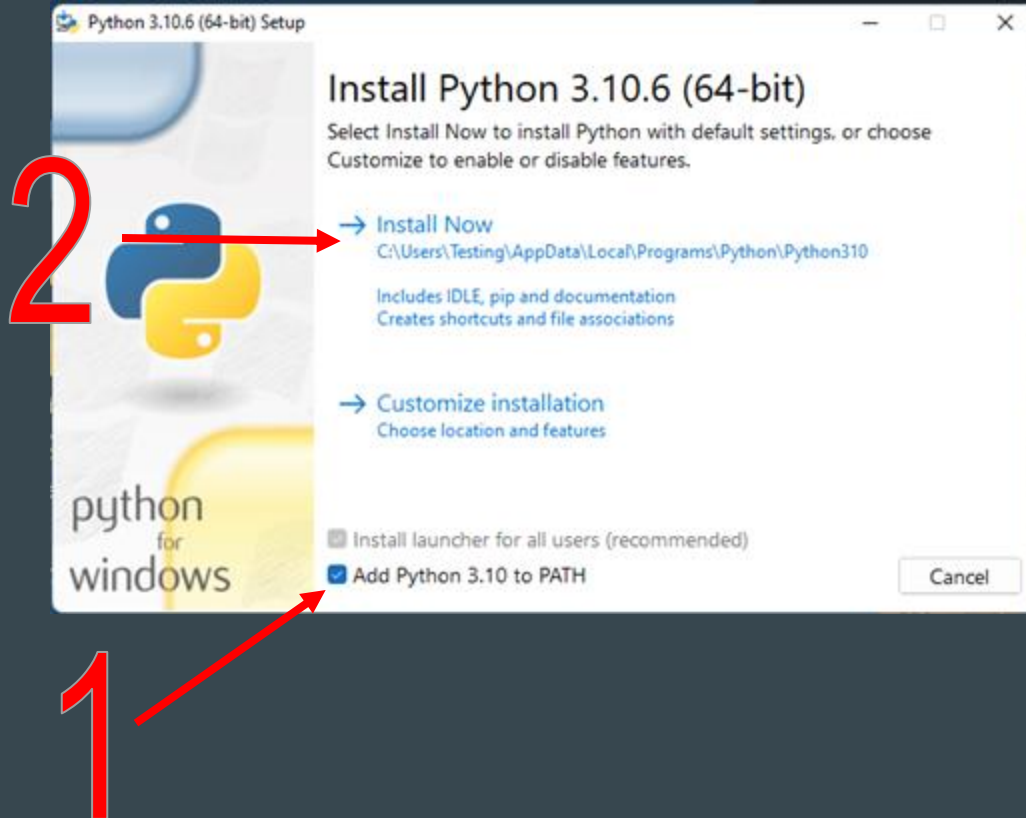
Looking for Python 2.7? See below for specific releases

