"Hello, World!" Program

In [1]:
```
print("Hello, World!")
```

```
Hello, World!
```

Variables

In [2]:
```
x = 5
print("x =", x)

x = "Five"
print("x =", x)
```

```
x = 5
x = Five
```

In [3]:
```
print('-' * 50)
```

```
--------------------------------------------------
```

Operators

In [4]:
```
x = 10
y = 4

print('x + y =', x+y)

print('x - y =', x-y)

print('x * y =', x*y)

print('x / y =', x/y)

print('x // y =', x//y)

print('x % y =', x%y)

print('x ** y =', x**y)
```

```
x + y = 14
x - y = 6
x * y = 40
x / y = 2.5
x // y = 2
x % y = 2
x ** y = 10000
```

In [5]:
```
x = 10

x+=1 #not x++
print(x)

x+=5
print(x)

x /= 5
print(x)
```

```
11
16
3.2
```

User Input

In [6]:
```
inputString = input('Enter a string:')
print('You\'ve entered:', inputString)
```

```
Enter a string:abc
You've entered: abc
```

Comments

In [7]:
```
# This is a comment

"""This is a
    multiline
    comment."""

'''This is also a
 multiline
 comment.'''
```

Out[7]:
```
'This is also a\n multiline\n comment.'
```

Type Conversion

In [8]:
```
var_int = 123  # Int
var_flo = 1.23  # Float

var_new = var_int + var_flo

print("value of var_new:",var_new)
print("type of var_new:",type(var_new))
```

```
value of var_new: 124.23
type of var_new: <class 'float'>
```

In [9]:
```
num_int = 123     # int
num_str = "456"   # str

#print(num_int + num_str)  # FAILS
print(num_int, num_str)
print(str(num_int) + num_str)
print((num_int) + int(num_str))
```

```
123 456
123456
579
```

Python Numeric Types

In [10]:
```
print(type(3))

print(type(3.0))

print(type(2 + 3j))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

Lists

In [11]:
```
language = ["French", "German", "English", "Polish"]

print("Type: ", type(language))

# 1st element
print(language[0])


# # 3rd element
print(language[2])

language[0] = "Italian"
print(language[0])

language[1:3]

# print(language)

# #remove second element
language.pop(0)
print(language)


language[-2]
```

```
Type:  <class 'list'>
French
English
Italian
['German', 'English', 'Polish']
```

Out[11]:
```
'English'
```

In [12]:
```
dir(language)
```

Out[12]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

Tuples

In [13]:
```
language = ("French", "German", "English", "Polish")

print(language[1])
print(language[3])
print(language[-1])
print(language[-2])


language[0] = "Italian"
```

```
German
Polish
Polish
English

---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-13-305a9751cd53> in <module>
      7
      8
----> 9 language[0] = "Italian"

TypeError: 'tuple' object does not support item assignment
```

String

In [14]:
```
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string for multiple lines
my_string = """Hello, welcome to
           the world of Python"""
print(my_string)


my_string = "hellodsfsdfmsklmewfewlkfnwelkfmw"\
"elkfmwe fnwelkfmwelkfmwe fnwelkfmwelkfmwe fnwelkf"\
"mwelkfmwe fnwelkfmwelkfmwe fkew fwefeworld"

my_string = """hellodsfsdfmsklmewfewlkfnwelkfmw
elkfmwe fnwelkfmwelkfmwe fnwelkfmwelkfmwe fnwelkf
mwelkfmwe fnwelkfmwelkfmwe fkew fwefeworld"""
print(my_string)
```

```
Hello
Hello
Hello
Hello, welcome to
           the world of Python
hellodsfsdfmsklmewfewlkfnwelkfmw
elkfmwe fnwelkfmwelkfmwe fnwelkfmwelkfmwe fnwelkf
mwelkfmwe fnwelkfmwelkfmwe fkew fwefeworld
```

In [15]:
```
# slicing
str = 'python'
print('str = ', str)

print('str[0] = ', str[0])

print('str[-1] = ', str[-1])

print('str[1:4] = ', str[1:4])

print('str[2:-2] = ', str[2:-2])
```

```
str =  python
str[0] =  p
str[-1] =  n
str[1:4] =  yth
str[2:-2] =  th
```

In [16]:
```
str1 = 'Hello '
str2 ='World!'

print(str1 + str2)

print("*" * 30)
```

```
Hello World!
******************************
```

Sets

In [17]:
```
# int set
my_set = {1, 2, 3}
print(my_set)

# set of mixed types
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

```
{1, 2, 3}
{1.0, (1, 2, 3), 'Hello'}
```

In [18]:
```
# int set
my_set = {1, 2, 3}

my_set.add(4)
print(my_set)

my_set.add(2)
my_set.add(2)
my_set.add(2)
my_set.add(2)
print(my_set)

my_set.update([3, 4, 5])
print(my_set)

my_set.remove(4)
print(my_set)
```

```
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4, 5}
{1, 2, 3, 5}
```

