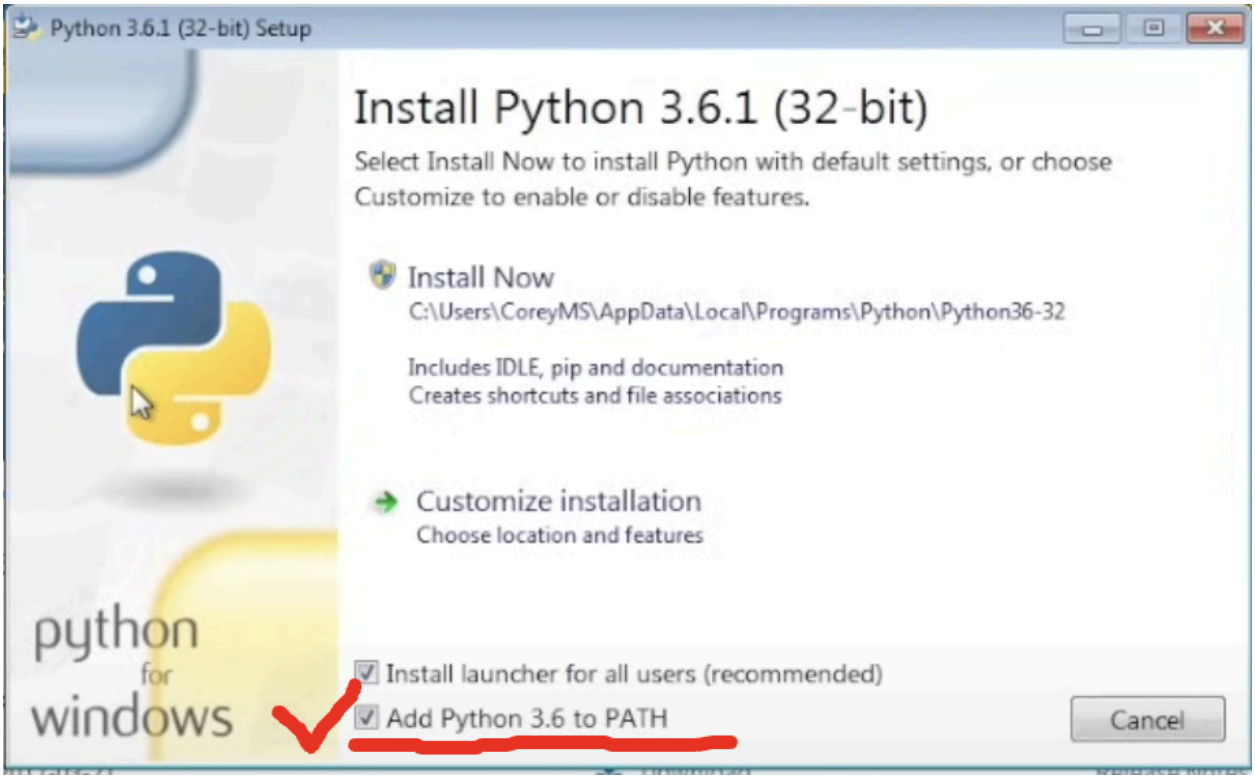
# Welcome to the Python World

## Download and Insatll Python:

https://www.python.org/downloads/

## Windows Users:




## Check Python Version:

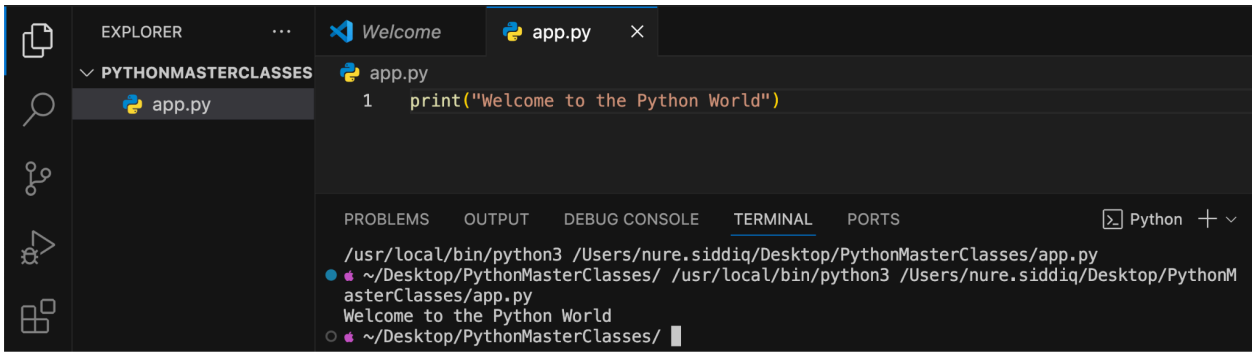
python --version or python3 --version or  python -V

