CS LAB s.c.
ElectronicLaboratory

Instrukcja Użytkownika

simCNC

oprogramowanie sterujące

Przewodnik po skryptach Python

Spis treści

I. Python - podstawowe informacje

Oprogramowanie sterujące simCNC zostało wyposażone w obsługę skryptów, które pozwalają na zautomatyzowanie takich czynności, jak wymiana narzędzi, czy też pomiar długości narzędzia. Zastosowanie dla skryptów nie kończy się jednak na tych czynnościach i niekiedy zachodzi potrzeba stworzenia o wiele bardziej skomplikowanych skryptów. Aby ułatwić użytkownikom oprogramowania tworzenie własnych skryptów, simCNC korzysta z Pythona, jako języka skryptowego.

Python to język programowania wysokiego poziomu o ogólnym przeznaczeniu i rozbudowanym pakiecie bibliotek standardowych, którego cechą główną jest czytelność i klarowność kodu źródłowego. Składnia języka jest natomiast przejrzysta i zwięzła.

Po język Python sięgają zarówno początkujący, jak i profesjonaliści. Tak duża popularność tego języka, przyczynia się do tworzenia wielu tutoriali i bibliotek, które można znaleźć na niezliczonych stronach internetowych. Ten stan rzeczy powoduje, że jedyną barierą, jaka może nam stanąć na drodze do napisania własnego skryptu, są chęci i nasza wyobraźnia.

II. SimCNC oraz Python, co warto wiedzieć?

Podczas instalacji oprogramowania simCNC na dysku twardym, jako odrębne oprogramowanie jest instalowany Python. Oprogramowania simCNC komunikuje się z Python poprzez protokół internetowy UDP za pomocą specjalnie przygotowanej biblioteki „___COMM.pyc”. Poza tą biblioteką programiści przygotowali także bibliotekę „___DEVICE.pyc", która zawiera zbiór funkcji pozwalających na sterowanie oprogramowaniem simCNC z poziomu makr.

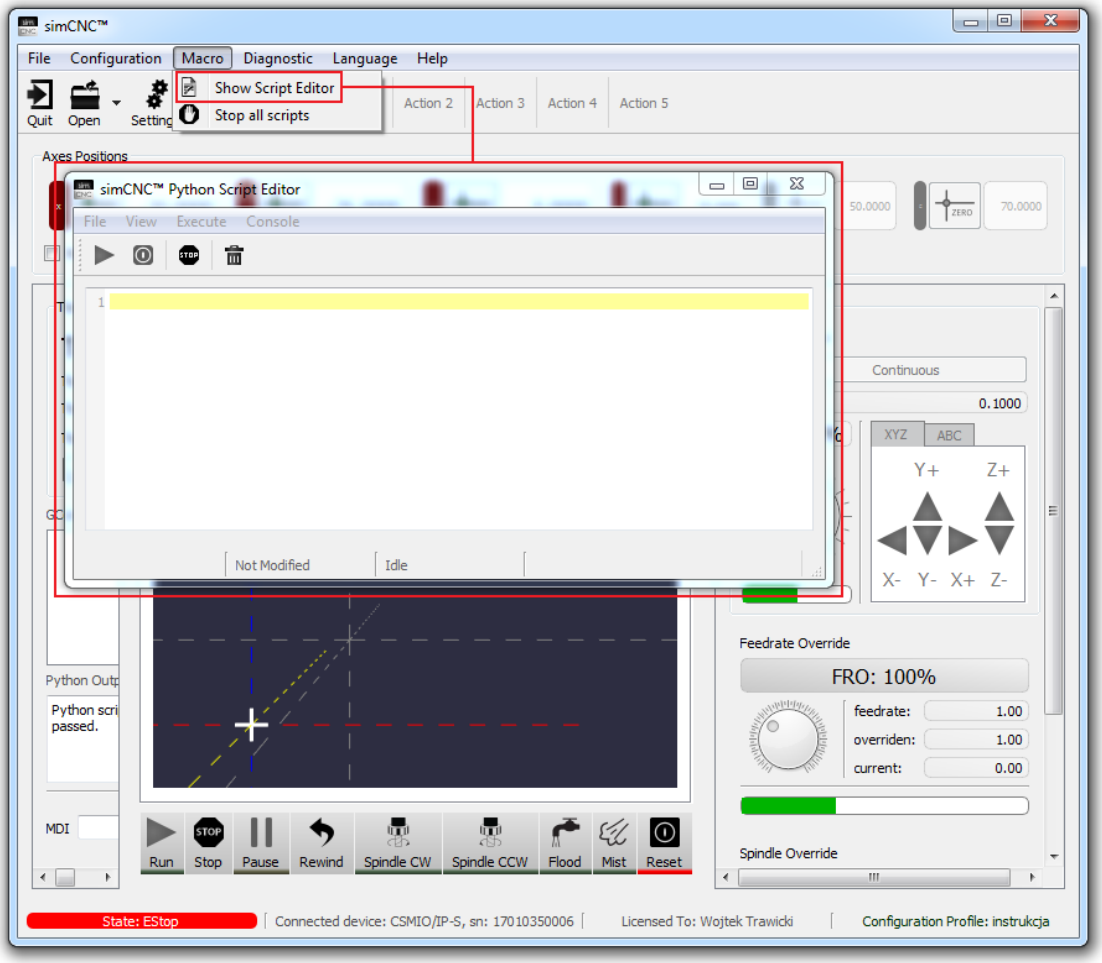
Rozwiązanie to pozwala na:

- używanie zawansowanego zewnętrznego edytora programistycznego z opcją debugowania np. Visual Studio Code.
- sterowanie oprogramowaniem simCNC poprzez UDP w lokalnej sieci z poziomu innego komputera PC.

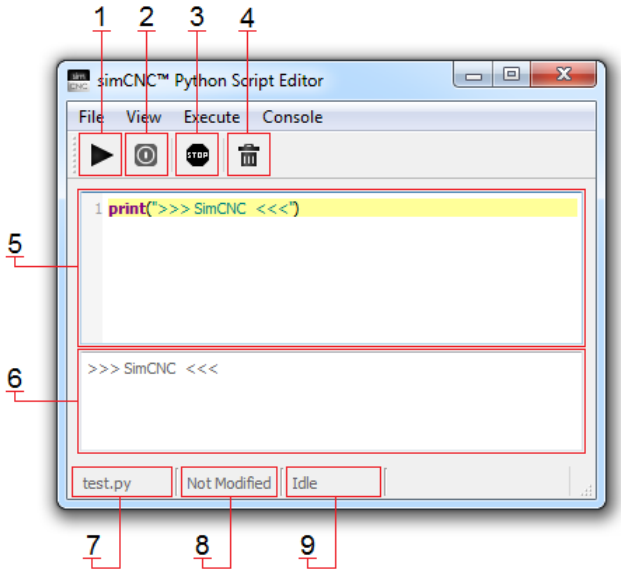
Oba powyższe rozwiązania omówione zostaną szerzej w dodatkowej dokumentacji. Niniejsza instrukcja odnosi się natomiast do edytora slryptów wbudowanego w oprogramowanie simCNC.

III. Uruchomienie i opis edytora skryptów Python

Aby uruchomić edytor skryptów Python należy na górnej belce okna oprogramowani simCNC kliknąć opcję „Macro” (Macra), następnie z rozwijanej listy wybrać opcję „Show Script Editor” (Pokaż edytor skryptów).



1. „Run” (Start) - uruchom skrypt.
2. „Stop” (Stop) - zatrzymaj skrypt.
3. „Stop Trajectory Planer” (Zatrzymaj planer ruchu) – odpowiednik przycisku „STOP” na głównym ekranie simCNC.
4. „Clear Console” (Wyczyść Konsolę) – usuwa zawartość okna konsoli Python (patrz punkt 6).
5. Edycja skryptu Python – w tym miejscu możemy tworzyć lub edytować skrypt.
6. Konsola Python – miejsce, w którym są wyświetlane informacje dotyczące skryptów (błędy i wyjątki), a także własne komunikaty (przykład pokazano obok).
7. Nazwa skryptu.
8. Informacja o tym czy skrypt od ostatniego zapisu był modyfikowane.
9. Informacja o stanie skryptu.



IV. Ctrl + Space – podpowiedzi w edytorze skryptów Python

Edytor skryptów Python wyposażono w mechanizm podpowiedzi, aby go uruchomić w oknie edytora wystarczy nacisnąć jednocześnie kombinacje klawiszy „Ctrl” + „Space”. Kombinację tą można zastosować w dowolnej chwili.

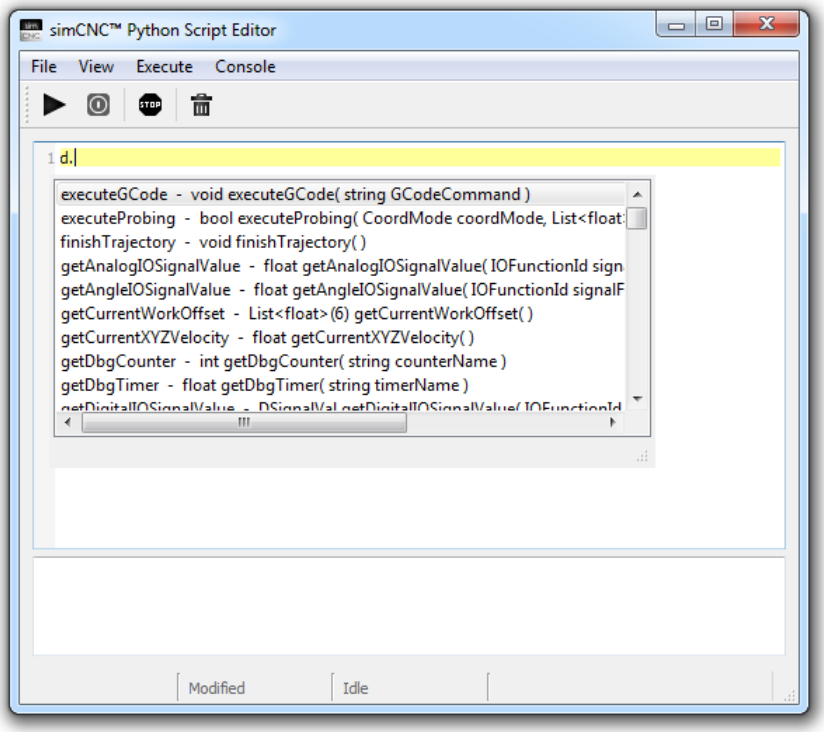
1) Gdy chcemy ujrzeć listę funkcji zawartych w bibliotece „___DEVICE.pyc”.

W tej sytuacji wpisujemy „d.” i naciskamy kombinacje klawiszy „Ctrl” + „Space”. Po wykonaniu tej czynności wyświetli się okno zawierającą listę funkcji znajdującą się w bibliotece „___DEVICE.pyc”

Aby wybrać którąś z funkcji musimy ją podświetlić za pomocą kursorów klawiatury i zatwierdzić wybór wciskając klawiszem „Enter” lub po prostu kliknąć wybraną funkcję za pomocą lewego klawisza myszki.

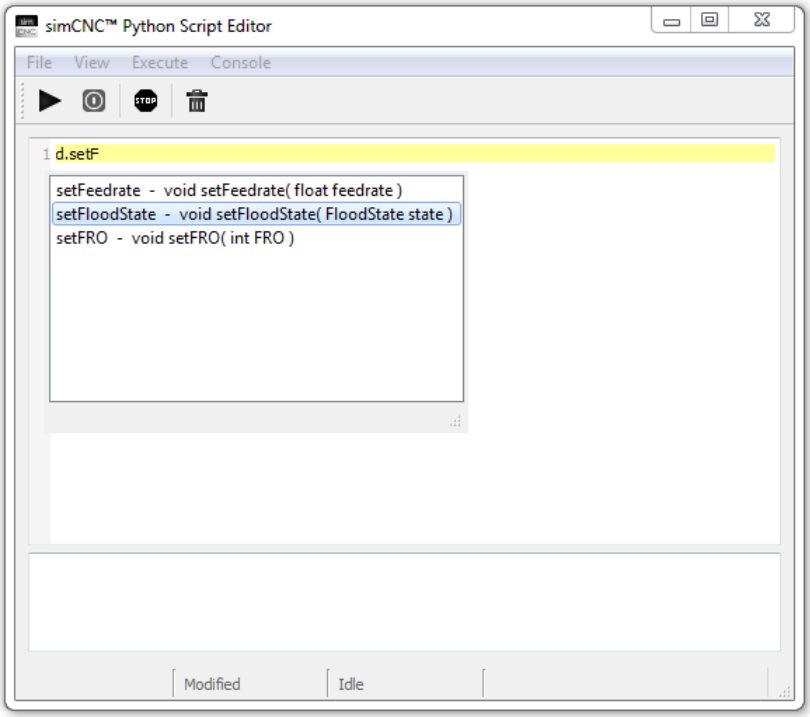
UWAGA!

Wspominana lista nie obejmuje funkcji, które wymagają pobrania modułu. Mowa tu o funkcjach dających bezpośredni dostęp do portów cyfrowych i analogowych.



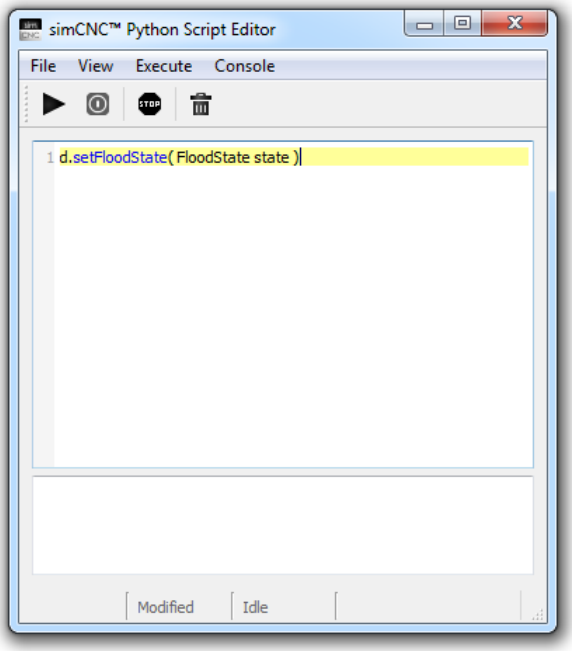
2) Gdy znamy niepełną nazwę funkcji:

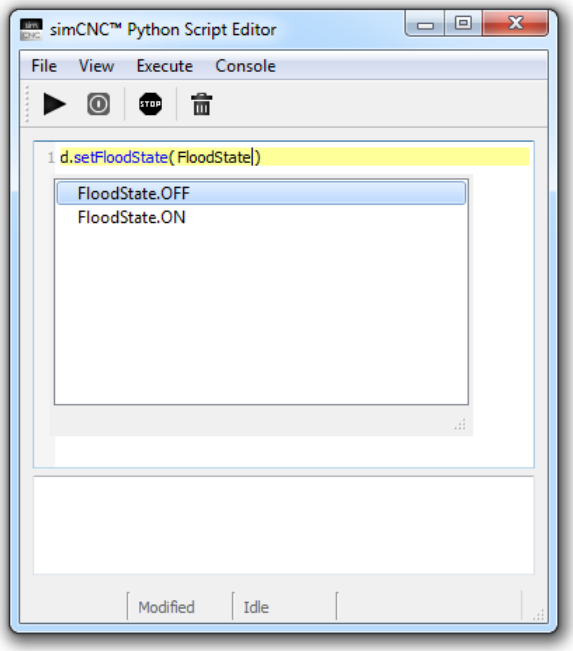
W tej sytuacji także wpisujemy „d.” i naciskamy kombinacje klawiszy „Ctrl” + „Space", lecz tym razem nie przeglądamy listy dostępnych funkcji tylko wpisujemy znany nam początek nazwy funkcji, a okno podpowiedzi wyświetli nam pasujące zakończenia.



3) Gdy odszukaliśmy potrzebną funkcje, ale nie znamy wartości argumentu.

W tej sytuacji wystarczy skasować nazwę argumentu i wcisnąć kombinacje klawiszy „Ctrl” + „Space” (o ile nie została naciśnieta wcześniej), a okno podpowiedzi wyświetli nam wszystkie pasujące do danej funkcji warianty wartości argumentu.





V. Porty cyfrowe i analogowe – bezpośredni dostęp

1) Pobranie modułu.

Aby uzyskać dostęp do portów cyfrowych lub analogowych urządzeń firmy CS-LAB należ pobrać w pierwszej kolejności moduł.

Module d.getModule(ModuleType type, int ID) – Zwraca moduł na urządzenie.

ARGUMENTY FUNKCJI:

ModuleType type – Typ urządzenia

WYRÓŻNIAMY:

- ModuleType.IP – Sterowniki CSMIO/IP-S, CSMIO/IP-A i CSMIO/IP-M
- ModuleType.MPG – Moduł ręcznej manipulacji osiami CSMIO-MPG
- ModuleType.ENC – Moduł gwintowania CSMIO-ENC
- ModuleType.IO – Moduł dodatkowych we/wy CSMIO-IO
- ModuleType.DRV – Napędy simDrive

int ID – Numer urządzenia.

WYRÓŻNIAMY:

- CSMIO-IO (ID od 0 do 15)
- SimDrive (ID od 0 do 5)

W przypadku, gdy używamy kilku modułów dodatkowych we/wy CSMIO-IO lub serwonapędów simDrive, aby było możliwe ich odróżnienie, należy podać właściwy numeru urządzenia. Zasada ta nie dotyczy kontrolera CSMIO/IP, modułu CSMIO-MPG i CSMIO-ENC. W ich przypadku numer urządzenia to zawsze 0, gdyż występują w systemie sterowania zawsze pojedynczo.

WYNIK FUNKCJI:

Module - Moduł na urządzenia

PRZYKŁADY:

```
mod_IP = d.getModule( ModuleType.IP, 0 )              # Pobranie modułu na sterownik CSMIO/IP
mod_MPG = d.getModule( ModuleType.MPG, 0 )            # Pobranie modułu na CSMIO-MPG
mod_ENC = d.getModule( ModuleType.ENC, 0 )            # Pobranie modułu na CSMIO-ENC
mod_IO_0 = d.getModule( ModuleType.IO, 0 )            # Pobranie modułu na CSMIO-IO numer 0
mod_IO_15 = d.getModule( ModuleType.IO, 15 )          # Pobranie modułu na CSMIO-IO numer 15
mod_simDrive_0 = d.getModule( ModuleType.DRV, 0 )     # Pobranie modułu na simDrive numer 0
mod_simDrive_5 = d.getModule( ModuleType.DRV, 5 )     # Pobranie modułu na simDrive numer 5
```

UWAGA!

„mod_IP”, „mod_MPG”, „mod_ENC”, „mod_IO_0”, „mod_IO_15”, „mod_simDrive_0”, i „mod_simDrive_5” to nazwy własne modułów (uchwytów), które zostały wymyślone na potrzeby instrukcji. Nazwy modułów mogą być dowolne, lecz jasno powinny określać, którego urządzenia dotyczą.

2) Porty cyfrowe

a) Odczyt z portów cyfrowych.

DIOPinVal getDigitalIO(IOPortDir direction, int digitalPin) – Zwraca stan pinu, portu cyfrowego wejściowego lub wyjściowego (zwraca stan wejścia lub wyjścia cyfrowego).

ARGUMENTY FUNKCJI:

IOPortDir direction – Określa kierunek portu

WYRÓŻNIAMY:

IOPortDir.InputPort - Port wejściowy
IOPortDir.OutputPort - Port wyjściowy

int digitalPin – Numer pinu (numer wejścia lub wyjścia cyfrowego)

WYNIK FUNKCJI:

DIOPinVal – Stan pinu.

WYRÓŻNIAMY:

DIOPinVal.PinReset – Stan niski pinu
DIOPinVal.PinSet – Stan wysoki pinu

PRZYKŁADY:

Kontroler CSMIO/IP-S, CSMIO/IP-A i CSMIO/IP-M

```
mod_IP = d.getModule( ModuleType.IP, 0 )

# Odczyt stanu wejścia cyfrowego nr 8
if mod_IP.getDigitalIO( IOPortDir.InputPort, 8 ) == DIOPinVal.PinReset :
    print("CSMIO/IP digital input 8 = 0")
if mod_IP.getDigitalIO( IOPortDir.InputPort, 8 ) == DIOPinVal.PinSet :
    print("CSMIO/IP digital input 8 = 1")
```

```
# Odczyt stanu wyjścia cyfrowego nr 4
if mod_IP.getDigitalIO( IOPortDir.OutputPort, 4 ) == DIOPinVal.PinReset :
    print("CSMIO/IP digital output 4 = 0")
if mod_IP.getDigitalIO( IOPortDir.OutputPort, 4 ) == DIOPinVal.PinSet :
    print("CSMIO/IP digital output 4 = 1")
```

CSMIO-MPG - Moduł ręcznej manipulacji osiami

```
mod_MPG = d.getModule( ModuleType.MPG, 0 )

# Odczyt stanu wejścia cyfrowego nr 3
if mod_MPG.getDigitalIO( IOPortDir.InputPort, 3 ) == DIOPinVal.PinReset :
    print("CSMIO-MPG digital input 3 = 0")
if mod_MPG.getDigitalIO( IOPortDir.InputPort, 3 ) == DIOPinVal.PinSet :
    print("CSMIO-MPG digital input 3 = 1")

# Odczyt stanu wyjścia cyfrowego nr 0.
if mod_MPG.getDigitalIO( IOPortDir.OutputPort, 0 ) == DIOPinVal.PinReset :
    print("CSMIO-MPG digital output 0 = 0")
if mod_MPG.getDigitalIO( IOPortDir.OutputPort, 0 ) == DIOPinVal.PinSet :
    print("CSMIO-MPG digital output 0 = 1")
```

CSMIO-IO - Moduł dodatkowych we/wy - numer 0

```
mod_IO_0 = d.getModule( ModuleType.IO, 0 )

# Odczyt stanu wejścia cyfrowego nr 7
if mod_IO _0.getDigitalIO( IOPortDir.InputPort, 7 ) == DIOPinVal.PinReset :
    print("CSMIO-IO 0, digital input 7 = 0")
if mod_IO_0.getDigitalIO( IOPortDir.InputPort, 7 ) == DIOPinVal.PinSet :
    print("CSMIO-IO 0, digital input 7 = 1")

# Odczyt stanu wyjścia cyfrowego nr 2
if mod_IO_0.getDigitalIO( IOPortDir.OutputPort, 2 ) == DIOPinVal.PinReset :
    print("CSMIO-IO 0, digital output 2 = 0")
if mod_IO_0.getDigitalIO( IOPortDir.OutputPort, 2 ) == DIOPinVal.PinSet :
    print("CSMIO-IO 0, digital output 2 = 1")
```

CSMIO-IO - Moduł dodatkowych we/wy - numer 15

```
mod_IO_15 = d.getModule( ModuleType.IO, 15 )

# Odczyt stanu wejścia cyfrowego nr 11
if mod_IO_15.getDigitalIO( IOPortDir.InputPort, 11 ) == DIOPinVal.PinReset :
    print("CSMIO-IO 15, digital input 11 = 0")
if mod_IO_15.getDigitalIO( IOPortDir.InputPort, 11 ) == DIOPinVal.PinSet :
    print("CSMIO-IO 15, digital input 11 = 1")
```

```
# Odczyt stanu wyjścia cyfrowego nr 5
if mod_IO_15.getDigitalIO( IOPortDir.OutputPort, 5) == DIOPinVal.PinReset :
    print("CSMIO-IO 15, digital output 5 = 0")
if mod_IO_15.getDigitalIO( IOPortDir.OutputPort, 5) == DIOPinVal.PinSet :
    print("CSMIO-IO 15, digital output 5 = 1")
```

b) Zapis do portu cyfrowego

void setDigitalIO(int digitalPin, DIOPinVal value) – Ustawia stan pinu, portu cyfrowego (ustawia stan wyjścia cyfrowego).

ARGUMENTY FUNKCJI:

int digitalPin – Numer pinu (numer wyjścia cyfrowego)
DIOPinVal value – Stan pinu

WYRÓŻNIAMY:

DIOPinVal.PinReset – Stan niski pinu
DIOPinVal.PinSet – Stan wysoki pinu

PRZYKŁAD:

Kontroler CSMIO/IP-S, CSMIO/IP-A i CSMIO/IP-M

```
import time
mod_IP = d.getModule( ModuleType.IP, 0 )

# Ustawienie stanu wysokiego na wyjściu cyfrowym nr 1
mod_IP.setDigitalIO( 1, DIOPinVal.PinSet )
time.sleep(1)

# Ustawienie stanu niskiego na wyjściu cyfrowym nr 1
mod_IP.setDigitalIO( 1, DIOPinVal.PinReset )
```

CSMIO-MPG - Moduł ręcznej manipulacji osiami

```
import time
mod_MPG = d.getModule( ModuleType.MPG, 0 )

# Ustawienie stanu wysokiego na wyjściu cyfrowym nr 0
mod_MPG.setDigitalIO( 0, DIOPinVal.PinSet )
time.sleep(1)

# Ustawienie stanu niskiego na wyjściu cyfrowym nr 0
mod_MPG.setDigitalIO( 0, DIOPinVal.PinReset )
```

CSMIO-IO - Moduł dodatkowych we/wy - numer 0

```
import time
mod_IO_0 = d.getModule( ModuleType.IO, 0 )

# Ustawienie stanu wysokiego na wyjściu cyfrowym nr 5
mod_IO_0.setDigitalIO( 5, DIOPinVal.PinSet )
time.sleep(1)

# Ustawienie stanu niskiego na wyjściu cyfrowym nr 5
mod_IO_0.setDigitalIO( 5, DIOPinVal.PinReset )
```

CSMIO-IO - Moduł dodatkowych we/wy - numer 15

```
import time
mod_IO_15 = d.getModule( ModuleType.IO, 15 )

# Ustawienie stanu wysokiego na wyjściu cyfrowym nr 3
mod_IO_15.setDigitalIO( 3, DIOPinVal.PinSet )
time.sleep(1)

# Ustawienie stanu niskiego na wyjściu cyfrowym nr 3
mod_IO_15.setDigitalIO( 3, DIOPinVal.PinReset )
```

3) Porty analogowe

a) Odczyt z portów analogowych

float getAnalogIO(IOPortDir direction, int analogPin) – Zwraca wartość napięcia pinu, portu analogowego wejściowego lub wyjściowego (zwraca wartość napięcie wejścia lub wyjścia analogowego).

ARGUMENTY FUNKCJI:

IOPortDir direction – Określa kierunek portu

WYRÓŻNIAMY:

IOPortDir.InputPort - Port wejściowy
IOPortDir.OutputPort - Port wyjściowy

int analogPin – Numer pinu (numer wejścia lub wyjścia analogowego)

WYNIK FUNKCJI:

float – Wartość napięcia na pinie

PRZYKŁADY:

Kontroler CSMIO/IP-S, CSMIO/IP-A i CSMIO/IP-M

```
mod_IP = d.getModule( ModuleType.IP, 0 )

# Odczyt wartości napięcia wejścia analogowego nr 0
val = mod_IP.getAnalogIO( IOPortDir.InputPort, 0 )
print("CSMIO/IP analog input 0 = " + str(val) + "V")

# Odczyt wartości napięcia wejścia analogowego nr 1
val = mod_IP.getAnalogIO( IOPortDir.InputPort, 1 )
print("CSMIO/IP analog input 1 = " + str(val) + "V")

# Odczyt wartości napięcia wyjścia analogowego nr 0
val = mod_IP.getAnalogIO( IOPortDir.OutputPort, 0 )
print("CSMIO/IP analog output 0 = " + str(val) + "V")

# Odczyt wartości napięcia wyjścia analogowego nr 1
val = mod_IP.getAnalogIO( IOPortDir.OutputPort, 1 )
print("CSMIO/IP analog output 1 = " + str(val) + "V")
```

CSMIO-MPG - Moduł ręcznej manipulacji osiami

```
mod_MPG = d.getModule( ModuleType.MPG, 0 )

# Odczyt wartości napięcia wejścia analogowego nr 0
val = mod_MPG.getAnalogIO( IOPortDir.InputPort, 0 )
print("CSMIO-MPG analog input 0 = " + str(val) + "V")

# Odczyt wartości napięcia wejścia analogowego nr 1
val = mod_MPG.getAnalogIO( IOPortDir.InputPort, 1 )
print("CSMIO-MPG analog input 1 = " + str(val) + "V")
```

b) Zapis do portów analogowych

void setAnalogIO(int analogPin, float value) – Ustawia napięcia na pinie portu analogowego.

ARGUMENTY FUNKCJI:

int analogPin – Numer pinu (numer wyjścia analogowego)
float value – Wartość napięcia na pinie

PRZYKŁADY:

Kontroler CSMIO/IP-S, CSMIO/IP-A i CSMIO/IP-M

```
mod_IP = d.getModule( ModuleType.IP, 0 )

# Ustawienie napięcia o wartości 5V na wyjściu analogowym numer 0
val = 5
mod_IP.setAnalogIO( 0, val )

# Ustawienie napięcia o wartości 2V na wyjściu analogowym numer 1
val = 2
mod_IP.setAnalogIO( 1, val )
```

UWAGA!

Sterowniki CSMIO/IP-S, CSMIO/IP-A, CSMIO/IP-M

Napięcie na pinach, portów analogowych wejściowych i wyjściowych, może przyjmować wartości od 0V do 10V z rozdzielczością 12bit.

Moduł ręcznej manipulacji osiami CSMIO-MPG

Napięcie na pinach, portu analogowego wejściowego, moduł ręcznej manipulacji osiami CSMIO-MPG, może przyjmować wartość od 0V do 5V z rozdzielczością 12bit.

4) Porty enkoderowe - CSMIO-ENC

a) Odczyt kąta enkodera

float getEncoderIOAngle(int channel = 0) - Zwraca kąt enkodera liczoną od sygnału index.

ARGUMENTY FUNKCJI:

int channel – Kanał enkoderowy

WYNIK FUNKCJI:

float – Kąt enkodera

PRZYKŁAD:

```
mod_ENC = d.getModule( ModuleType.ENC, 0)

# Odczyt kąta enkodera numer 0
val = mod_ENC.getEncoderIOAngle( 0 )
print("CSMIO-ENC encoder chanel 0 angle = " + str(val))
```

b) Odczyt pozycji enkodera

int getEncoderIOPostion(int channel = 0) - Zwraca pozycję enkodera liczoną od pierwszego sygnału index.

ARGUMENTY FUNKCJI:

int channel – Kanał enkoderowy

WYNIK FUNKCJI:

int – Pozycja enkodera

PRZYKŁAD:

```
mod_ENC = d.getModule( ModuleType.ENC, 0 )

# Odczyt pozycji enkodera numer 2
val = mod_ENC.getEncoderIOPosition( 2 )
print("CSMIO-ENC encoder chanel 0 postion = " + str(val))
```

c) Odczyt prędkości enkodera

float getEncoderIORPM(int channel = 0) – Zwraca prędkości obrotową enkodera (RPM).

ARGUMENTY FUNKCJI:

int channel – Kanał enkoderowy

WYNIK FUNKCJI:

float – Prędkości obrotowa enkodera

PRZYKŁAD:

```
mod_ENC = d.getModule( ModuleType.ENC, 0)

# Odczyt prędkości obrotowej enkodera numer 4
val = mod_ENC.getEncoderIORPM( 4 )
print("CSMIO-ENC encoder chanel 0 rpm = " + str(val))
```

UWAGA!

Kanały modułu gwintowania o numerach 0, 2 i 4 to fizyczne kanały z kolei kanały o numerach 1, 3 i 5 to wirtualne kopie fizycznych kanałów, które zostaną wykorzystane w przyszłości.

Do odwołania wspominane kanały fizyczne 0, 2 i 4 odpowiadają kanałom o numerach 0, 1 i 2 w konfiguracji simCNC.

VI. Odczyt i zapis koordynat osi

1) Odczyt koordynat

List<float>(6) getPosition(CoordMode coordMode) - Zwraca aktualną wartość koordynat maszynowych lub programowych wszystkich osi.

ARGUMENTY FUNKCJI:

CoordMode coordMode - Określa rodzaj koordynat

WYRÓŻNIAMY:

CoordMode.Machine – Koordynaty maszynowe
CoordMode.Program – Koordynaty programowe

WYNIK FUNKCJI:

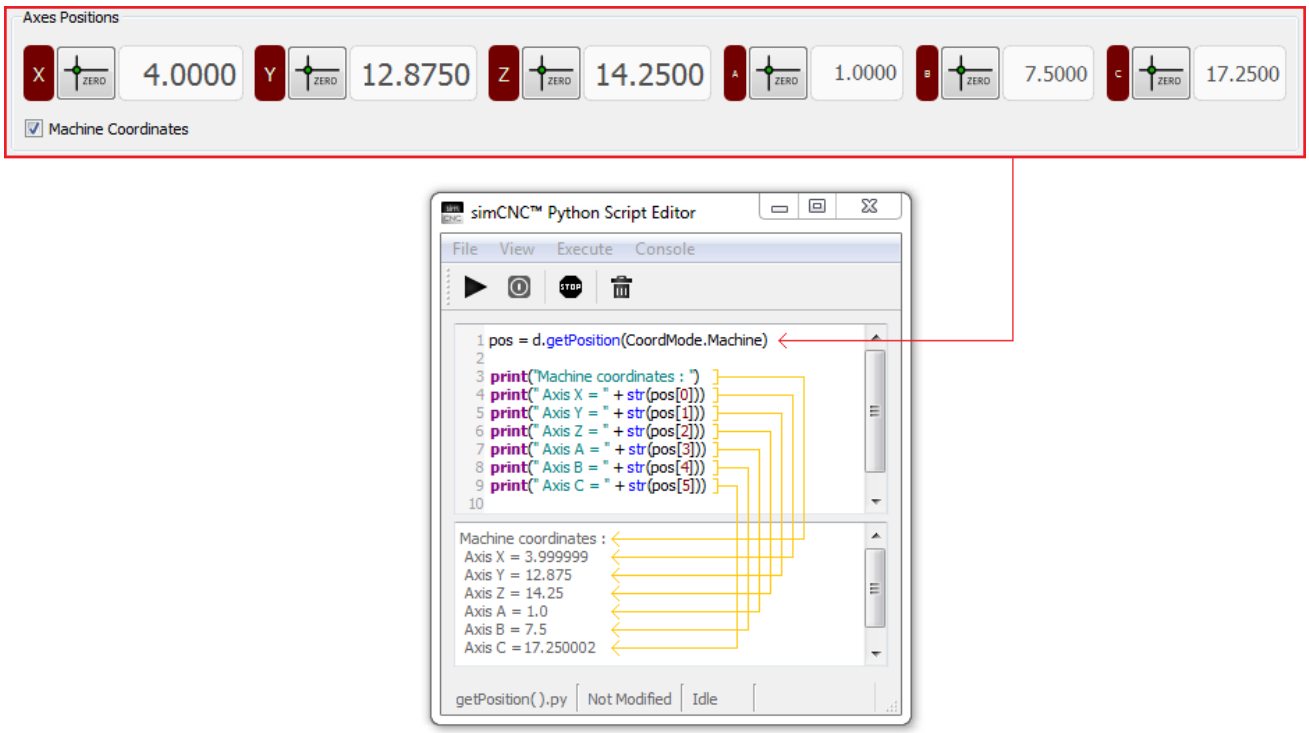
List<float>(6) – lista 6 współrzędnych (X, Y, Z, A, B, C) aktualnej pozycji

PRZYKŁADY:

Odczyt koordynat maszynowych osi X, Y, Z, A, B i C

```
pos = d.getPosition( CoordMode.Machine )

print("Machine coordinates : ")
print(" Axis X = " + str(pos[0]))
print(" Axis Y = " + str(pos[1]))
print(" Axis Z = " + str(pos[2]))
print(" Axis A = " + str(pos[3]))
print(" Axis B = " + str(pos[4]))
print(" Axis C = " + str(pos[5]))
```

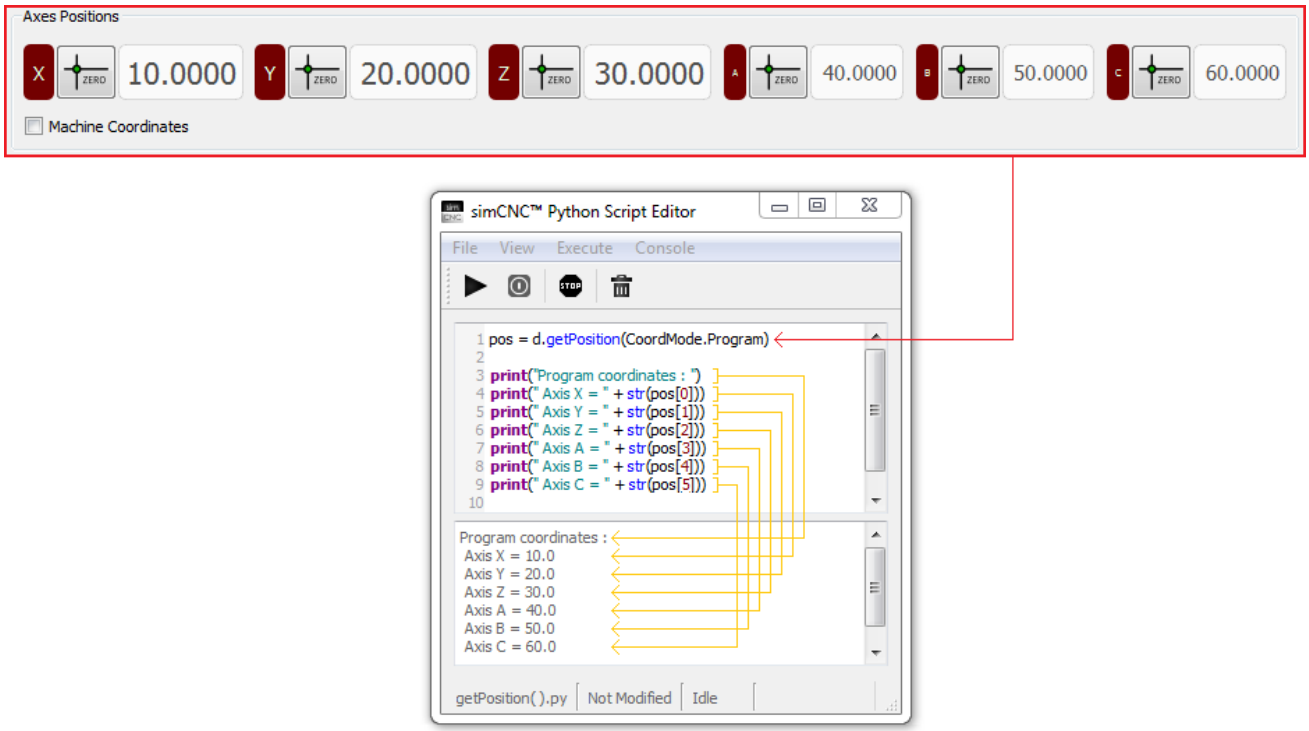
Gdy uruchomisz powyższy przykład, funkcja „d.getPosition(CoordMode.Machine)" odczyta koordynaty maszynowe wszystkich 6 osi i umieści je w 6 elementowej liście indeksowanej od 0 do 5. Następnie za pomocą funkcji „print( )" przechowywane wartości koordynat maszynowych w liście zostaną wyświetlone w konsoli Python.



Odczyt koordynat programowych osi X, Y, Z, A, B i C

```
pos = d.getPosition( CoordMode.Program )

print("Program coordinates : ")
print(" Axis X = " + str(pos[0]))
print(" Axis Y = " + str(pos[1]))
print(" Axis Z = " + str(pos[2]))
print(" Axis A = " + str(pos[3]))
print(" Axis B = " + str(pos[4]))
print(" Axis C = " + str(pos[5]))
```

Gdy uruchomisz powyższy przykład, funkcja „d.getPosition(CoordMode.Machine)” odczyta koordynaty maszynowe wszystkich 6 osi i umieści je w 6 elementowej liście indeksowanej od 0 do 5. Następnie za pomocą funkcji „print( )” przechowywane wartości koordynat maszynowych w liście zostaną wyświetlone w konsoli Python.



2) Zapis koordynat

void setAxisProgPosition(Axis axis, float value) – Ustawia wartość koordynat programowych wybranej osi.

ARGUMENTY FUNKCJI:

Axis axis – Oś do której zostanie zapisana nowa wartość koordynat programowych

WYRÓŻNIAMY:

- Axis.X – Oś X
- Axis.Y – Oś Y
- Axis.Z – Oś Z
- Axis.A – Oś A
- Axis.B – Oś B
- Axis.C – Oś C

float value – Nowa wartość koordynat programowych

PRZYKŁADY:

Zapis koordynat programowych osi X, Y, Z, A, B i C.

```
x = 10
y = 20
z = 30
a = 40
b = 50
c = 60

d.setAxisProgPosition( Axis.X, x )
d.setAxisProgPosition( Axis.Y, y )
d.setAxisProgPosition( Axis.Z, z )
d.setAxisProgPosition( Axis.A, a )
d.setAxisProgPosition( Axis.B, b )
d.setAxisProgPosition( Axis.C, c )
```

To samo tylko z wykorzystaniem listy.

```
pos = (10, 20, 30, 40, 50, 60)

d.setAxisProgPosition(Axis.X, pos[0])
d.setAxisProgPosition(Axis.Y, pos[1])
d.setAxisProgPosition(Axis.Z, pos[2])
d.setAxisProgPosition(Axis.A, pos[3])
d.setAxisProgPosition(Axis.B, pos[4])
d.setAxisProgPosition(Axis.C, pos[5])
```

VII. Poruszanie osiami maszyny

void moveAxisIncremental(Axis axis, float distance, float velocity) - Wykonuje ruch wybraną osią o zadany dystans (ruch inkrementalny) z zadaną prędkością.

ARGUMENTY:

Axis axis - Oś która wykona ruch

WYRÓŻNIAMY:

- Axis.X – Oś X
- Axis.Y – Oś Y
- Axis.Z – Oś Z
- Axis.A – Oś A
- Axis.B – Oś B
- Axis.C – Oś C

float distance – Dystans ruchu
float velocity – Prędkość ruchu

PRZYKŁADY:

```
d.moveAxisIncremental( Axis.X, 20, 1000 )
d.moveAxisIncremental( Axis.Y, 20, 1000 )
d.moveAxisIncremental( Axis.X, -20, 1000 )
d.moveAxisIncremental( Axis.Y, -20, 1000 )
```

Gdy uruchomisz powyższy przykład oś X i Y wykonają ruch z bieżącej pozycji, po obwodzie kwadratu, o boku 20mm lub cali z prędkością 1000mm/min lub cali/min.

void moveToPosition(CoordMode coordMode, List<float>[6] position, float velocity) - Wykonuje ruch do zadanej pozycji, wyrażonej w koordynatach maszynowych lub programowych z zadaną prędkością wypadkową.

ARGUMENTY:

CoordMode coordMode - Określa rodzaj koordynat

WYRÓŻNIAMY:

- CoordMode.Machine – Koordynaty maszynowe
- CoordMode.Program – Koordynaty programowe

List<float>[6] position - Lista 6 współrzędnych (X, Y, Z, A, B, C) pozycji docelowej
float velocity - Zadana prędkość wypadkowa

PRZYKŁADY:

```
X = 0
Y = 1
pos = d.getPosition(CoordMode.Program)

pos[X] = pos[X] + 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
pos[Y] = pos[Y] + 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
pos[X] = pos[X] - 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
pos[Y] = pos[Y] - 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
```

Gdy uruchomisz powyższy przykład osie X i Y wykona za pomocą koordynat programowych, ruch z bieżącej pozycji, po obwodzie kwadratu, o boku 20mm lub cali z prędkością 1000mm/min lub cali/min.

```
X = 0
Y = 1
pos = d.getPosition(CoordMode.Machine)

pos[X] = pos[X] + 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
pos[Y] = pos[Y] + 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
pos[X] = pos[X] - 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
pos[Y] = pos[Y] - 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
```

Gdy uruchomisz powyższy przykład osie X i Y wykona za pomocą koordynat maszynowych, ruch z bieżącej pozycji, po obwodzie kwadratu, o boku 20mm lub cali z prędkością 1000mm/min lub cali/min.

Oba powyższe przykłady pokazują, w jaki sposób można użyć list do modyfikacji koordynat.
Ten sposób obliczeń będziesz mógł bardzo często zobaczyć w skryptach, które będą udostępniane na stronie CS-LAB, dlatego warto się temu przyjrzeć już teraz.

void parallelMoveToPosition(CoordMode coordMode, Axis axis, float position, float velocity) - Wykonuje ruch wybraną osią niezależną, do zadanej pozycji wyrażonej w koordynatach maszynowych lub programowych z zadaną prędkością. Funkcja działa gdy nie zaznaczono opcji „Use external points" („Sterowanie z planera ruchu „)

ARGUMENTY:

CoordMode coordMode - Określa rodzaj koordynat

WYRÓŻNIAMY:

CoordMode.Machine – Koordynaty maszynowe
CoordMode.Program – Koordynaty programowe

Axis axis – Niezależna oś, która wykona ruch

WYRÓŻNIAMY:

Axis.X – Oś X
Axis.Y – Oś Y
Axis.Z – Oś Z
Axis.A – Oś A
Axis.B – Oś B
Axis.C – Oś C

float position - Docelowa pozycja osi
float velocity - Zadana prędkość

PRZYKŁADY:

```
X = 0
pos = d.getPosition(CoordMode.Program)

d.parallelMoveToPosition( CoordMode.Program, Axis.X, 25, 100 )
d.parallelMoveToPosition( CoordMode.Program, Axis.X, pos[X], 20 )
```

Gdy uruchomisz powyższy przykład oś X wykona ruch za pomocą koordynat programowych do pozycji 25 mm lub cali z prędkością 100mm/min lub cali/min, a następnie powróci do punktu startu z prędkością 20mm/min lub cali/min.

```
Z = 2
pos = d.getPosition(CoordMode.Program)

d.parallelMoveToPosition( CoordMode. Machine, Axis.Z, 78, 350 )
d.parallelMoveToPosition( CoordMode. Machine, Axis.Z, pos[Z], 500 )
```

Gdy uruchomisz powyższy przykład oś Z wykona ruch za pomocą koordynat maszynowych do pozycji 78 mm lub cali z prędkością 350mm/min lub cali/min, a następnie powróci do punktu startu z prędkością 500mm/min lub cali/min.

float getCurrentXYZVelocity( ) - Zwraca aktualną prędkość wypadkową osi X, Y i Z.

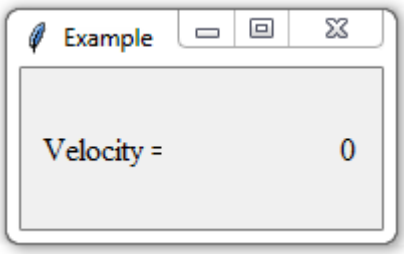
WYNIK FUNKCJI:

float - Prędkość wypadkowa

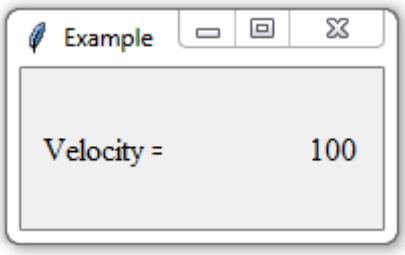
PRZYKŁADY:

```
from tkinter import *

def show():
    velocity = d.getCurrentXYZVelocity( )
    L2["text"] = str(int(velocity))
    window.after(5, show)

window = Tk()
window.geometry("180x80")
window.title( "Example" )

L1 = Label(window, text="Velocity =", font= ("Times New Roman",12))
L1.place(x=10, y=30, height=20, width=65)

L2 = Label(window, font= ("Times New Roman",12), anchor = "e")
L2.place(x=70, y=30 ,height=20, width=100)

window.after(5, show)
mainloop()
```

Gdy uruchomisz powyższy przykład skrypt wyświetli okno przedstawiające aktualną prędkością wypadkową osi X, Y i Z. Obok zdjęcie przedstawiające sytuację, w której żadna z osi nie porusza się.



Aby sprawdzić czy skrypt działa zaznacz opcji „Kontrola z klawiatury” lub użyj skrótu „Alt + j” i porusz osią lub osiami za pomocą kursorów klawiatury.



VIII. Probing

bool executeProbing(CoordMode coordMode, List<float>[6] position, int probeIndex, float velocity) - Wykonuje sondowania wybraną sondą do zadanej pozycji wyrażonej w koordynatach maszynowych lub programowych z zadaną prędkością wypadkową.

ARGUMENTY:

CoordMode coordMode - Określa rodzaj koordynat

WYRÓŻNIAMY:

CoordMode.Machine - Koordynaty maszynowe
CoordMode.Program - Koordynaty programowe

List<float>[6] position - Lista 6 współrzędnych (X, Y, Z, A, B, C) pozycji docelowej
int probeIndex - Numer sondy (konfigóracje sond można znaleźć w Settings/Modules/Special IO)
float velocity - Prędkość wypadkowa

WYNIK FUNKCJI:

bool – Rezultat sondowania

WYRÓŻNIAMY:

True - W przypadku gdy sonda zadziałała
False - W przypadku gdy oś lub osie osiągnęły pozycję docelową a sonda nie zadziała

PRZYKŁADY znajdziesz w opisie następnej funkcji

List<float>[6] getProbingPosition(CoordMode coordMode) - Zwraca pozycję zadziałania sondy lub pozycję docelową wyrażoną w koordynatach maszynowych lub programowych. Funkcja zwraca pozycję docelową tylko w przypadku gdy sonda podczas procesu sondowania nie zadziała.

ARGUMENTY:

CoordMode coordMode - Określa rodzaj koordynat

WYRÓŻNIAMY:

CoordMode.Machine - Koordynaty maszynowe
CoordMode.Program - Koordynaty programowe

WYNIK FUNKCJI:

List<float>[6] - lista 6 współrzędnych (X, Y, Z, A, B, C) pozycji zdziałania sondy lub pozycji docelowej.

PRZYKŁADY do obu powyższych:

```
Probing_vel = 10
Return_vel = 100
Distance = 20

Starting_position = d.getPosition( CoordMode.Program )
Maximum_position = Starting_position.copy()
Maximum_position[2] -= Distance

if ( d.executeProbing( CoordMode.Program, Maximum_position, 0, Probing_vel ) == False ):
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    msg.err( "Probing Failed !!!" , "Inactive probe" )
else:
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    Probe_position = d.getProbingPosition( CoordMode.Program )
    msg.info( "Position = " + str(Probe_position[2]),"Probing - Z axis" )
```

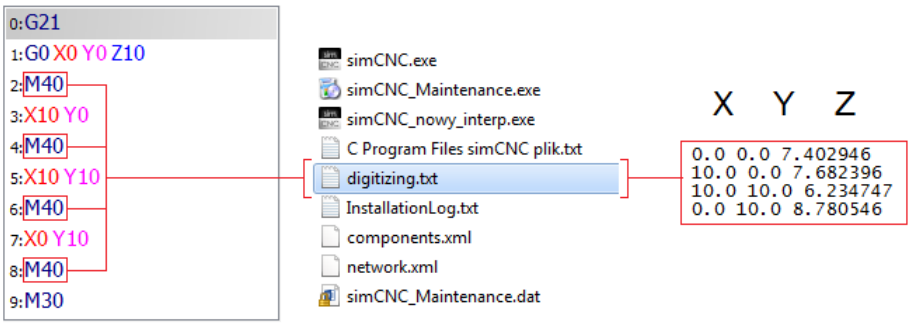
Powyższy skrypt pokazuje jak wykonać sondowanie za pomocą osi Z, z bieżącej pozycji na dystansie określnym przez parametr „Distance”. Po zakończeniu sondowania niezależnie od jego rezultatu oś Z zawsze powraca na pozycję startową. Parametr „Probing_vel” odpowiada za prędkość sondowania, a parametr „Return_vel” odpowiada za prędkość powrotu sondy do pozycji startowej. Wynik sondowania jest prezentowany za pomocą wyskakującego okna.

Prosty sposób na Digitalizację!

Prezentowany wcześniej skrypt po drobnej modyfikacji może pełnić funkcję makra, które zapisuje siatkę punktów sondowanej powierzchni. Proces zapisywania siatki punktów jest najczęściej używany podczas kopiowania płaskorzeźba, czy też frezowanie lub grawerowanie na nierównej powierzchni np. na powierzchni kamienia.

```
import sys

File_path = "digitizing.txt"                #C:\Program Files\simCNC\digitizing.txt

try:
    File = open(File_path, "a+")
except IOError:
    sys.exit("\n No access to file !!!")

Probing_vel = 100
Return_vel = 1000
Distance = 20
Starting_position = d.getPosition( CoordMode.Program )
Maximum_position = Starting_position.copy()
Maximum_position[2] -= Distance

if ( d.executeProbing( CoordMode.Program, Maximum_position, 0, Probing_vel ) == False ):
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    Saved_text ="Probing Failed !!! \n"
else:
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    Probe_position = d.getProbingPosition( CoordMode.Program )
    Saved_text = str(Probe_position[0]) + " " + str(Probe_position[1]) + " " + str(Probe_position[2]) + "\n"

try:
    File.write(Saved_text)
except IOError:
    File.close()
    sys.exit("\n File write error !!!")
```

Wystarczy teraz zapisać powyższy skrypt np. pod nazwą M40 i z poziomu gcod wywoływać je co pewien dystans. Gdy skrypt zostanie wywołany, wykona sondowanie, a następnie zapiszę pozycje zadziałania sondy wraz z pozycją osi X i Y do pliku C:\Program Files\simCNC\digitizing.txt. Poniżej można zobaczyć efekt działania makra na przykładzie prostego gcodu tworzącego ruch po obwodzie kwadrat o wymiarach 10 x 10.



W przypadku gdyby sonda nie zadział na dystansie określnym przez parametr „Distance” w pliku „digitizing.txt” zostanie umieszczony wpis „Probing Failed !!!”.

```
0.0 0.0 6.716997
10.0 0.0 7.318946
Probing Failed !!!
0.0 10.0 8.143046
```

VIII. Magazyn narzędzi

1) Numer narzędzia.

int getSelectedToolNumber( ) - Zwraca numer wybranego narzędzia z poziomu Gcodu, linii MDI lub skryptu Python.

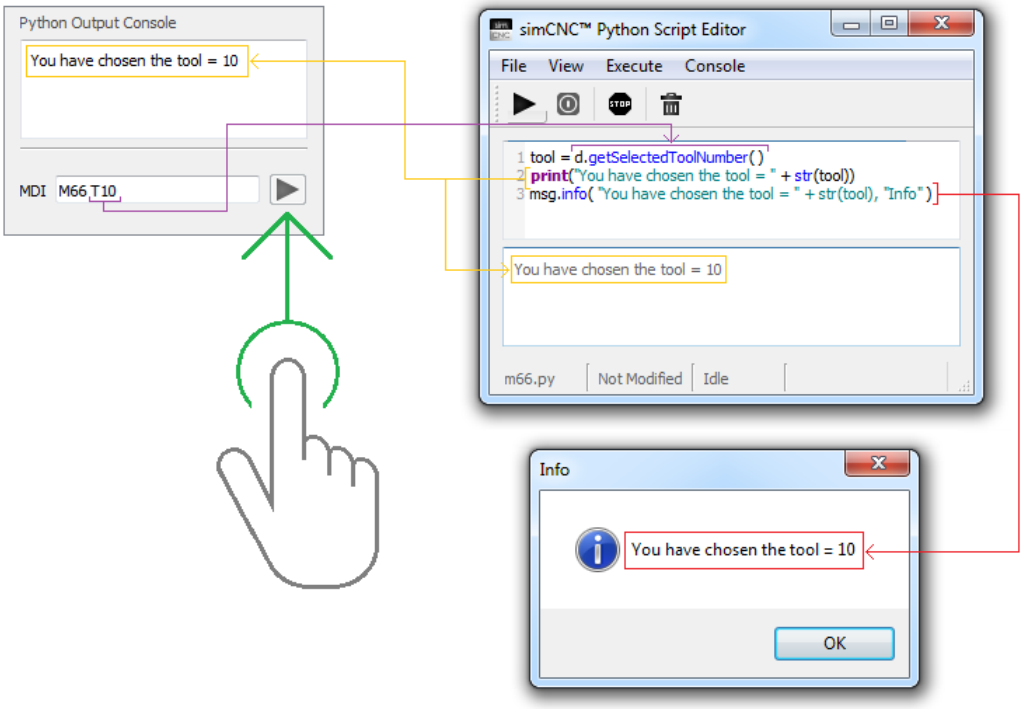
WYNIK FUNKCJI:

int - Numer narzędzia

PRZYKŁAD:

```
tool = d.getSelectedToolNumber( )
print("You have chosen the tool = " + str(tool))
msg.info( "You have chosen the tool = " + str(tool), "Info" )
```

Powyższy skrypt zapisz jako np. M66, a następnie wywołaj z poziomu Gcodu lub linii MDI dodając komendę „T” + dowolny numer narzędzia (np. M66 T10). Po tej czynności powinno się pojawić okno informujące nas jaki numerem narzędzia został wybrany. Dla przykładu użyliśmy linii MDI i narzędzia numer 10. Poleciliśmy użyć makra M66 celowo, aby pokazać że funkcja d.getSelectedToolNumber( ) działa z dowolnym makrem, a nie tylko z M6.



void setSelectedToolNumber(int toolNumber) - Ustawia numer wybranego narzędzia - odpowiednik komendy „Txx” wywołanej z poziomu Gcodu lub linii MDI.

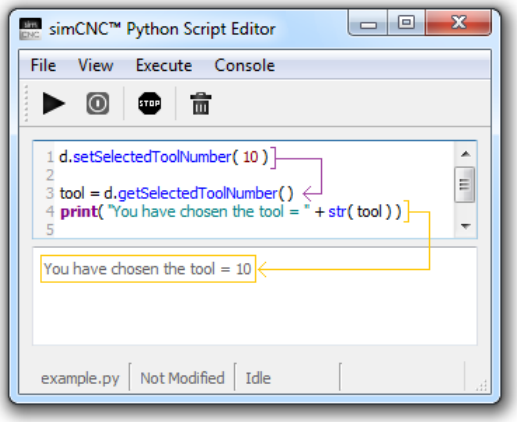
ARGUMENTY:

int toolNumber - Numer narzędzia

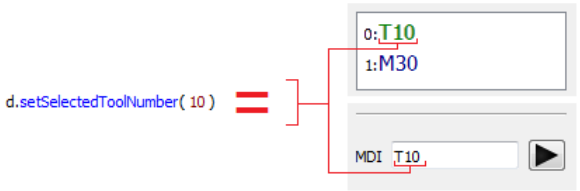
PRZYKŁAD:

```
d.setSelectedToolNumber( 10 )                    # Ustawia numer wybranego narzędzia

tool = d.getSelectedToolNumber( )                # Zwraca numer wybranego narzędzia
print( "You have chosen the tool = " + str( tool ) )
```

Gdy uruchomieniu powyższego przykład funkcja „d.setSelectedToolNumber( )” ustawi numer wybranego narzędzia na 10, a już znana nam funkcja „d.getSelectedToolNumber( )” zwraci ten sam numer narzędzia.



Jak już teraz łatwo zauważyć funkcja d.setSelectedToolNumber(10) wykonuje dokładnie to samo zadanie co komenda T10 wywołana z poziomu Gcodu lub linii MDI.



void setSpindleToolNumber(int toolNumber) - Ustawia numer narzędzia we wrzecionie.

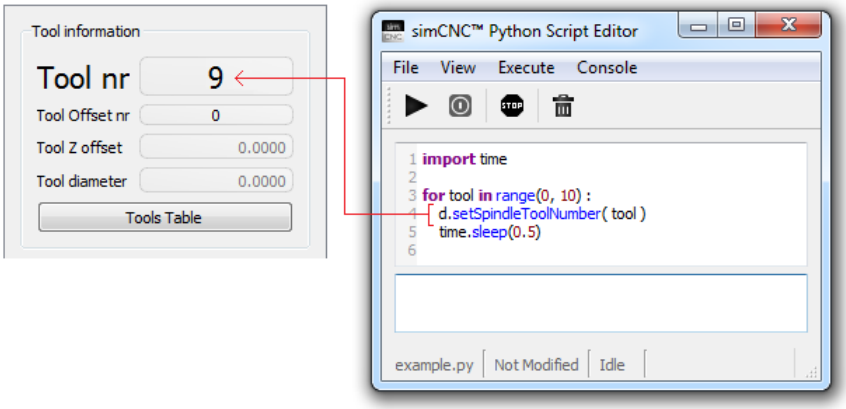
ARGUMENTY:

int toolNumber - Numer narzędzia

PRZYKŁAD:

```
import time

for tool in range(0, 10) :
  d.setSpindleToolNumber( tool )
  time.sleep(0.5)
```

Gdy uruchomisz powyższy przykład, funkcja „d.setSpindleToolNumber(tool)" umieszczona w pętli „for", co 0.5 sekundy będzie zwiększała numer narzędzia na ekranie simCNC aż osiągnie wartość 9.



int getSpindleToolNumber( ) - Zwraca numer narzędzia umieszczonego we wrzecionie.

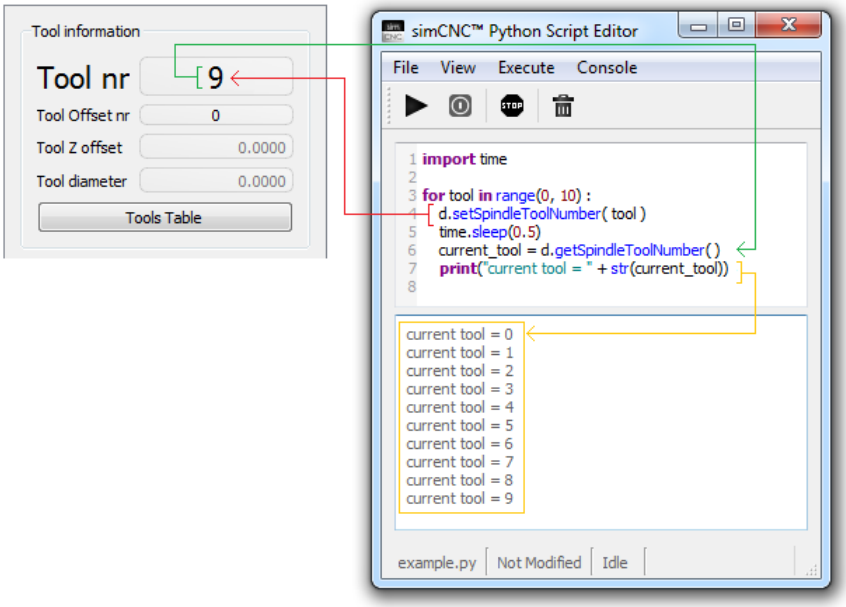
WYNIK FUNKCJI:

int - Numer narzędzia

PRZYKŁAD:

```
import time

for tool in range(0, 10) :
  d.setSpindleToolNumber( tool )
  time.sleep(0.5)
  current_tool = d.getSpindleToolNumber( )
  print("current tool = " + str(current_tool))
```

Powyższy skrypt jest kontynuacją poprzedniego przykładu. Do pętli „for” tym razem dodano funkcję „current_tool = d.getSpindleToolNumber( )”, która za każdym obiegiem pętli „for” zwraca aktualny numer narzędzia umieszczonego w wrzecionie, następnie numer narzędzia za pomocą funkcji „print” jest prezentowany w konsoli Python.



2) Długość narzędzia

void setToolLength(int toolNumber, float toolLength) - Ustawia wartość offsetu długości narzędzia.

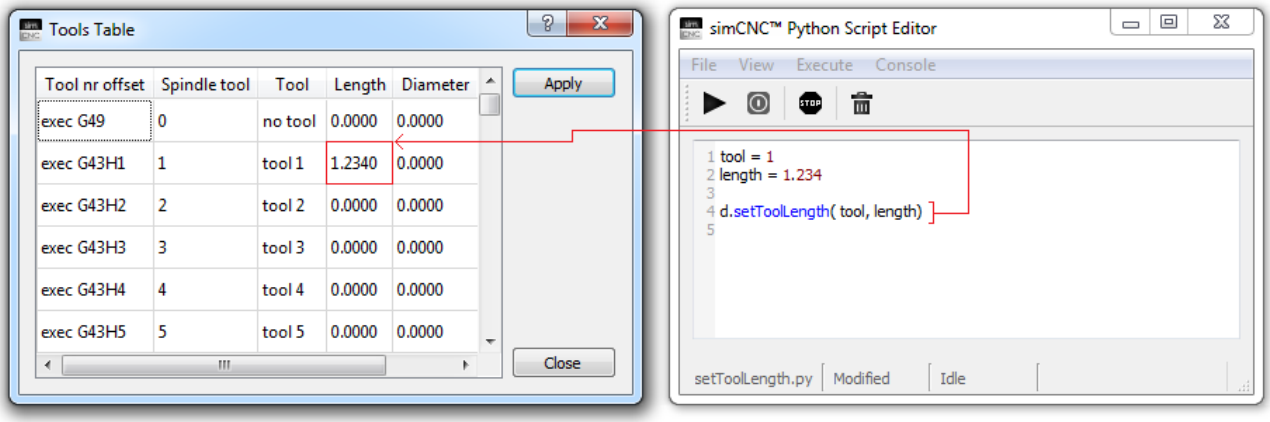
ARGUMENTY:

int toolNumber - Numer narzędzia
float toolLength - Wartość offsetu długości narzędzia

PRZYKŁAD:

```
tool = 1
length = 1.234

d.setToolLength( tool, length )
```

Gdy uruchomisz powyższy przykład, offset długości narzędzia numer 1, zostanie ustawiony na wartość 1.234.



float getToolLength(int toolNumber) - Zwraca wartość offsetu długości narzędzia.

ARGUMENTY:

int toolNumber - Numer narzędzia

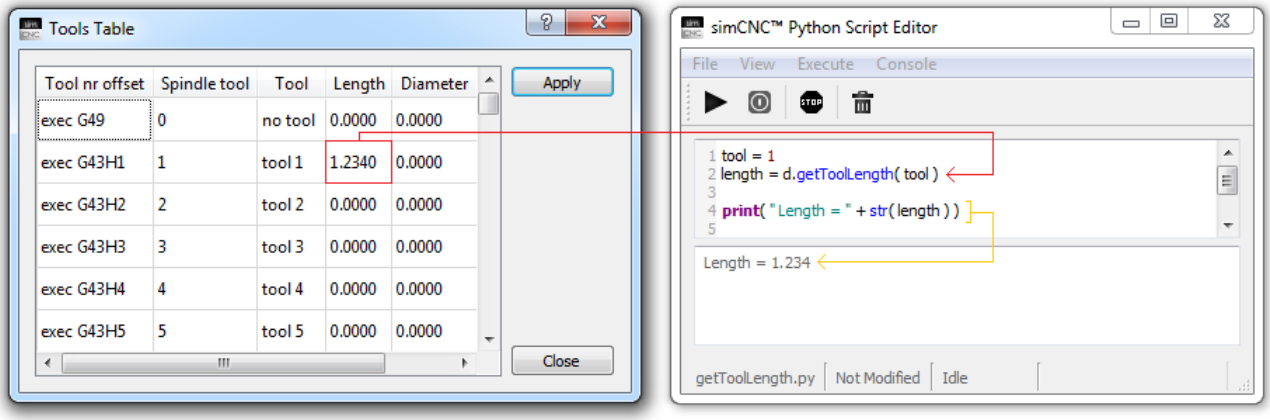
WYNIK FUNKCJI:

float - Wartość offsetu długości narzędzia

PRZYKŁAD:

```
tool = 1
length = d.getToolLength( tool )

print( " Length = " + str( length ) )
```

Gdy uruchomisz powyższy przykład w konsoli Python, zostanie wyświetlona wartość offsetu długości, narzędzia numer 1.



void setToolOffsetNumber(int toolNumber) - Ustawia numer offset długości narzędzia.

ARGUMENTY:

int toolNumber - Numer narzędzia

PRZYKŁAD:

```
tool = 1

d.setToolOffsetNumber( tool )
```

Gdy uruchomisz powyższy przykład numer offsetu długości narzędzia zostanie ustawiony na 1.

int getToolOffsetNumber( ) - Zwraca aktualnie używany numer offsetu długości narzędzia.

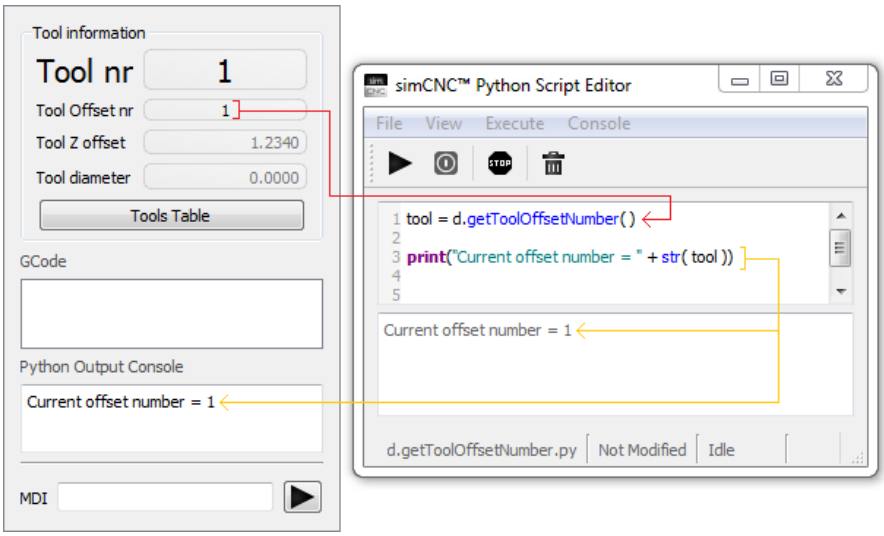
ARGUMENTY:

int - Numer narzędzia

PRZYKŁAD:

```
tool = d.getToolOffsetNumber( )

print("Current offset number = " + str( tool ))
```

Gdy uruchomisz powyższy przykład w konsoli Python zostanie wyświetlony numer aktualnie używanego offsetu długości narzędzia.



3) Średnica narzędzia

UWAGA!

SimCNC obecnie nie wykorzystuje wartości średnicy narzędzia, oznacza to że simCNC obecnie nie obsługuje funkcji kompensacji średnicy narzędzia.

void setToolDiameter(int toolNumber, float toolDiameter) - Ustawia wartość średnicy narzędzia.

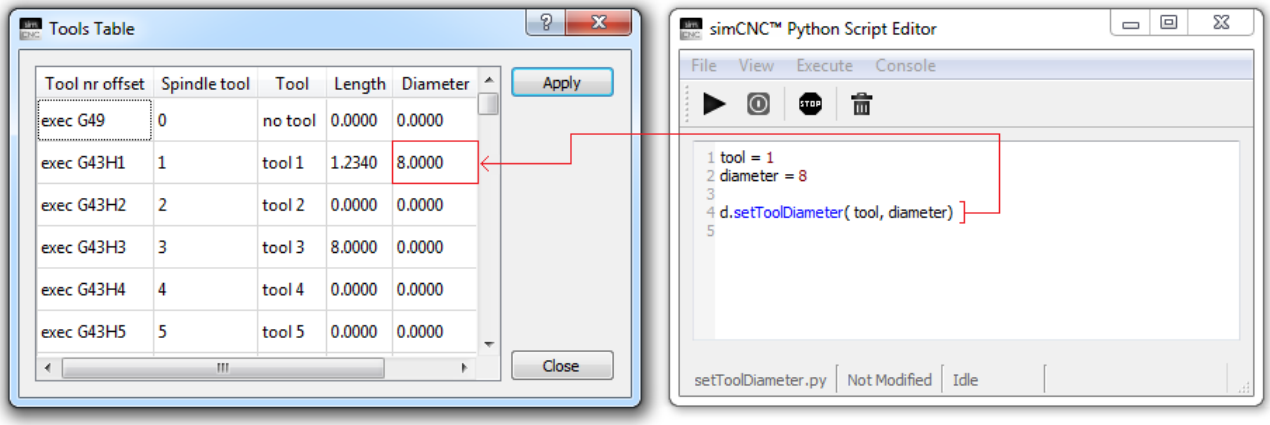
ARGUMENTY:

int toolNumber - Numer narzędzia
float toolDiameter - Średnica narzędzia

PRZYKŁAD:

```
tool = 1
diameter = 8

d.setToolDiameter( tool, diameter )
```

Gdy uruchomisz powyższy przykład, średnicy narzędzia numer 1, zostanie ustawiony na wartość 8.



float getToolDiameter(int toolNumber) - Zwraca wartość średnicy narzędzia narzędzi.

ARGUMENTY:

int toolNumber - Numer narzędzia

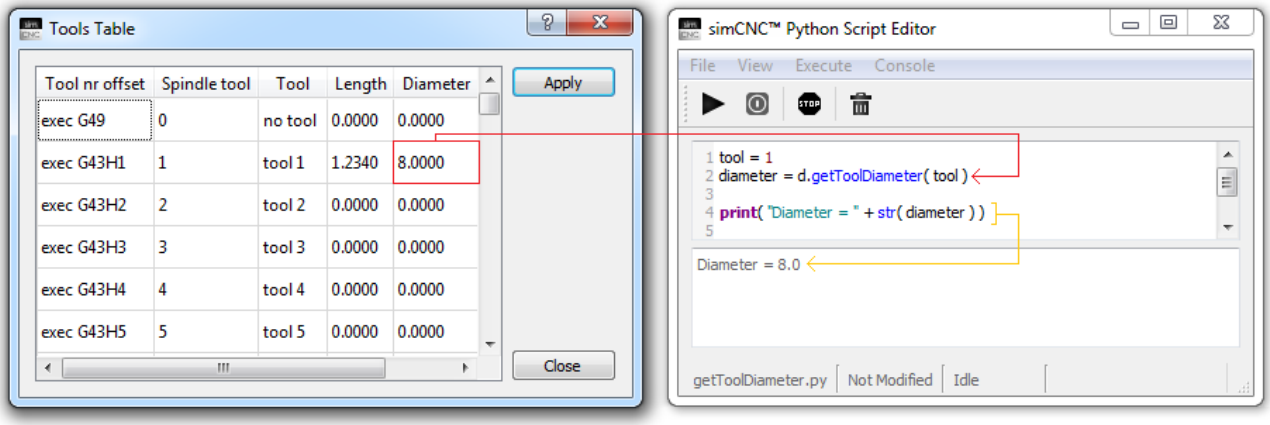
WYNIK FUNKCJI:

float - Średnica narzędzia

PRZYKŁAD:

```
tool = 1
diameter = d.getToolDiameter( tool )

print( "Diameter = " + str( diameter ) )
```

Gdy uruchomisz powyższy przykład w konsoli Python zostanie wyświetlona wartość średnicy narzędzia numer 1.



IX. Wrzeciono, chłodziwo i mgła.

1) Wrzeciono

void setSpindleSpeed(float spindleSpeed) - Ustawia zadaną prędkość obrotową wrzeciona (RPM).