Active Python Releases

python-3.10.6-amd64.exe

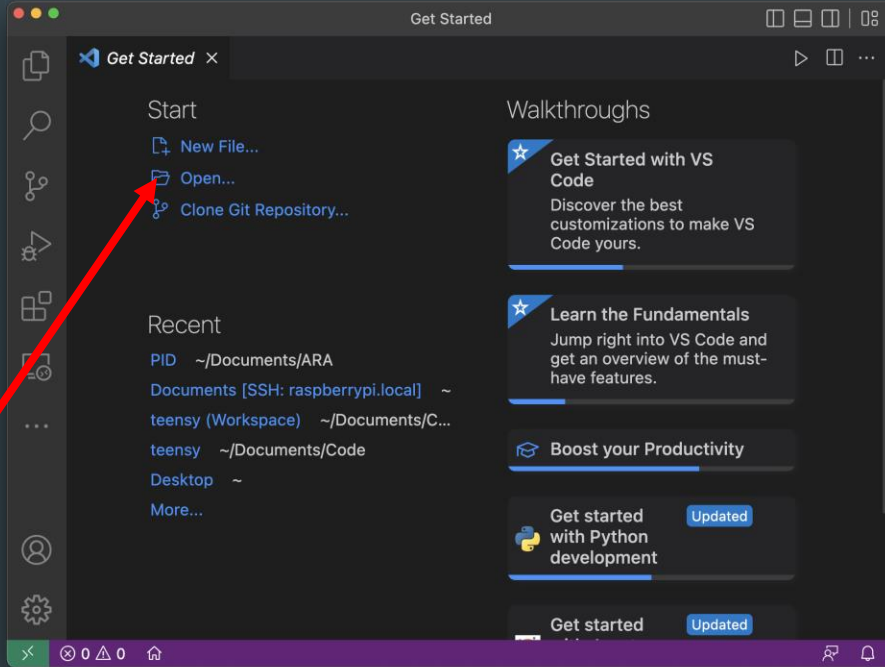


# Downloading Python

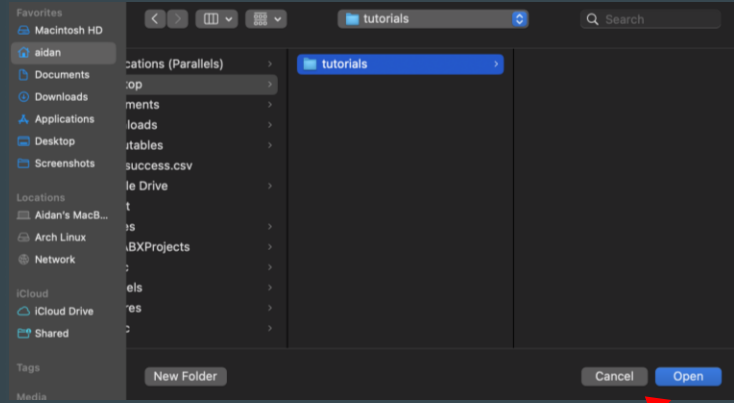


# Setting up “Hello World”

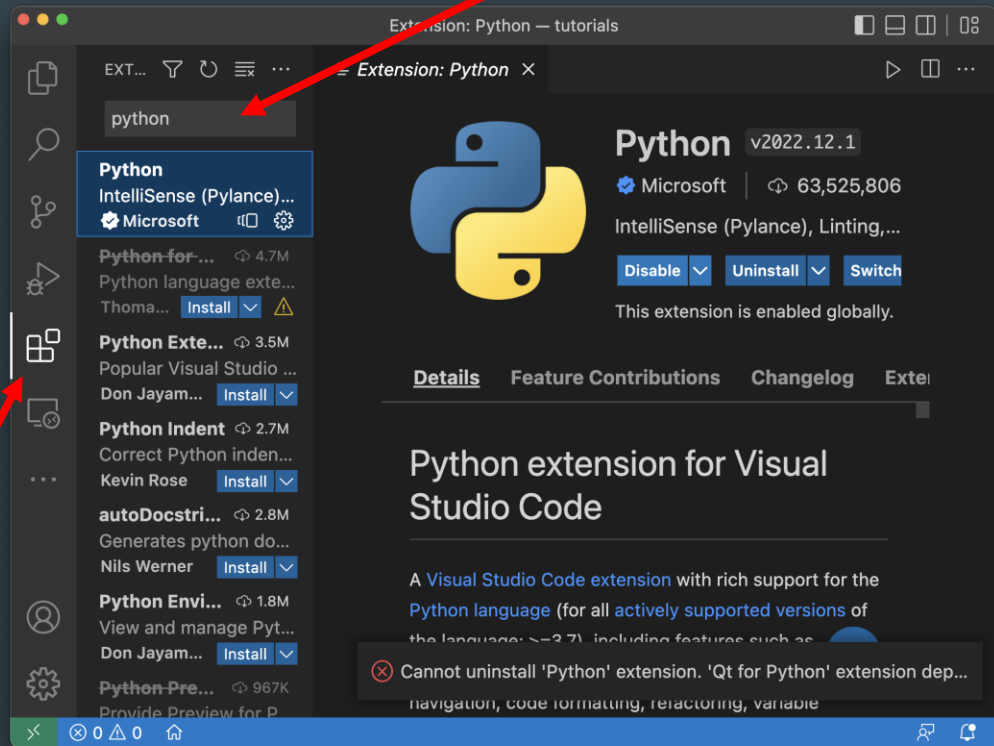
# Create a new project



# Open a folder on your computer where you wish to store all your programs

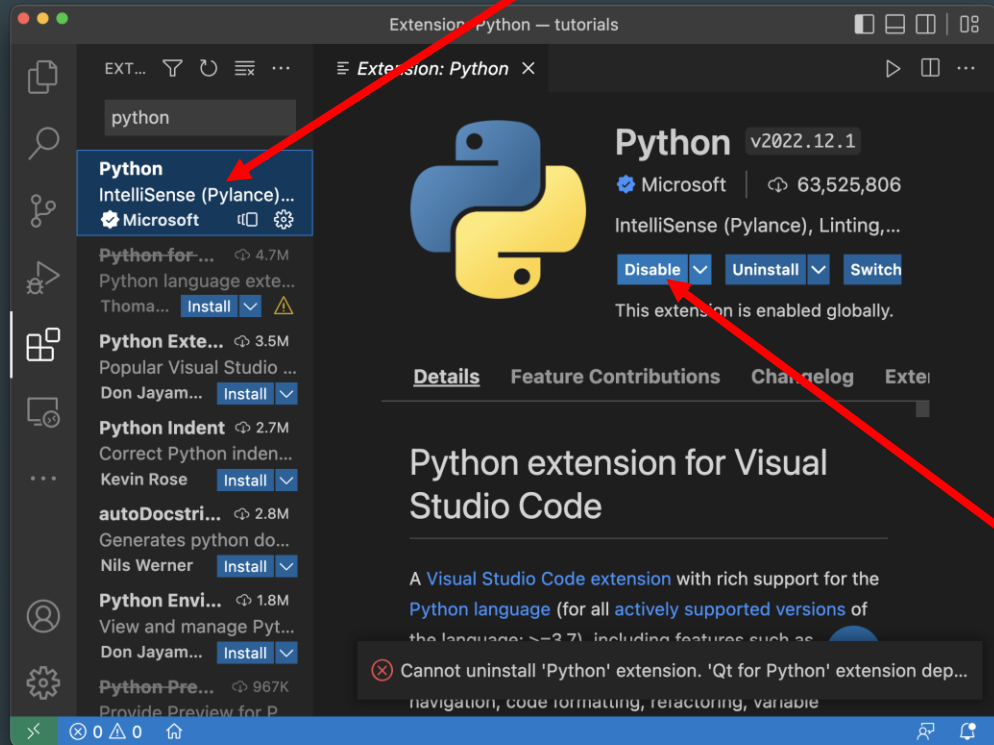


# Python Extension



1. Select the “Extensions” tab
2. Search for the “Python” extension

# Python Extension

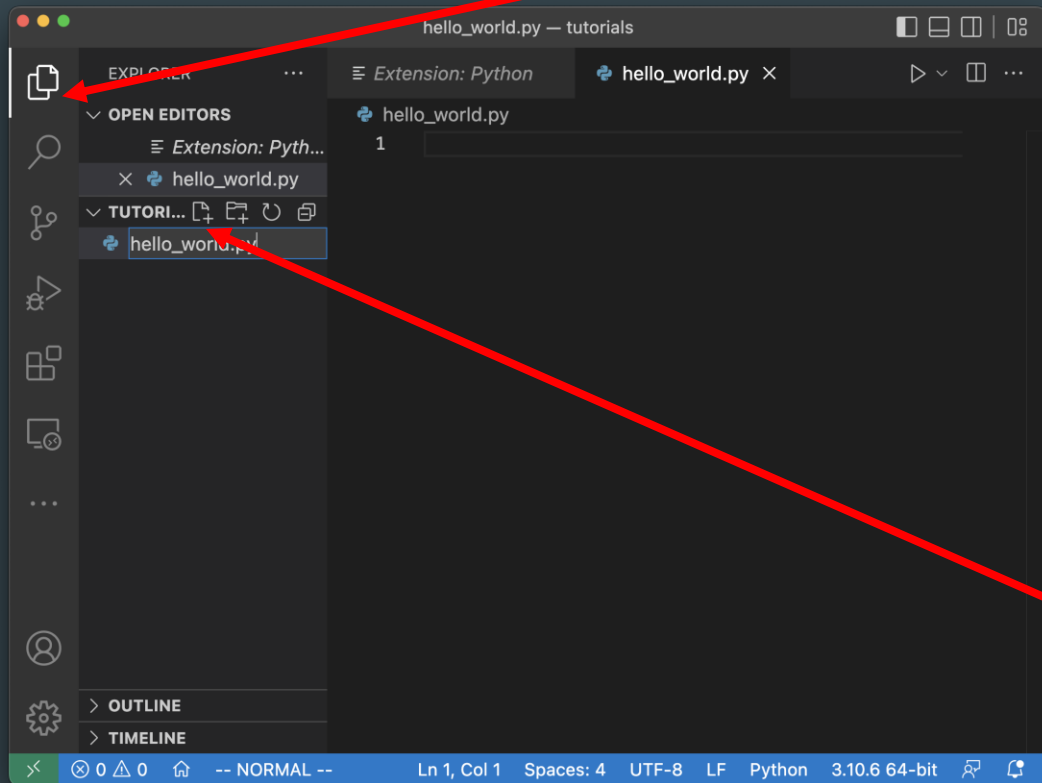


3. Select the first result

4. Click the “Install” button

# Create a New File

5



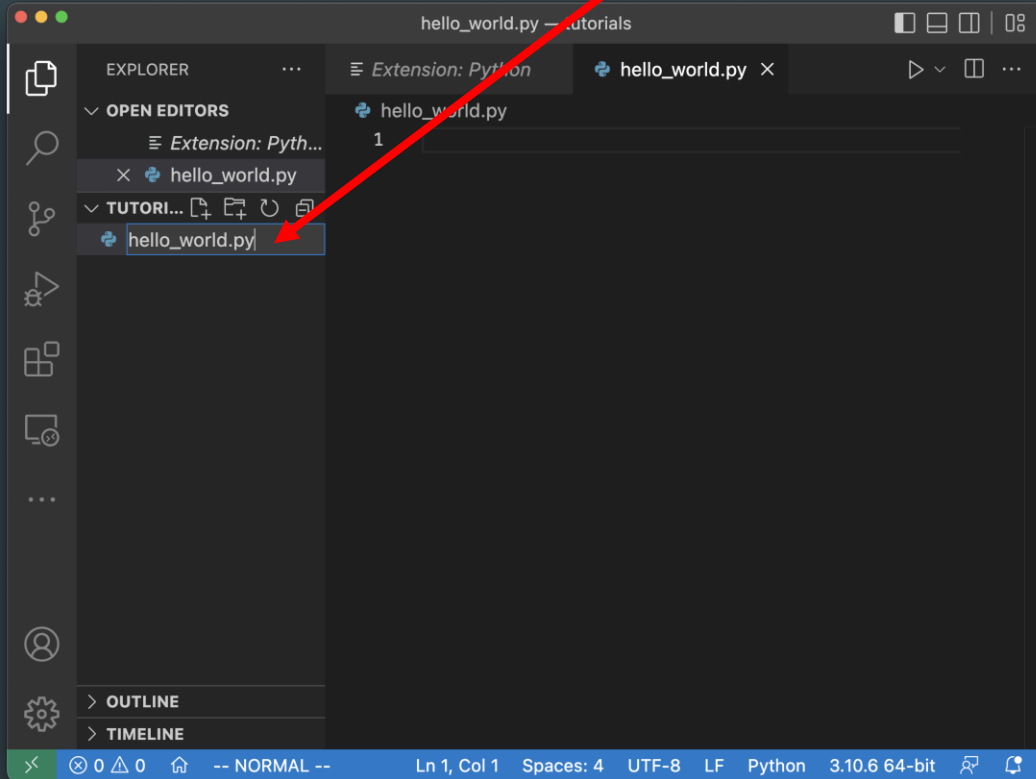