In [19]:
```
A = {1, 2, 3}
B = {2, 3, 4, 5}

# union
print(A | B) # 1,2,3,4,5

# intersection
print (A & B)  # 2, 3

# difference
print (A - B)  #1

# symmetric difference
print(A ^ B)  #1,4,5
```

```
{1, 2, 3, 4, 5}
{2, 3}
{1}
{1, 4, 5}
```

Dictionaries

In [20]:
```
# empty dict
my_dict = {}

# dict with int keys
my_dict = {1: 'apple', 2: 'pear'}

# dictionary with mixed keys
my_dict = {'name': 'Alice', 1: [2, 4, 3]}
```

In [21]:
```
peter = {'name':'Peter', 'age': 25, 'salary': 25000}
print(peter['age'])
```

```
25
```

In [22]:
```
person = {'name':'Oscar', 'age': 30}

print(person)

# Changing age to 36
person['age'] = 36
print(person)

# Adding height
person['height'] = 6
print(person)


# Deleting age
del person['age']
print(person)


for k,v in person.items():
    print('Key: ', k ,' Value: ' , v)
```

```
{'name': 'Oscar', 'age': 30}
{'name': 'Oscar', 'age': 36}
{'name': 'Oscar', 'age': 36, 'height': 6}
{'name': 'Oscar', 'height': 6}
Key:  name  Value:  Oscar
Key:  height  Value:  6
```

In [23]:
```
[x for x in person.keys()]
```

Out[23]:
```
['name', 'height']
```

range()

In [24]:
```
print(list(range(6)))

print(tuple(range(1,6)))

print(set(range(5,10)))
```

```
[0, 1, 2, 3, 4, 5]
(1, 2, 3, 4, 5)
{5, 6, 7, 8, 9}
```

In [25]:
```
# step range
print(list(range(1, 10 , 1)))

print(list(range(1, 10, 3)))

print(list( range(10, 0, -2)))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 4, 7]
[10, 8, 6, 4, 2]
```

if/else

In [26]:
```
num = -1

if num > 0:
    print("Positive")
elif num == 0:
    print("Zero")
else:
    print("Negative")
```

```
Negative
```

while loop

In [27]:
```
n = 10

sum = 0 #store sume
i = 1

while i <= n:
    sum += i
    i += 1    # increment

print("Sum =", sum)
```

```
Sum = 55
```

for loop

In [28]:
```
numbers = [1, 2, 3, 4, 5]

sum = 0

# iterate over the list
for x in numbers:
    sum += x

print("Sum =", sum)
```

```
Sum = 15
```

break statement

In [29]:
```
i = 0

while True:
    if i == 5:
        break
    print(i)
    i += 1
```

```
0
1
2
3
4
```

continue statement

In [30]:
```
for i in range(10):
    if i == 3:
        continue  # Skip
    print(i)
```

```
0
1
2
4
5
6
7
8
9
```

In [31]:
```
# and or in if showcase
for i in range(10):
    if i > 2 and i < 8:  #FAIL
        continue
    print(i)
```

```
0
1
2
8
9
```

---

pass statement

In [32]:
```
for i in range(3):
    pass  # IMPORTANT
```

Function

In [33]:
```
def add_numbers(a, b):
    """ Adds two numbers together"""

    # Calculate the total and return
    return a + b

print(add_numbers(6, 11))
```

```
17
```

Lambda Function

In [34]:
```
square_func = lambda x: x ** 2

print("Square =", square_func(5))


# add_numbers_lambda = lambda x,y: x + y

# print("Sum =", add_numbers_lambda(6,11))
```

```
Square = 25
```

Modules

In [35]:
```
import math

result = math.log10(100)
print(result)

print(math.pi)
```

```
2.0
3.141592653589793
```

Classes and Objects

In [36]:
```
class MyClass:
    """ This is a dummy class """
    a = 10

    def __init__(self, x):
        self.x = x

    def printx(self):
        print(self.x)

    def func(self):
        print('Hello')



print(MyClass.a)


# my_class_obj = MyClass(123)

# my_class_obj.func()


# print(MyClass.__doc__)

# my_class_obj = MyClass(123)
# my_class_obj.printx()


# my_class_obj2 = MyClass(456)
# my_class_obj2.printx()
```

```
10
```

In [37]:
```
class ComplexNumber:
    # constructor
    def __init__(self,r = 0, i = 0):
        self.real = r
        self.imag = i

    def getData(self):
        print("{0}+{1}j".format(self.real,self.imag))


c1 = ComplexNumber(2,3)
c1.getData()

c2 = ComplexNumber()
c2.getData()
```

```
2+3j
0+0j
```

In [ ]: