Python Programming for Robotics

>>python

or

>>python3

print("Hello")
print(4)

Keywords:

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	async
def	for	lambda	return	await

Operators

+	Addition	1 + 1 = 2
-	Subtraction	2 - 1 = 1
*	Multiplication	2 * 2 = 4
/	Division	5 / 2 = 2
%	Modulus	5 % 2 = 1

=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x − 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%	x %= 3	x = x % 3

Expressions
An expression is a combination of values, variables, and operators. A value all by itself is considered an expression, and so is a variable, so the following are all legal expressions.

Order of operations
When more than one operator appears in an expression, the order of evaluation depends on the rules of precedence. For mathematical operators, Python follows mathematical

convention. The acronym PEMDAS is a useful way to remember the rules: Parentheses, Exponentiation, Multiplication and Division.

Modulus operator
The modulus operator works on integers and yields the remainder when the first operand is divided by the second. In Python, the modulus operator is a percent sign (%).

String operations
The + operator works with strings, but it is not addition in the mathematical sense. Instead, it performs concatenation, which means joining the strings by linking them end to end.

Asking the user for input
Sometimes we would like to take the value for a variable from the user via their keyboard. Python provides a built-in function called input that gets input from the keyboard. When this function is called, the program stops and waits for the user to type something. When the user presses Return or Enter, the program resumes and input returns what the user typed as a string.

Comments
For this reason, it is a good idea to add notes to your programs to explain in natural language what the program is doing. These notes are called comments, and in Python they start with the # symbol.

Conditional execution
Boolean expressions
A Boolean expression is an expression that is either true or false. The following examples use the operator ==, which compares two operands and produces True if they are equal and False otherwise:

==	Equal	5 == 5
!=	Not Equal	4 != 5
>>	Greater than	5 >> 4
<	Less than	4 < 5
>=	Greater than or equal to	5 >= 4
<=	Less than or equal to	4 <= 5

if loop

```
i=1
if i==1:
    print('first')
elif i==2:
```

```
    print('second')
elif i==3:
    print('middle')
else:
    print('last')
```

while loop

```
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
print("ROS")
```

for loop

```
for x in range(5):
    print(x)
```

```
while (1):
    print('enter a digit')
    num=input()
    var=str(num)
    if (ord(var) in range (48,58)):
        break
print('you entered BCD')
```

```
for x in range(10):
    if x % 2 == 0:
        continue
    print(x)
```

Exception Handling in Python

```
print('num')
num=input()
print('den')
den=input()
try:
    res= int(num)/int(den)
```

```python
except:
    print("den cannot be 0")
else:
    print(res)
```

Functions

Built-in functions
Python provides a number of important built-in functions that we can use without needing to provide the function definition. The creators of Python wrote a set of functions to solve common problems and included them in Python for us to use.

```
int
len
max
min
type
float
str
```

Math functions

```python
import math
math.sin()
math.sqrt()
math.log10
```

Python Coding in Visual Studio

```
mkdir python_basics
cd python_basics
touch hello_world.py
chmod +x hello_world.py

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

Save the code
Open the integrated terminal

```
python3 hello_world.py
```

Replace the content by the following:

```python
def my_func(num):
    return num*2

def main():
        seq=[2,3,4,5,6,7]
```

```
        map(my_func,seq)
        a= list(map(my_func,seq))
        print(a)

if __name__ == "__main__":
 # execute only if run as a script
        main()
```

Save the code
Open the integrated terminal

python3 hello_world.py

Replace the content by the following:

Classes - Example 1


```
class Number:
        def __init__(self, val):
                self.val = val

obj = Number(2)
obj.val
```

Save the code
Open the integrated terminal

python3 hello_world.py

Replace the content by the following:

Classes - Example 2

```
#!/usr/bin/python3

class Point:
   def reset(self):
      self.x = 0
      self.y = 0
p = Point()
p.reset()
print(p.x, p.y)
```

Save the code
Open the integrated terminal

python3 hello_world.py
```

Replace the content by the following:

## Classes -  Example 3

```python
#!/usr/bin/python3

import math
class Point:
    def move(self, x, y):
        self.x = x
        self.y = y
    def reset(self):
        self.move(0, 0)
    def calculate_distance(self, other_point):
        return math.sqrt((self.x - other_point.x)**2 +
                    (self.y - other_point.y)**2)

point1 = Point()
point2 = Point()
point1.reset()
point2.move(5,0)
print(point2.calculate_distance(point1))
print(point1.calculate_distance(point2))
print(point1.calculate_distance(point1))
```

Save the code
Open the integrated terminal

python3 hello_world.py

Replace the content by the following:

Classes - Example

```python
#!/usr/bin/python3

import math
class Point:
    def __init__(self, x=0, y=0):
        '''
        #TODO
        '''
        self.move(x, y)
    def move(self, x, y):
        '''
        Move the point to a new location in 2D space
        '''
        self.x = x
        self.y = y
```

```python
    def reset(self):
        '''
        Reset the point back to the geometric origin: 0, 0
        '''
        self.move(0, 0)
    def calculate_distance(self, other_point):
        '''
        Calculate the distance from this point to a second
        point passed as a parameter.
        This function uses the Pythagorean Theorem to calculate
        the distance between the two points. The distance is
        returned as a float
        '''
        return math.sqrt(
        (self.x - other_point.x)**2 +
        (self.y - other_point.y)**2)

point1 = Point()
point2 = Point()
point1.reset()
point2.move(5,0)
print(point2.calculate_distance(point1))
print(point1.calculate_distance(point2))
print(point1.calculate_distance(point1))
```

Save the code
Open the integrated terminal

python3 hello_world.py

Practice the codes from the following Libraries:

 NumPy
 math
 random