## bongoDev Python Play List:

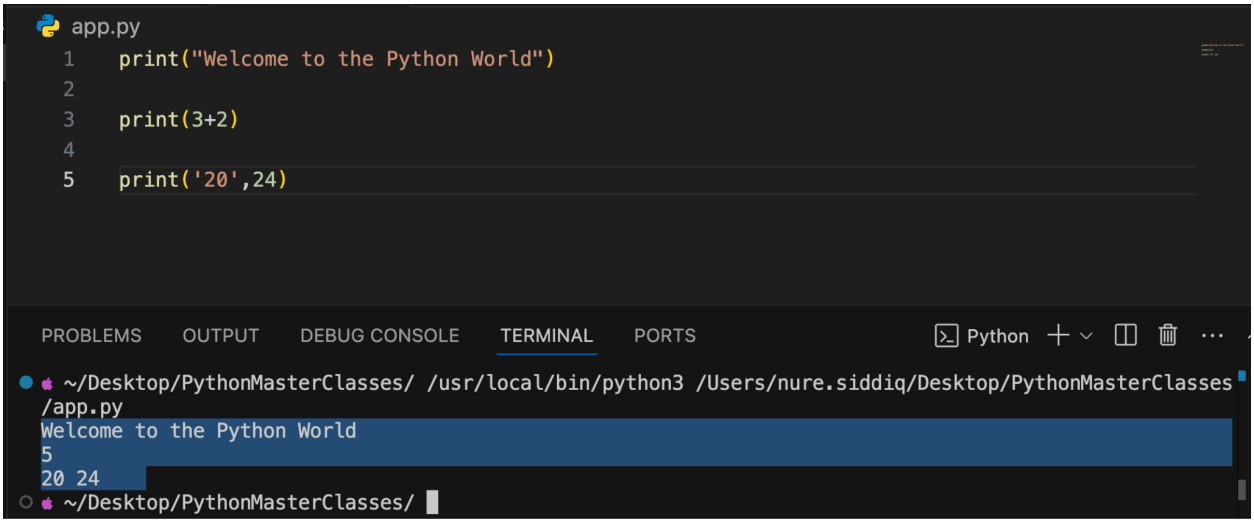
https://www.youtube.com/watch?v=mjYXQxDSQds&list=PL4VsqV8BmJN-hfUap967qOtflDM3gdufY

## Enable Python in VS Code:


## First Python Program:




## Python RUNS Line by Line (Top to bottom)



## Data Types:

```
app.py
1   print(type("Welcome to the Python World"))
2
3   print(type(3))
4
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

~/Desktop/PythonMasterClasses/ /usr/local/bin/python3 /Users/
/app.py
<class 'str'>
<class 'int'>
~/Desktop/PythonMasterClasses/
```

## Variable:

Is a holder of data that holds/store a value and we can use multiple times
name = 'Nure Siddiq'  (Is a Python statement)
name is variable name
= is an assignment operator which assigns value 'Nure Siddiq' to variable name
'Nure Siddiq' is the value

```
app.py > ...
1   name = 'Nure Siddiq'
2
3   print(name)
4
5   age = 30
6   print(age)
7
PROBLEMS  OUTPUT  DEBUG CONSOLE  T

~/Desktop/PythonMasterClasses/ /usr/lo
/app.py
Nure Siddiq
30
~/Desktop/PythonMasterClasses/
```

## Important Features in Python:

- [ ] High level programming language (Like plain English)
- [ ] Dynamically typed language (No need to define data type)
- [ ] Huge popular in task automation (Scripting language)
- [ ] Python is both interpreted and compiled
- [ ] Case sensitive language (Nure and nure are different)