5. Go back to the “Files” tab

6. Add a new file to your project

6

# Create a New File

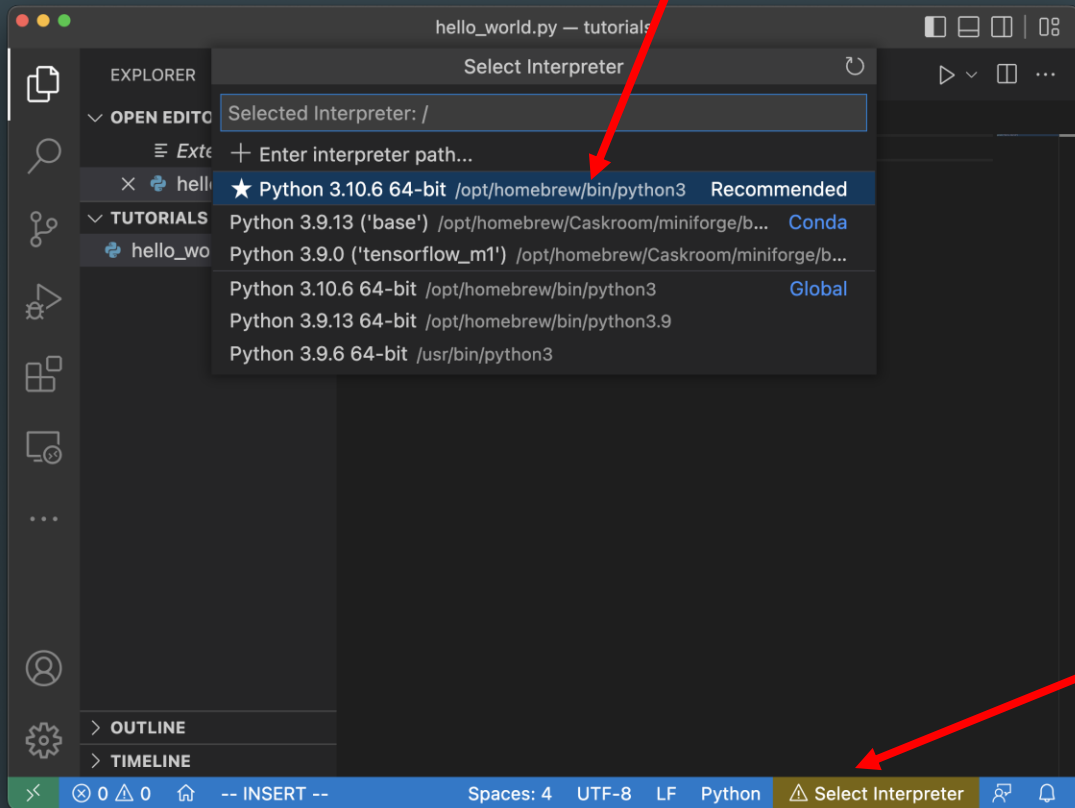
7



7. Name your file with a “.py” extension

# Select an Interpreter

9



8. Click the “Select Interpreter” button

9. Select the interpreter you installed earlier

8