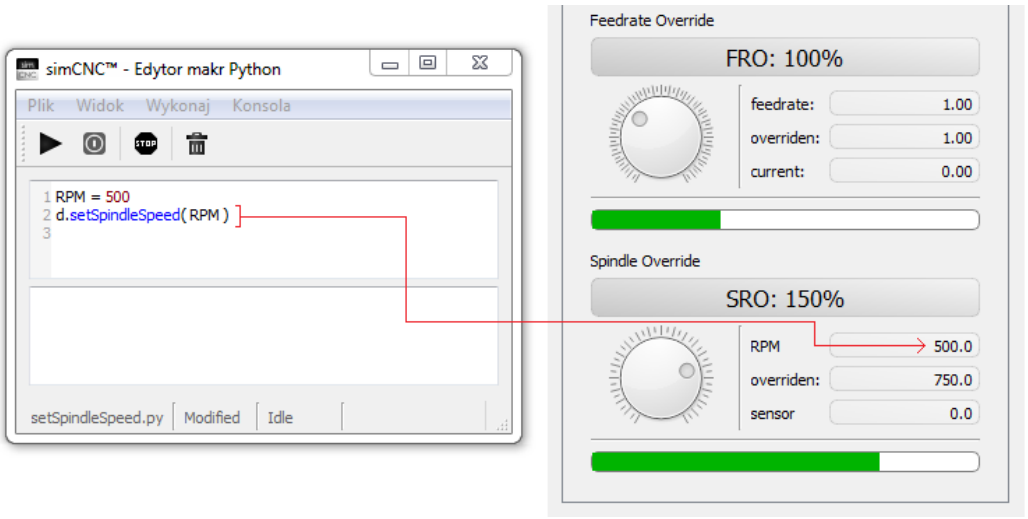
ARGUMENTY:

float spindleSpeed – zadana prędkość obrotowa wrzeciona (RPM)

PRZYKŁAD:

```
RPM = 500
d.setSpindleSpeed( RPM )
```

Gdy uruchomisz powyższy przykład zadana prędkość obrotowa wrzeciona zostanie ustawiona na 500 RPM.



float getSpindleSpeed( ) - Zwraca zadaną prędkość obrotową wrzeciona (RPM).

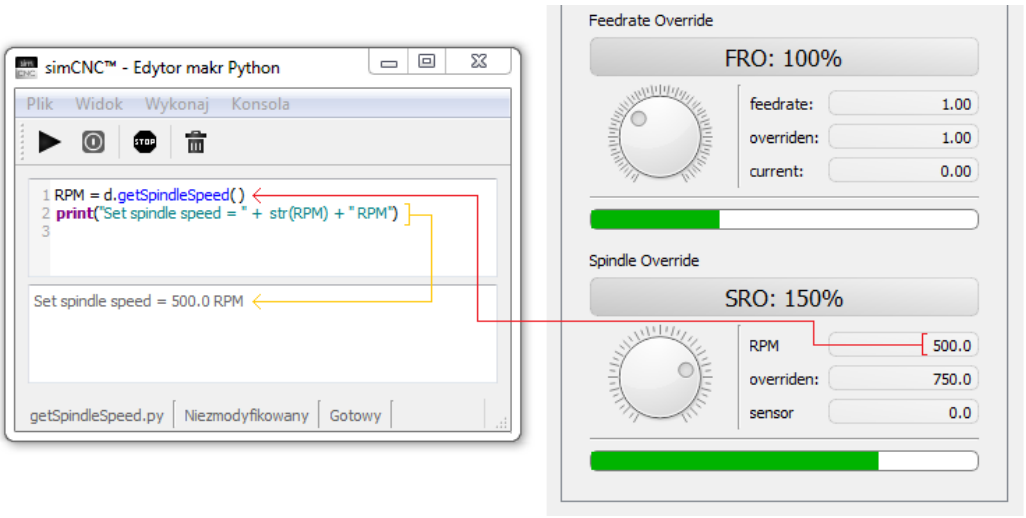
WYNIK FUNKCJI:

float – zadana prędkość obrotowa wrzeciona (RPM)

PRZYKŁAD:

```
RPM = d.getSpindleSpeed( )
print("Set spindle speed = " + str(RPM))
```

Gdy uruchomisz powyższy przykład w konsoli Python zostanie wyświetlona zadana prędkość obrotowa wrzeciona.



void setSpindleState(SpindleState spindleState) - Ustawia stan wrzeciona

ARGUMENTY:

SpindleState spindleState - stan wrzeciona

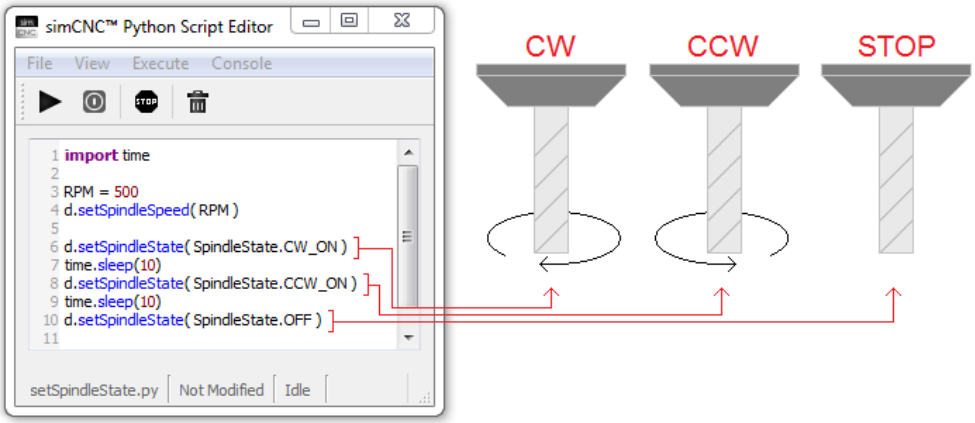
WYRÓŻNIAMY:

SpindleState.CW_ON	- Obroty wrzeciona zgodne z ruchem wskazówek zegara
SpindleState.CCW_ON	- Obroty wrzeciona przeciwne do ruchu wskazówek zegara
SpindleState.OFF	- Obroty wrzeciona wyłączone

PRZYKŁAD:

```
import time

RPM = 500
d.setSpindleSpeed( RPM )

d.setSpindleState( SpindleState.CW_ON )
time.sleep(10)
d.setSpindleState( SpindleState.CCW_ON )
time.sleep(10)
d.setSpindleState( SpindleState.OFF )
```

Gdy uruchomisz powyższy przykład, wrzeciono załączy się na 10 sekund w kierunku zgodnym z ruchem wskazówek zegara, następnie zmieni kierunek na przeciwny do ruchu wskazówek zegara i po 10 sekundach się wyłączy.
Należy pamiętać że funkcja setSpindleState wstrzymuje wykonywanie się skryptu na czas trwania rampy rozpędzającej i hamującej wrzeciona.



SpindleState getSpindleState( ) - Zwraca aktualny stan wrzeciona.

WYNIK FUNKCJI:

SpindleState - stan wrzeciona

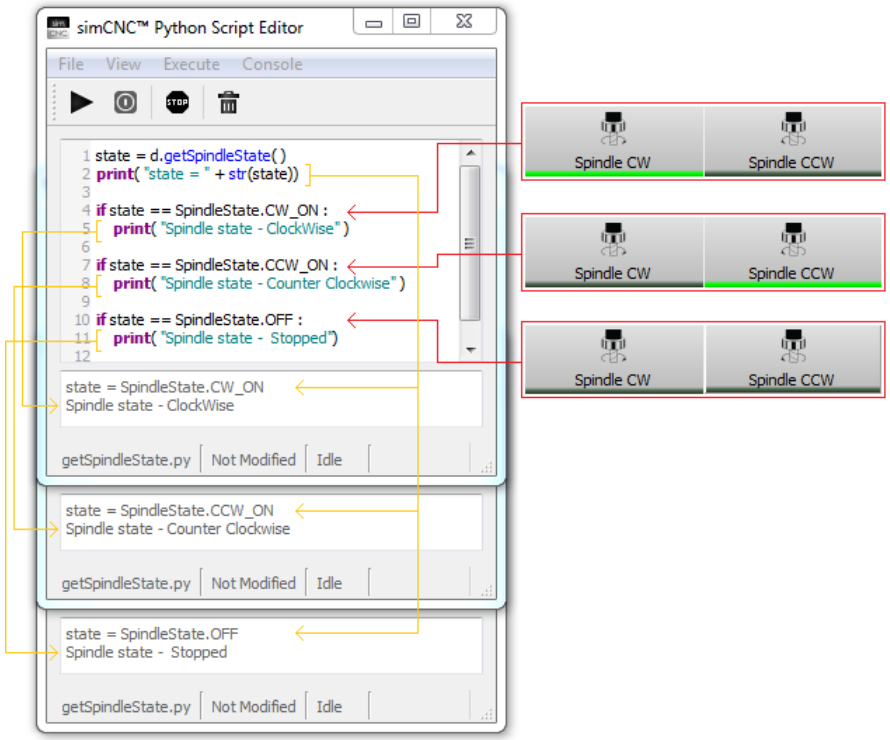
WYRÓŻNIAMY:

SpindleState.CW_ON	- Obroty wrzeciona zgodne z ruchem wskazówek zegara
SpindleState.CCW_ON	- Obroty wrzeciona przeciwne do ruchu wskazówek zegara
SpindleState.OFF	- Obroty wrzeciona wyłączone

PRZYKŁAD:

```
state = d.getSpindleState( )
print( "state = " + str(state))

if state == SpindleState.CW_ON :
  print( "Spindle state - ClockWise" )

if state == SpindleState.CCW_ON :
  print( "Spindle state - Counter Clockwise" )

if state == SpindleState.OFF :
  print( "Spindle state - Stopped")
```

Gdy uruchomisz powyższy przykład w pierwszej linii konsoli Python zostanie wyświetlony aktualny stan wrzeciona, będący wynikiem funkcji d.getSpindleState( ). W drugiej linii zostanie wyświetlony komunikat własny, opisujący jaśniej bieżący stan wrzeciona.
Na poniższym szkicu przedstawiono działanie skryptu w przypadku wszystkich stanów wrzeciona.



void waitForSpindleSetSpeed(int timeout_sec) - Oczekuje na przekroczenie przez wrzeciono określonej prędkości obrotowej, przez określony czas. Oczekiwana prędkość obrotowa wrzeciona jest określana przez zadaną prędkość obrotową i parametr „ Spindle Ready Level" („Próg gotowości").

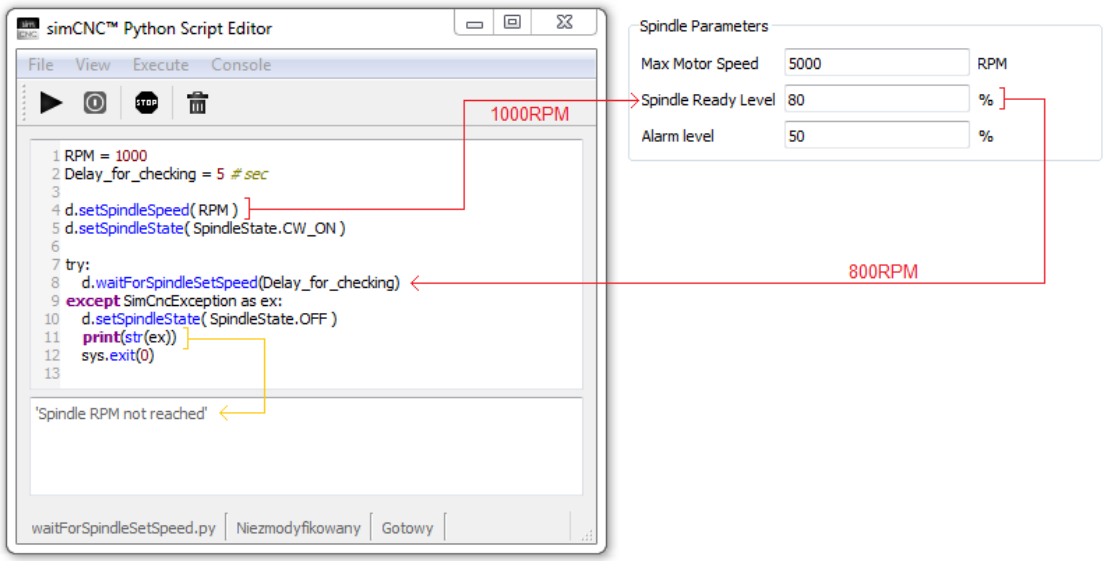
ARGUMENTY:

int timeout_sec - czas oczekiwania (w sekundach)

PRZYKŁAD:

```
RPM = 500
Delay_for_checking = 5 # sec

d.setSpindleSpeed( RPM )
d.setSpindleState( SpindleState.CW_ON )

try:
  d.waitForSpindleSetSpeed(Delay_for_checking)
except SimCncException as ex:
  print(str(ex))
  sys.exit(0)
```

Gdy uruchomisz powyższy przykład, wrzeciono zostanie załączone w kierunku zgodnym z ruchem wskazówek zegara (linia 5 – patrz na zdjęcie poniżej) z zadaną prędkością obrotową równą 1000 RPM (linia 4). Następnie funkcja „waitForSpindleSetSpeed" wstrzyma wykonywanie skryptu do momentu osiągnięcia przez wrzeciono 80% zadanej prędkości obrotowej wrzeciona czyli 800RPM (80% z 1000 RPM = 800RPM). W przypadku, gdy wrzeciono nie osiągnie 800RPM w ciągu 5 sekund, funkcja „waitForSpindleSetSpeed" zgłosi wyjątek typu „SimCncException" (linia 9). Spowoduje to zatrzymanie wrzeciona (linia 10), wyświetlenie w konsoli Python treści wyjątku (linia 11) i zakończenie skryptu na żądanie (linia 12) w celu nie dopuszczenia do dalszego jego wykonywania się. Do działania powyższego przykładu jest wymagane posiadanie modułu CSMIO-ENC gdyż stanowi on źródłem informacji o aktualnej prędkości obrotowej wrzeciona.



Poprzedni przykład po drobnej modyfikacji może wykorzystać jako makro M3, które będzie wstrzymywało wykonywanie się gcodu na czas rozpędzenia wrzeciona do oczekiwanej prędkości obrotowej lub zatrzymywało pracę maszyny w przypadku problemów z osiągnieciem tej prędkości obrotowej.

```
Delay_for_checking = 5 # sec

d.setSpindleState( SpindleState.CW_ON )

try:
  d.waitForSpindleSetSpeed( Delay_for_checking )
except SimCncException as ex:
  d.setSpindleState( SpindleState.OFF )
  print(str(ex))
  d.stopTrajectory( )
  sys.exit(0)
```

2) Chłodziwo

void setFloodState(FloodState state) - Ustawia stan chłodziwa płynnego.

ARGUMENTY:

FloodState state - Stan chłodziwa płynnego

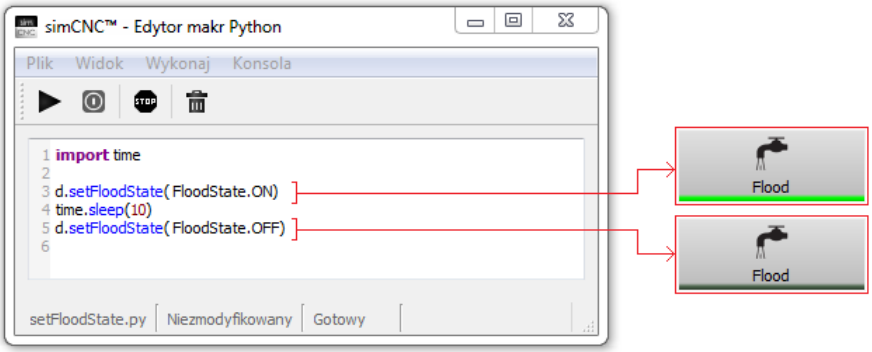
WYRÓŻNIAMY:

FloodState.ON	- Włącz chłodziwo
FloodState.OFF	- Wyłącz chłodziwo

PRZYKŁAD:

```
import time

d.setFloodState( FloodState.ON )
time.sleep(10)
d.setFloodState( FloodState.OFF )
```

Gdy uruchomisz powyższy przykład chłodziwo zostanie załączone na 10 sekund.