## Operators In Python:

### 1. Arithmetic Operators

- `+` : Addition
- `-` : Subtraction
- `*` : Multiplication
- `/` : Division (float division)
- `//` : Floor Division
- `%` : Modulus
- `**` : Exponentiation

## 2. Comparison Operators

- `==` : Equal to
- `!=` : Not equal to
- `>` : Greater than
- `<` : Less than
- `>=` : Greater than or equal to
- `<=` : Less than or equal to

```
# Variables for comparison
a = 10
b = 5
c = 10

# 1. Equal to (==)
print(a == c)   # True, because 10 is equal to 10

# 2. Not equal to (!=)
print(a != b)   # True, because 10 is not equal to 5

# 3. Greater than (>)
print(a > b)    # True, because 10 is greater than 5

# 4. Less than (<)
print(b < a)    # True, because 5 is less than 10

# 5. Greater than or equal to (>=)
print(a >= c)   # True, because 10 is equal to 10

# 6. Less than or equal to (<=)
print(b <= c)   # True, because 5 is less than or equal to 10
```

## 3. Logical Operators

- `and` : Returns `True` if both operands are true
- `or` : Returns `True` if at least one operand is true
- `not` : Returns `True` if operand is false

```
# Variables for logical operations
x = True
y = False
a = 10
b = 5

# 1. 'and' operator
# Returns True only if both conditions are True
print(x and (a > b))   # True, because both x is True and 10 > 5

print(y and (a > b))   # False, because y is False

# 2. 'or' operator
# Returns True if at least one condition is True
print(x or y)          # True, because x is True

print(y or (b > a))    # False, because both y is False and 5 is not greater than 10

# 3. 'not' operator
# Returns True if operand is False
print(not x)           # False, because x is True

print(not y)           # True, because y is False
```

## 4. Membership Operators

- `in` : Returns `True` if a value is found in a sequence
- `not in` : Returns `True` if a value is not found in a sequence

```
# Example sequence (list)

fruits = ["apple", "banana", "cherry"]

# 1. 'in' operator
# Returns True if the element is in the list
print("banana" in fruits)      # True, because "banana" is in the list

print("grape" in fruits)       # False, because "grape" is not in the list

# 2. 'not in' operator
# Returns True if the element is not in the list
print("grape" not in fruits)   # True, because "grape" is not in the list

print("apple" not in fruits)   # False, because "apple" is in the list
```

## 5. Identity Operators

- `is` : Returns `True` if both variables point to the same object
- `is not` : Returns `True` if both variables do not point to the same object

```
# Variables for identity operations
a = [1, 2, 3]
b = a          # 'b' is assigned to the same object as 'a'
c = [1, 2, 3]  # 'c' is a different object with the same content as 'a'

# 1. 'is' operator
# Returns True if both variables point to the same object
print(a is b)       # True, because 'b' is assigned to the same object as 'a'

print(a is c)       # False, because 'a' and 'c' have the same content but are different objects

# 2. 'is not' operator
# Returns True if both variables do not point to the same object
print(a is not c)   # True, because 'a' and 'c' are different objects

print(a is not b)   # False, because 'a' and 'b' point to the same object
```

## User Input: (Always STRING - str data type)

```python
num_1 = input("Type first number: ") #user input is always STRING (str) data type

num_2 = input("Type second number: ") #user input is always STRING (str) data type

print(num_1 + num_2)    #sum
print(num_1 - num_2)    #Subtract
print(num_1 * num_2)    #Multiply
print(num_1 / num_2)    #Devide
print(num_1 % num_2)    #Reminder (mod / modulus)
print(num_1 ** num_2)   #Power
```

```
~/Desktop/PythonMasterClasses/ /usr/local/bin/python3 /Users/nure.siddiq/Desktop/PythonMasterClasses
/app.py
Type first number: 5
Type second number: 3
53
Traceback (most recent call last):
  File "/Users/nure.siddiq/Desktop/PythonMasterClasses/app.py", line 6, in <module>
    print(num_1 - num_2)    #Subtract
          ~~~~~~^~~~~~
TypeError: unsupported operand type(s) for -: 'str' and 'str'
~/Desktop/PythonMasterClasses/
```

## Type Casting: (Convert one data type to another)

Needed to convert user input **str** to **int**

```python
num_2 = int(input("Type second number: ")) #Converted string to integer

print(num_1 + num_2)    #sum
print(num_1 - num_2)    #Subtract
print(num_1 * num_2)    #Multiply
print(num_1 / num_2)    #Devide
print(num_1 % num_2)    #Reminder (mod / modulus)
print(num_1 ** num_2)   #Power
```

```
~/Desktop/PythonMasterClasses/ /usr/local/bin/python3 /Users/nure.siddiq/Desktop/PythonMasterC
/app.py
Type first number: 5
Type second number: 3
8
2
15
1.6666666666666667
2
125
~/Desktop/PythonMasterClasses/
```

## Data Types In Detail:

```
13  print(type('Hello Nure'))
14  print(type(25))
15  print(type(12.07))
16  print(type(True))
17
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

~/Desktop/PythonMasterClasses/ /usr/local/bin/python3 /Us
/app.py
~/Desktop/PythonMasterClasses/ /usr/local/bin/python3 /Us
/app.py
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
~/Desktop/PythonMasterClasses/
```

- [ ] String (str): Textual data

```
'Hello'or "Hello" or """Hello"""

"""
Line 1
line 2
line 3

"""
```

- [ ] Integer (int) - number 3, 4, 5

- [ ] Float - decimal number

```
>>> type(13.123)
<class 'float'>
```

- [ ] Boolean (True, False)

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

```
>>> 5 > 3
True
>>> 13 < 5
False
>>>
```

## Conditions / logics in Python:

In Python, conditions are used to perform different actions based on whether a condition is `True` or `False`. Python commonly uses `if`, `elif`, and `else` statements for conditional checks.

### Example: if and else:

```
temperature = 30

if temperature > 25:
    print("It's a warm day.")
else:
    print("It's a cool day.")
```

Output: It's a warm day.

### Example: if, elif, and else:

```
score = 85

if score >= 90:
    print("Excellent!")

elif score >= 70:
    print("Good job!")

else:
    print("You can improve.")
```

## Solve the Problems:

### Problem1:

Write a Python program that takes a string input from the user and checks if Enjoy available in the input string:

"Hello, world! Hello everyone. Welcome to the world of Python. Enjoy coding in Python."

### Problem2:

Write a Python program that takes an input of average marks from the user and then categorizes the grade as follows:

- If marks are greater than or equal to 90, the grade is "A+."
- If marks are less than 90 but greater than or equal to 70, the grade is "A-."
- If marks are less than 70 but greater than or equal to 50, the grade is "B."
- If marks are less than 50, the grade is "Fail."

### Problem3:

Write a Python program that takes a single integer `n` as input from the user. The program should output:

- `"Fizz"` if `n` is a multiple of 3.
- `"Buzz"` if `n` is a multiple of 5.
- `"FizzBuzz"` if `n` is a multiple of both 3 and 5.
- Otherwise, output not a FizzBuzz number.

## Problem4:

Write a calculator program that takes three inputs from the user:

1. **Input1**: A number (float or integer).
2. **Input2**: A number (float or integer).
3. **Operator**: A character representing a mathematical operation. The operator can be one of the following: `+`, `-`, `*`, `/`, or `%`.

The program should perform the following tasks:

- Validate the inputs to ensure that `Input1` and `Input2` are valid numbers and that the `Operator` is one of the specified characters.
- Use conditional statements to determine which operation to perform based on the `Operator` provided.
- If the operator is `+`, return the sum of `Input1` and `Input2`.
- If the operator is `-`, return the difference between `Input1` and `Input2`.
- If the operator is `*`, return the product of `Input1` and `Input2`.
- If the operator is `/`, return the quotient of `Input1` divided by `Input2`. If `Input2` is zero, return an appropriate error message indicating that division by zero is not allowed.
- If the operator is `%`, return the remainder of `Input1` divided by `Input2`.
- Display the result of the operation to the user.

### Example Input/Output:

1. Input: `Input1 = 10`, `Input2 = 5`, `Operator = +`
   Output: `The result is 15`
2. Input: `Input1 = 10`, `Input2 = 0`, `Operator = /`
   Output: `Error: Division by zero is not allowed.`

**Cheers!**