# Run "Hello World!"

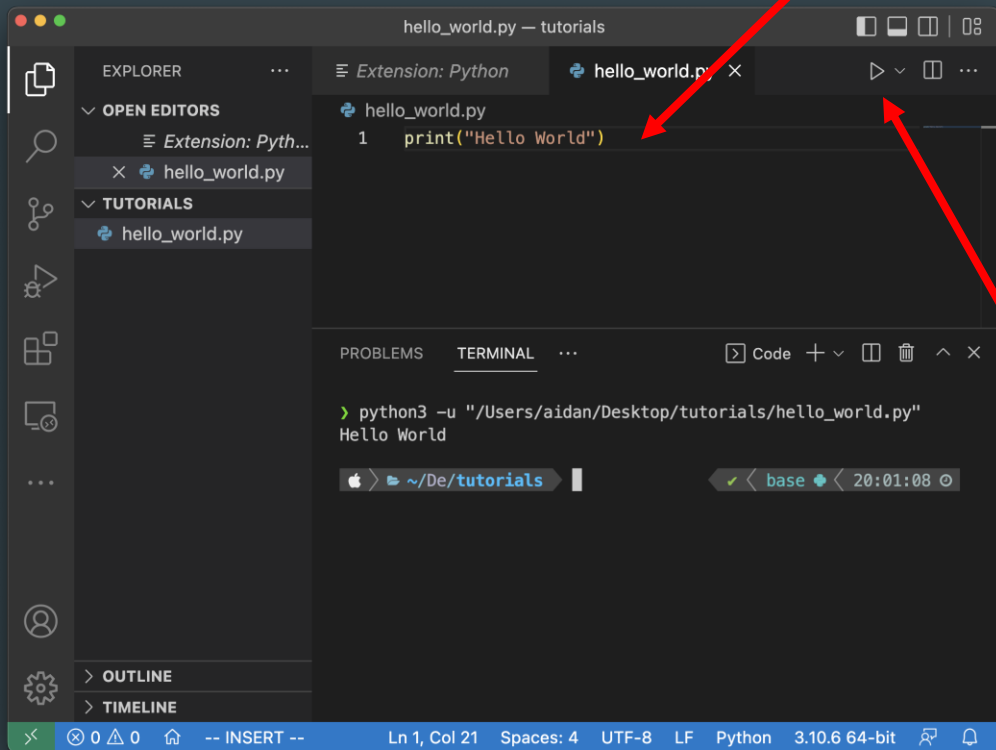
# 10

## 10. Type this statement:

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

## 11. Click the "run" button

# 11



**NOW LET'S GET TO  
PROGRAMMING**

# Variables

Let's look at 4 common variable types

- Strings
- Integers
- Floats
- Booleans

See Python Built-in Types:

<https://docs.python.org/3/library/stdtypes.html>

# Declaring Variables

Strings are denoted by " or '

*name = "Avery"*

Integers are denoted by a number without a decimal

*age = 19*

Floats/Doubles are denoted by a number with a decimal

*bankAccount = 125.43*

Booleans are denoted by True or False

*single = True*

# Testing Variable Types

```
print(type(variable_name))
```

```
print(type(name))
```

Output: <class 'str'>

# Variable Names

In the world of programming, there are two common ways of naming variables:

Camel Case

- myVariable

Snake Case

- my\_variable

# Math Operations

Now let's focus on mathematical operations:

It is the basics of programming

- Add +
- Subtract -
- Multiply \*
- Divide /
- Integer Division //
- Remainders %
- Exponents \*\*

<https://docs.python.org/3/library/operator.html#module-operator>

# Math with Variables

*a = 8*

*b = 6*

*print(a + b) # add*

*print(a - b) # subtract*

*print(a \* b) # multiply*

*print(a / b) # divide*

*print(a // b) # floor division, rounds down*

*print(a % b) # modulo, returns remainder*

*print(a \*\* b) # power, exponents*

## Outputs

14

2

48

1.3333333

1

2

262144



## Exercise 1 – Evaluating a Function:

- Use a python program to evaluate the following equation for  $f(4)$ :

$$f(x) = \frac{12x^4 - 4x^2 + 9}{x^5 - 31}$$

## Possible Answer

$x = 4$

$equation = (12 * x^{**} 4 - 4 * x^{**} 2 + 9) / (x^{**} 5 - 31)$

$print(equation)$

3.0382678751258814

## Increment Math Operations

Say we want to adjust the value of a variable using itself

*$a = 6$*

*$a = a + 5$*

*$print(a)$*

# Increment Math Operations

- Add +=
- Subtract -=
- Multiply \*=
- Divide /=
- Integer Division //=
- Remainders %=
- Exponents \*\*=

[https://www.w3schools.com/python/gloss\\_python\\_assignment\\_operators.asp](https://www.w3schools.com/python/gloss_python_assignment_operators.asp)

# Incrementation

```
n = 10
```

```
print(n)
```

```
10
```

```
print(n+1)
```

```
print(n)
```

```
11
```

```
10
```

```
n = n+1
```

```
print(n)
```

```
11
```

```
n += 1
```

```
print(n)
```

```
12
```

```
n /= 2
```

```
print(n)
```

```
6.0
```

# *String Operations*

Now let's focus on strings:

Can concatenate or format.

<https://docs.python.org/3/tutorial/inputoutput.html>

# Using only print statements

```
first = "Avery"  
second = "Fernandez"  
print("My name is ")  
print(first)  
print(second)
```

**My name is**  
**Avery**  
**Fernandez**

Each print statement is on its own line

# Formatting

```
print("My name is", first, second)
```

**My name is Avery Fernandez**

```
print(f"My name is {first} {second}")
```

**My name is Avery Fernandez**



# Concatenation

```
print("My name is " + first + " " + second)
```

**My name is Avery Fernandez**

```
age = 19
```

```
print("My age is " + age)
```

receive an error, can only concatenate str (not "int")

# Casting Variables

```
print(type(age))  
<class 'int'>
```

We can't concatenate since the variable is an integer

```
ageString = str(age)  
print(type(ageString))  
<class 'str'>
```

Now we can concatenate

```
print("My age is " + ageString)  
print("My age is " + str(age))
```

**My age is 19**

**My age is 19**

Works for all data types

```
print(f"Examples: {str(age)} {str(bankAccount)} {str(single)}")
```

**Examples: 19 125.43 True**

# Taking User Inputs

How can we input our own data

<https://docs.python.org/3/library/2to3.html?highlight=input#to3fixer-input>

# Inputs

```
name = input("What is our name")  
print(name)
```

```
print(type(name))  
<class 'str'>
```

```
number = input("Give me a number")  
print(number)  
print(type(number))
```

```
<class 'str'>
```

```
number = int(input("Give me a number"))  
print(number)  
print(type(number))  
<class 'int'>
```

# Booleans and Comparisons

Booleans can be used to evaluate statements that, for example, are either True or False

Statements like: (is 2 even )would be **True**

Booleans can either be displayed with **True** and **False**

or

**1** and **0**

Now this moves us into if and else statements.