FloodState getFloodState( ) - Zwraca stan chłodziwa płynnego.

WYNIK FUNKCJI:

FloodState - Stan chłodziwa płynnego

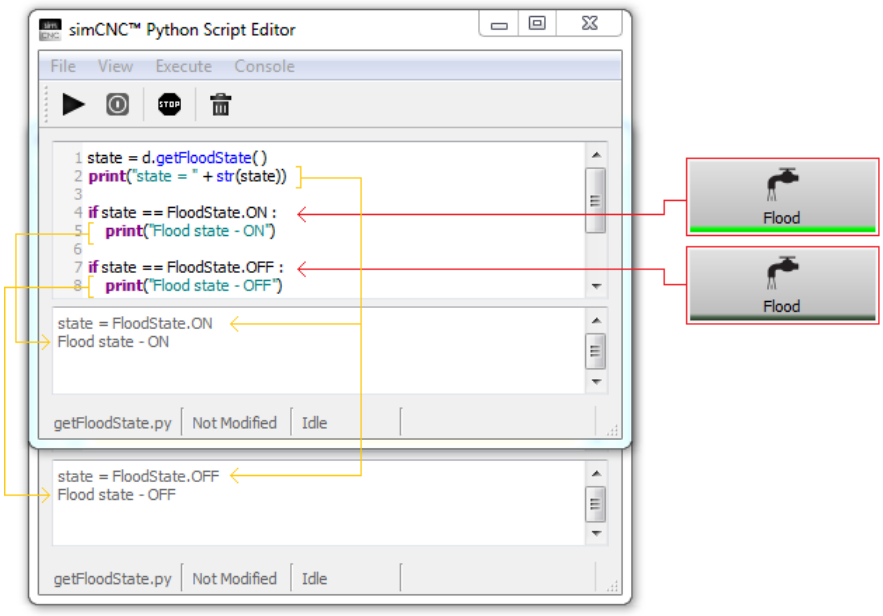
WYRÓŻNIAMY:

FloodState.ON - Włączone chłodziwo
FloodState.OFF - Wyłączone chłodziwo

PRZYKŁAD:

```
state = d.getFloodState( )
print("state = " + str(state))

if state == FloodState.ON :
    print("Flood state - ON")

if state == FloodState.OFF :
    print("Flood state - OFF")
```

Gdy uruchomisz powyższy przykład w pierwszej linii konsoli Python zostanie wyświetlony aktualny stan chłodziwa, będący wynikiem funkcji d.getFloodState( ). W drugiej linii zostanie wyświetlony komunikat własnym, opisujący jaśniej bieżący stan chłodziwa.
Na poniższym szkicu przedstawiono działanie skryptu w przypadku obu stanów chłodziwa.



3) Mgła

void setMistState(MistState state) - Ustawia stan mgły chłodzącej.

ARGUMENTY:

MistState state - Stan mgły chłodzącej

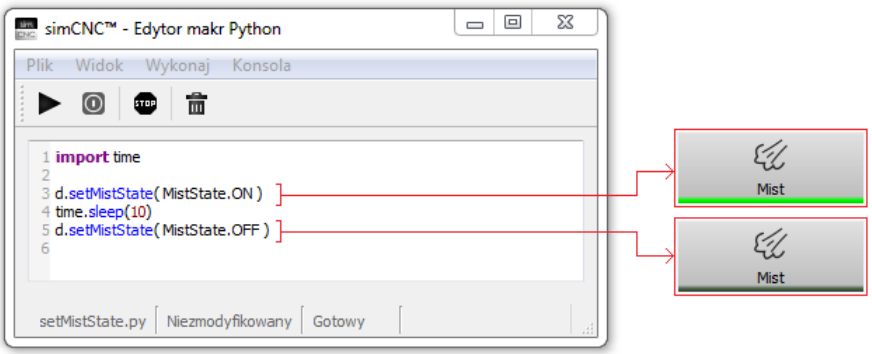
WYRÓŻNIAMY:

MistState.ON - Włącz mgłę
MistState.OFF - Wyłącz mgłę

PRZYKŁAD:

```
import time

d. ( MistState.ON )
time.sleep(10)
d.setMistState( MistState.OFF )
```

Gdy uruchomisz powyższy przykład chłodziwo zostanie załączone na 10 sekund.



MistState getMistState( ) - Zwraca stan mgły chłodzącej.

WYNIK FUNKCJI:

MistState – Stan mgły chłodzącej

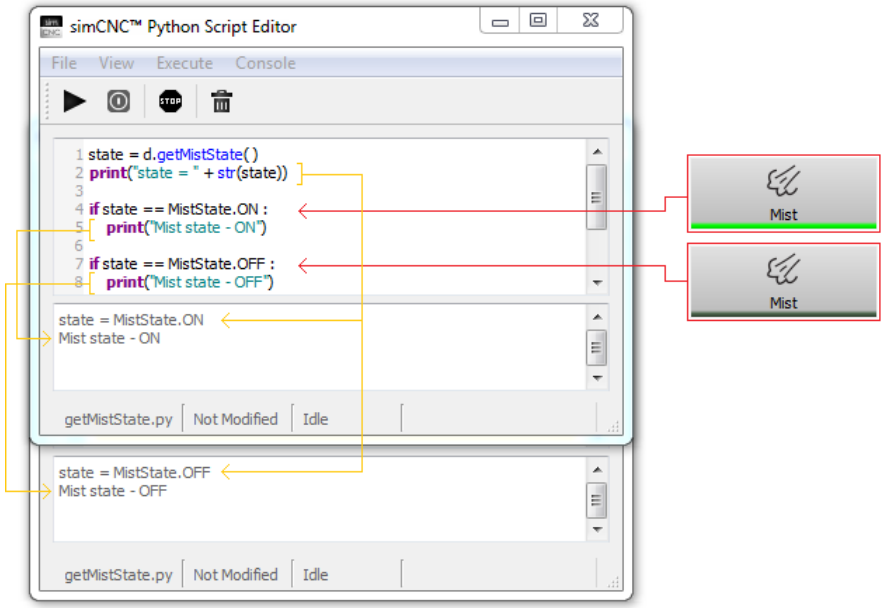
WYRÓŻNIAMY:

MistState.ON - Mgła włączona
MistState.OFF - Mgła wyłączona

PRZYKŁAD:

```
state = d.getMistState( )
print("state = " + str(state))

if state == MistState.ON :
    print("Mist state - ON")

if state == MistState.OFF :
    print("Mist state - OFF")
```

Gdy uruchomisz powyższy przykład w pierwszej linii konsoli Python zostanie wyświetlony aktualny stan mgły, będący wynikiem funkcji getMistState( ). W drugiej linii zostanie wyświetlony komunikat własnym, opisujący jaśniej bieżący stan mgły.
Na poniższym szkicu przedstawiono działanie skryptu w przypadku obu stanów mgły.



X. Offset roboczy - bazy materiałowe.

List<float>[6] getCurrentWorkOffset( ) – Zwraca wartości współrzędnych aktualnego offsetu roboczego.

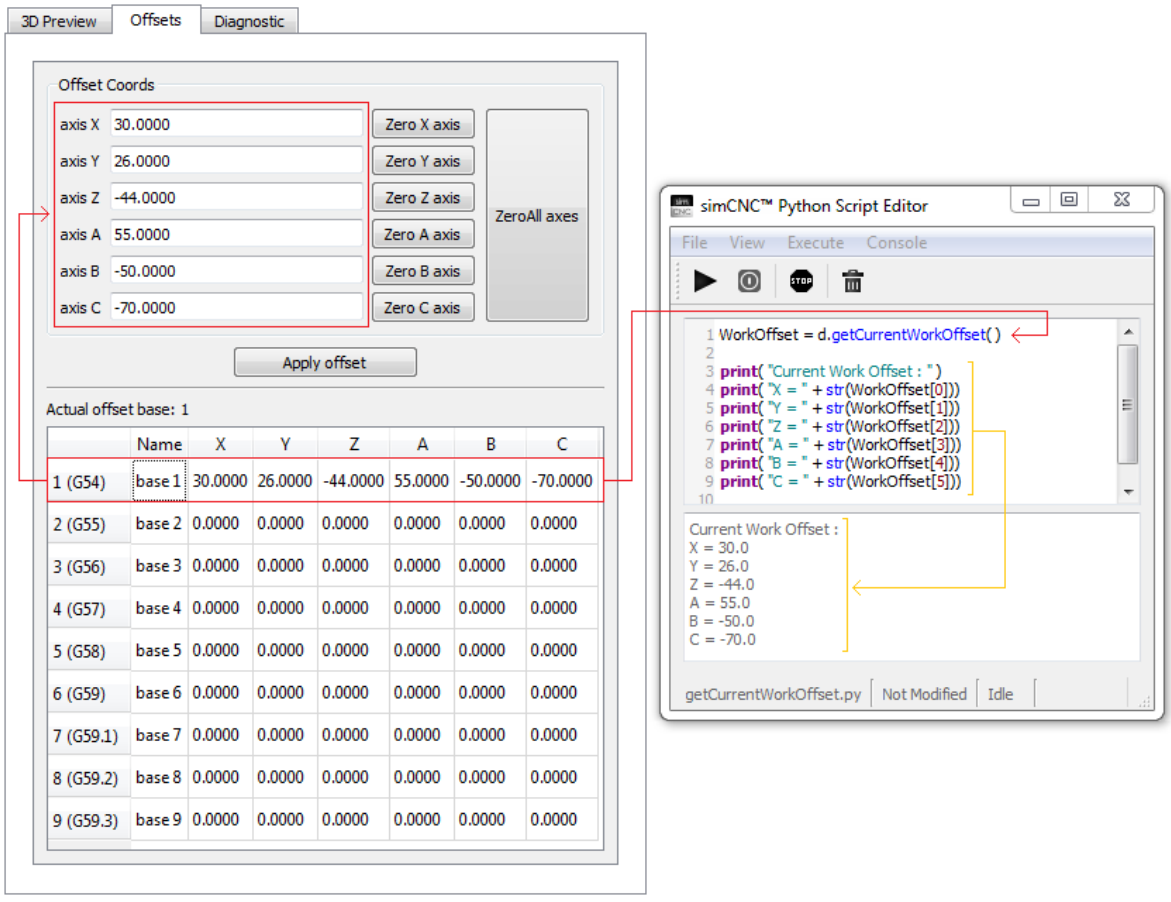
WYNIK FUNKCJI:

List<float>[6] - lista 6 współrzędnych (X, Y, Z, A, B, C) określających offset roboczy

PRZYKŁAD:

```
CurrentWorkOffset = d.getCurrentWorkOffset( )

print( "Current Work Offset : " )
print( "X = " + str(CurrentWorkOffset[0]))
print( "Y = " + str(CurrentWorkOffset[1]))
print( "Z = " + str(CurrentWorkOffset[2]))
print( "A = " + str(CurrentWorkOffset[3]))
print( "B = " + str(CurrentWorkOffset[4]))
print( "C = " + str(CurrentWorkOffset[5]))
```

Gdy uruchomisz powyższy przykład, zostanie odczytana i wyświetlony w konsoli Python, wartość współrzędnych aktualnego offsetu roboczego.



List<float>[6] getWorkOffset(int workOffsetIndex) - Zwraca wartości współrzędnych offsetu roboczego o zadanym indeksie.

ARGUMENTY:

int workOffsetIndex - index (numer) offsetu roboczego

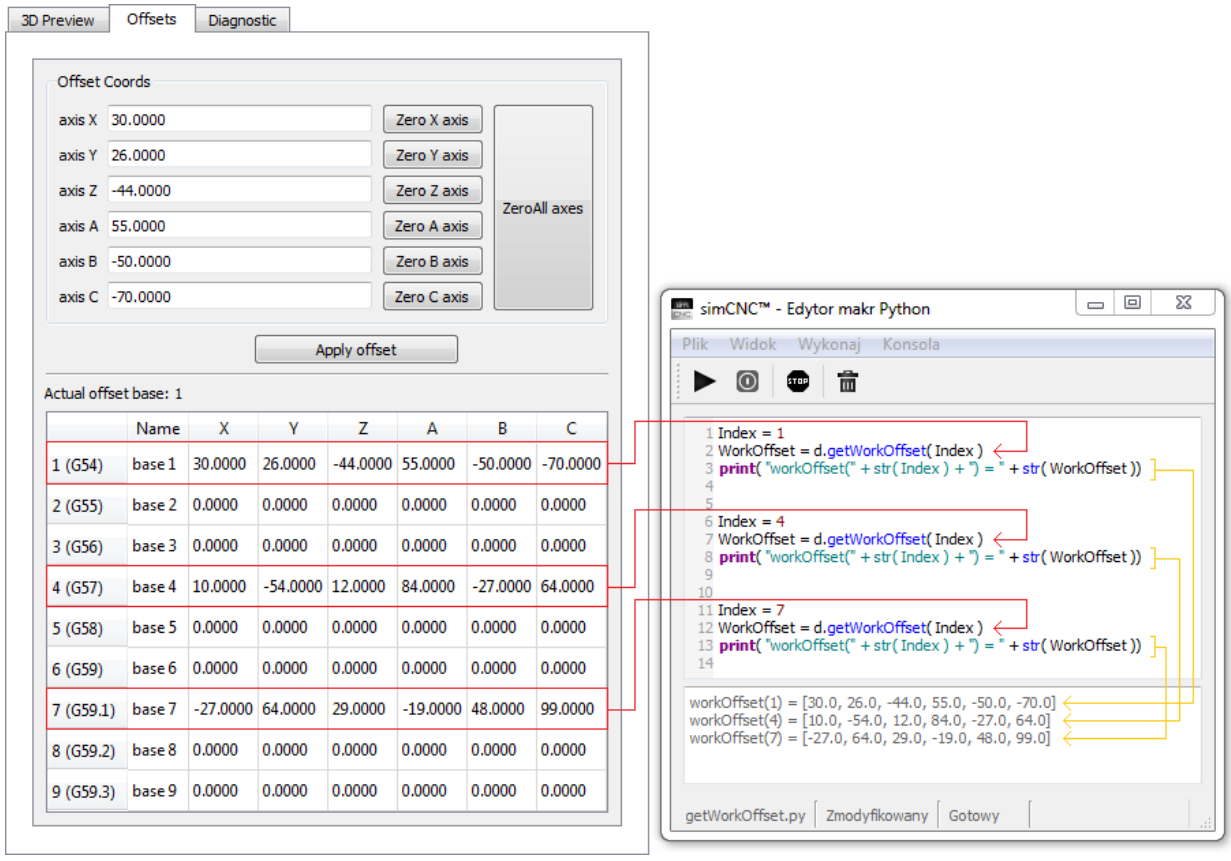
WYNIK FUNKCJI:

List<float>[6] - lista 6 współrzędnych (X, Y, Z, A, B, C) offsetu roboczego

PRZYKŁAD:

```
Index = 1
WorkOffset = d.getWorkOffset( Index )
print( "workOffset(" + str( Index ) + ") = " + str( WorkOffset ))

Index = 4
WorkOffset = d.getWorkOffset( Index )
print( "workOffset(" + str( Index ) + ") = " + str( WorkOffset ))

Index = 7
WorkOffset = d.getWorkOffset( Index )
print( "workOffset(" + str( Index ) + ") = " + str( WorkOffset ))
```

Gdy uruchomisz powyższy przykład, zostanie odczytana i wyświetlony w konsoli Python, wartość współrzędnych offsetu roboczego numer 1 (G54), 4 (G57) i 7 (G59.1).



