Python CODES

*Try **Python Tutor** online if you don't have a Python IDE to run codes. Also, go to **w3schools** for more in-depth stuff*

# This is a comment. Comments are not executed. It is written by placing a "#" sign at the beginning of a line.

# Variables are used to store data.

name = "Bard"

# Print statements are used to output text to the console.

print("Hello, world!")

# Data types are used to specify the type of data that a variable can store.

my_integer = 10	*positive/negative numbers*
my_float = 3.14	*decimal*
my_string = "Hello, world!"	*texts, enclosed by " "*
my_boolean = True	*True or False*

# Casting is basically used to convert data types.

my_cast1 = int("10")	*convert string to integer.*
my_cast2 = str(10)	*convert integer to string. etc.*

# Arithmetic operators are used to perform mathematical operations.

sum = 1 + 2

difference = 10 - 5

product = 2 * 3

quotient = 10 / 2

modulo = 10 % 3	*for remainder*
power = 5 ** 4	*for exponents*

# Lists are used to store collections of data in a specific order.

position is 0 1 2 3 4	*can also be read position as -5 -4 -3 -2 -1*
my_list = [1, 2, 3, 4, 5]	**my_list = [1, 2, 3, 4, 5]**

# Tuples are like lists, but they are immutable, meaning that they cannot be changed once they are created.

my_tuple = (1, 2, 3, 4, 5)

# Dictionaries are used to store key-value pairs of data.

my_dictionary = {"name": "Bard", "age": 2}

# Getting user's input.

user_input = input()

# Indentation

Indentation is important to know which code belongs to a certain part.

# Conditional operators

== *equal to,* **!=** *not equal to,* **<** *less than,* **<=** *less than or equal to*

> *greater than,* **>=** *greater than or equal to*

# Conditional statements are used to control the flow of execution of a program.

if = 1 condition, if-else = 2 conditions, if-elif-else = 3+ conditions

```
if sum > difference:
   print("The sum is greater than the difference.")
else:
   print("The sum is not greater than the difference.")
```

# Logical operators

and *returns True only if both of its operands are True.*

```
print(True and True)  # True
print(True and False)  # False
```

or *returns True if one of its operands is True*

```
print(False or False)  # False
print(True or False)  # True
```

not reverses the truth value of its operand

```
print(not True)  # False
print(not False)  # True
```

# Loops are used to repeat a block of code until a certain condition is met.

# For loops are used to iterate over a sequence of items, such as a list, tuple, or string.

Syntax:

```
for <variable> in <sequence>:
    <code block>
```

```
my_list = [1, 2, 3, 4, 5]
for number in my_list:
   print(number)
```

# While loops are used to execute a block of code while a condition is true.

Syntax:

while <condition>:

<code block>

```
count = 1
while count <= 5:
   print(count)
   count += 1
```

# Nested loops are loops inside of other loops. Can be used to perform more complex tasks.

```
my_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for row in my_list:
   for number in row:
      print(number)
```

# Functions are used to group code together and perform a specific task.

Syntax:

def <function_name>(<parameter/s>):

<code block>

```
def add_numbers(num1, num2):
   return num1 + num2
```

used to get result by returning values

# To call a function, you simply use the function name followed by parentheses and inside are the values

Continuation of above code

```
add_numbers(1, 2)
```