<https://docs.python.org/3/library/stdtypes.html#builtin-boolean-values>

<http://swcarpentry.github.io/python-novice-gapminder/13-conditionals/index.html>

# Boolean Statements

```
apple = True  
if apple:  
    print("Hi")  
else:  
    print("Rawr")
```

Hi

```
apple = 1  
if apple:  
    print("Hi")  
else:  
    print("Rarw")
```

Hi

# Some Boolean and Comparison Operators

- `==`
- `<`
- `>`
- `<=`
- `>=`
- `!=`

- *and*
- *or*
- *not*

It will output True or False based on the statement

<https://docs.python.org/3/library/stdtypes.html#boolean-operations-and-or-not>

# Boolean Examples

```
print(1>0)
```

**True**

```
print(10!=5)
```

**True**

```
print(4<3)
```

**False**

```
print(4 > 3 and 3 > 4)
```

**False**

```
user = int(input("Enter a number"))
```

```
if user > 50:
```

```
    print("Number is greater than 50")
```

```
else:
```

```
    print("Number is less than 50")
```



## Exercise 2 – Even/Odd numbers:

Make a program which lets a user input a number, and tells them if the number is even or odd

Hint: %

# Possible Answers

```
user = int(input("Enter a number"))  
if user % 2:  
    print("Odd Number")  
else:  
    print("Even Number")
```

```
user = int(input("Enter a number"))  
if user % 2 == 0:  
    print("Even Number")  
else:  
    print("Odd Number")
```

# For Loops

Do repeated operations

<https://nbviewer.org/github/jakevdp/WhirlwindTourOfPython/blob/master/07-Control-Flow-Statements.ipynb>

# Basic For Loop: in range()

```
for i in range(10):  
    print(i)
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

# For Loops Syntax: in range()

*for (variable) in range(starting number, end number, increment):*

*for i in range(1, 10):*  
*print(i)*

1  
2  
3  
4  
5  
6  
7  
8  
9

# Full Syntax Example

```
for i in range(0, 10, 3):  
    print(i)
```

```
0  
3  
6  
9
```

## Exercise 3 – Factorials:

Make a program which lets a user input a number  $n$  and evaluates:

$$n! = 1 * 2 * \cdots * (n - 1) * n$$

Hint: Use a storage variable  
 $0 * \text{any number} = 0$

## Possible Answers

*n=1*

*user = int(input("Enter a number"))*

*for i in range(1, user+1):*

*n\*=i*

*print(n)*



# While Loops

Uses Boolean argument

*While (true statement):*

<https://nbviewer.org/github/jakevdp/WhirlwindTourOfPython/blob/master/07-Control-Flow-Statements.ipynb>

# While Loop Example

```
i = 0
while True:
    print("Hi", i)
    i += 1
    if (i > 10):
        break
```

Hi 0  
Hi 1  
Hi 2  
Hi 3  
Hi 4  
Hi 5  
Hi 6  
Hi 7  
Hi 8  
Hi 9  
Hi 10

```
while (input("name?") != "Avery"):
    print("Not my owner")
```

```
i = 1
while (i < 10 and i > 0):
    print(i)
    i+=1
```

1  
2  
3  
4  
5  
6  
7  
8  
9

## Exercise 4 – Least Common Multiples:

Find the first number which is divisible by 1, 2, 3, 4, 5, 6, and 7 – in other words, the least common multiple of those numbers

# Possible Solutions

```
i = 1
while (True):
    if i % 1 == 0 and i % 2 == 0 and i % 3 == 0 and i % 4 == 0 and i % 5 == 0 and i % 6 == 0 and i % 7 == 0:
        break;
    i += 1
print(i)
```

```
i = 1
while not (i % 1 == 0 and i % 2 == 0 and i % 3 == 0 and i % 4 == 0 and i % 5 == 0 and i % 6 == 0 and i % 7 == 0):
    i += 1
print(i)
```

420

THANKS SO MUCH



Jupyter Notebook