UWAGA!

Obecnie simCNC posiada możliwość zapisanie 9 offsetów roboczych (9 baz materiałowych).
Co za tym idzie offsety robocze mogą przyjmować index od 1 (G54) do 9 (G59.3).
Dla jasności : 1 = G54 | 2 = G55 | 3 = G56 | 4 = G57 | 5 = G58 | 6 = G59 | 7 = G59.1 | 8 = G59.2 | 9 = G59.3

int getWorkOffsetNumber( ) - Zwraca index aktualnego offsetu roboczego.

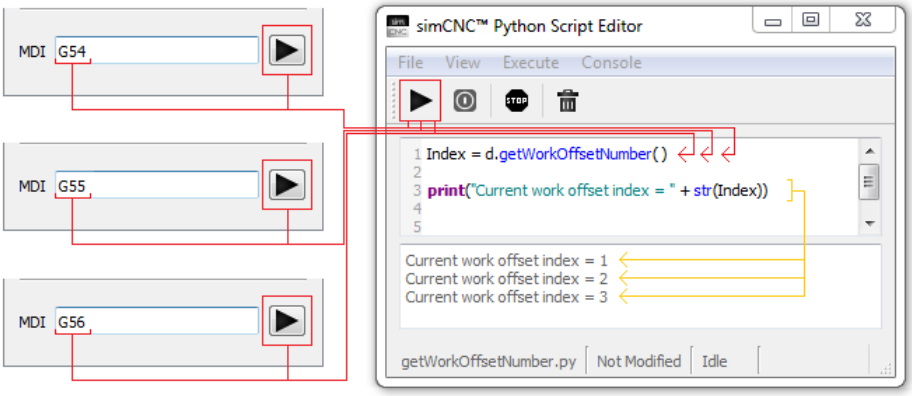
WYNIK FUNKCJI:

int - index (numer) offsetu roboczego

PRZYKŁAD:

```
Index = d.getWorkOffsetNumber( )

print("Current work offset index = " + str(Index))
```

Przed uruchomieniem powyższego przykład wpisz w linie MDI komendę „G54” i wykonaj ją po przez naciśniecie przycisku znajdującym się po prawej stronie linii MDI. Następnie uruchom przykładowy skrypt, gdy to zrobisz w konsoli Python pojawi się informacja o indeksie aktualnie używanego offsetu roboczego. Powtórz te same czynności z komendą „G55” i „G56” a uzyskasz taki sam efekt jak na poniższym zdjęciu.



setWorkOffset(int workOffsetIndex, List<float>[6] workOffsetValues) - Zmienia wartości współrzędnych offsetu roboczego o zadanym indeksie.

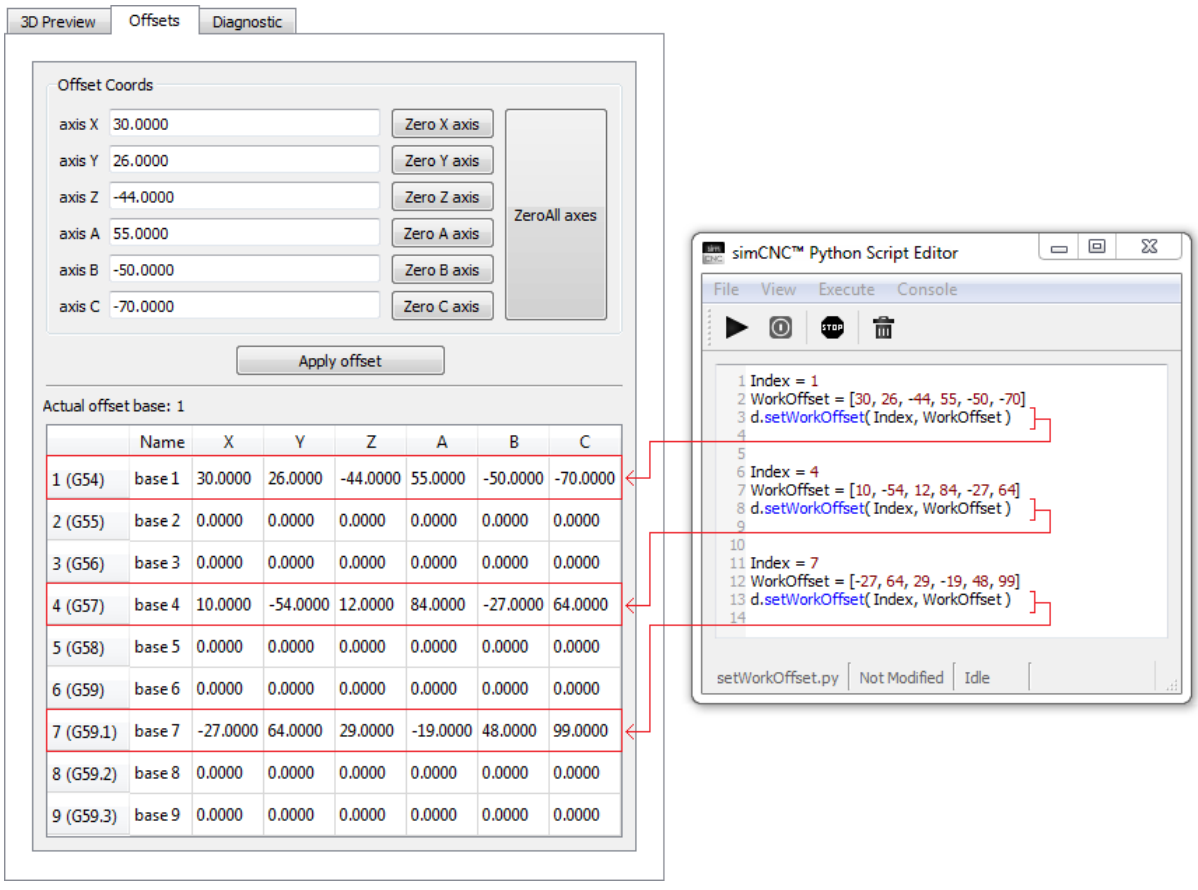
ARGUMENTY:

int workOffsetIndex - index (numer) offsetu roboczego
List<float>[6] workOffsetValues - nowe wartości 6 współrzędnych (X, Y, Z, A, B, C) offsetu roboczego

PRZYKŁAD:

```
Index = 1
WorkOffset = [30, 26, -44, 55, -50, -70]
d.setWorkOffset( Index, WorkOffset )

Index = 4
WorkOffset = [10, -54, 12, 84, -27, 64]
d.setWorkOffset( Index, WorkOffset )

Index = 7
WorkOffset = [-27, 64, 29, -19, 48, 99]
d.setWorkOffset( Index, WorkOffset )
```

Gdy uruchomisz powyższy przykład offsety roboczy o indeksie 1 (G54), 4 (57) i 7 (G59.1) zostaną ustawione na nowe wartości współrzędnych [30, 26, -44, 55, -50, -70], [10, -54, 12, 84, -27, 64] i [-27, 64, 29, -19, 48, 99].



setCurrentWorkOffset(List<float>[6] workOffsetValues) - Zmienia wartości współrzędnych aktualnego offsetu roboczego.

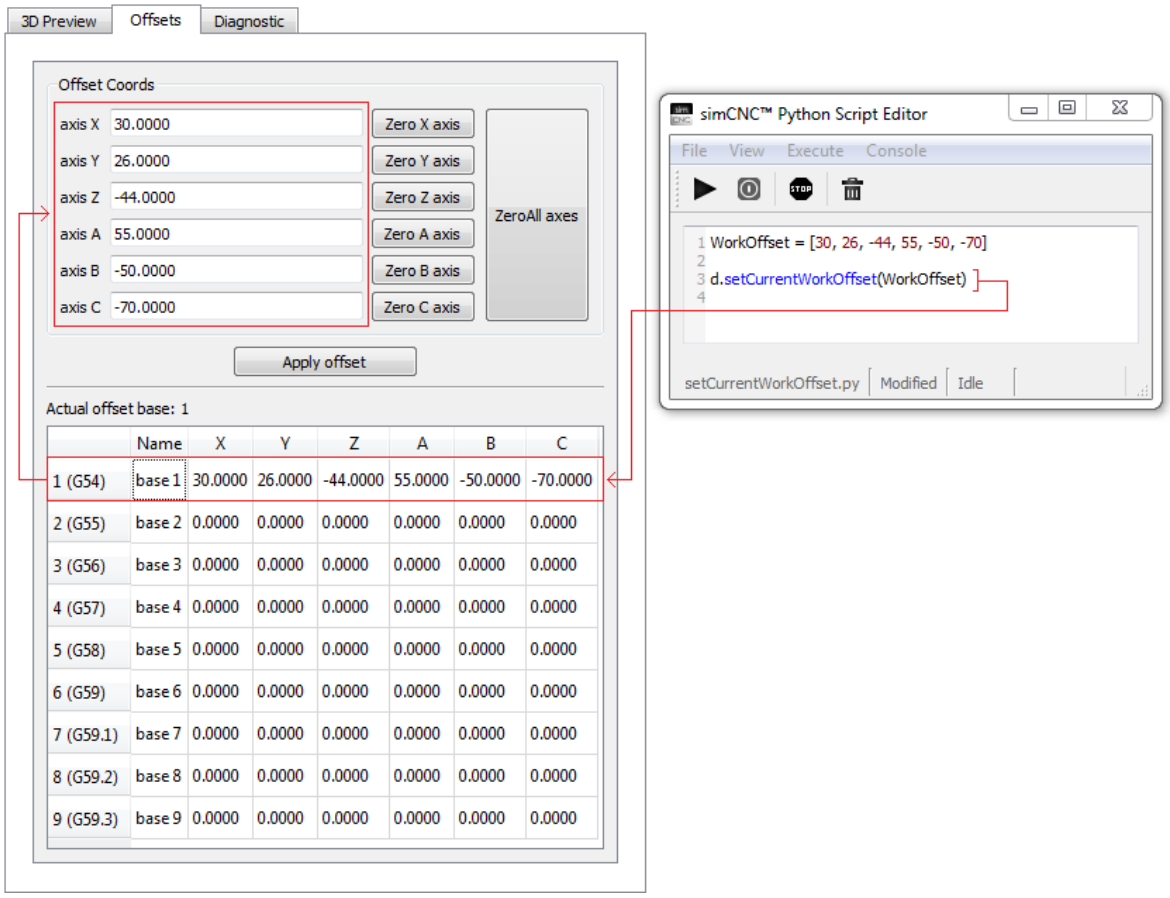
ARGUMENTY:

List<float>[6] workOffsetValues - nowe wartości 6 współrzędnych(X, Y, Z, A, B, C) aktualnego offsetu roboczego

PRZYKŁAD:

```
WorkOffset = [30, 26, -44, 55, -50, -70]

d.setCurrentWorkOffset( WorkOffset )
```

Gdy uruchomisz powyższy przykład offset roboczy o indeksie 1 (G54) zostanie ustawiony na nową wartość współrzędnych [30, 26, -44, 55, -50, -70].



setWorkOffsetNumber(int workOffsetIndex) - Zmienia index aktualnego offsetu roboczego.

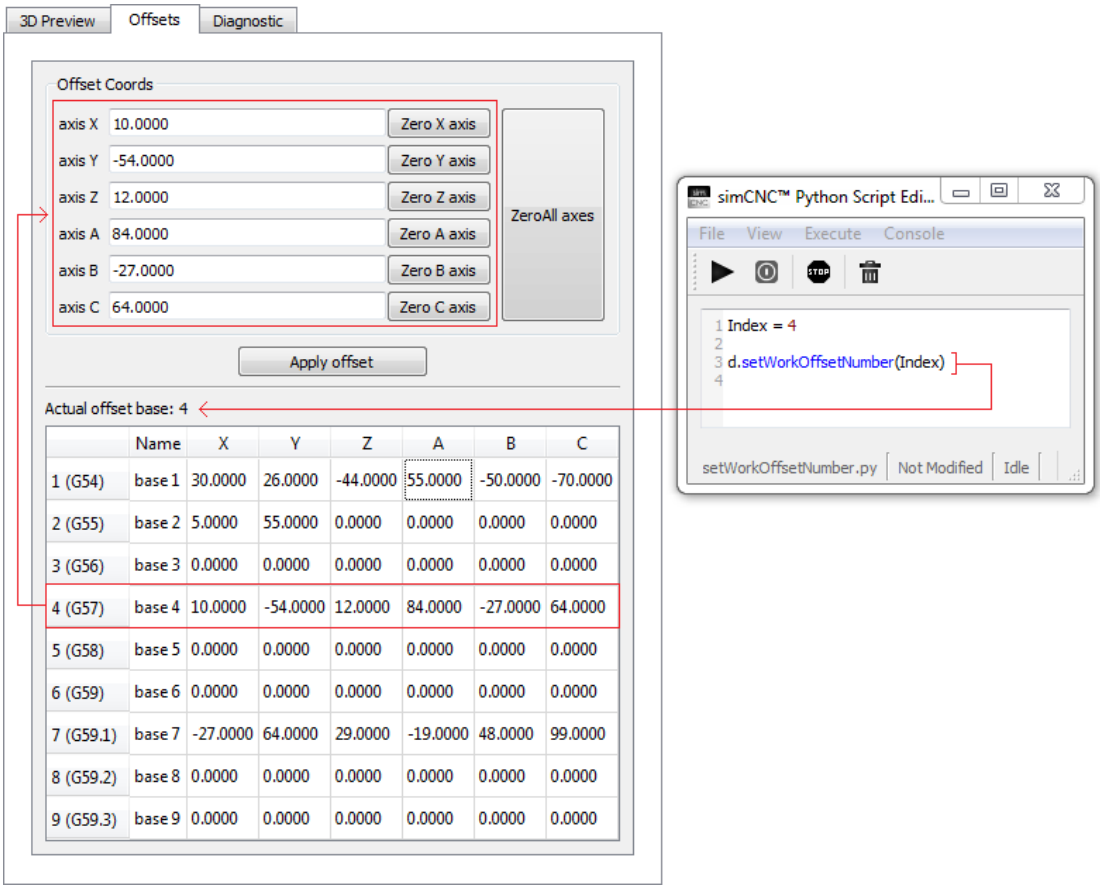
ARGUMENTY:

int workOffsetIndex – index (numer) offsetu roboczego

PRZYKŁAD:

```
Index = 4

d.setWorkOffsetNumber( Index )
```

Gdy uruchomisz powyższy przykład aktualnie używany offset roboczy zostanie zmieniony na offset roboczy o indexie 4 (G57).



	Name	X	Y	Z	A	B	C
1 (G54)	base 1	30.0000	26.0000	-44.0000	55.0000	-50.0000	-70.0000
2 (G55)	base 2	5.0000	55.0000	0.0000	0.0000	0.0000	0.0000
3 (G56)	base 3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4 (G57)	base 4	10.0000	-54.0000	12.0000	84.0000	-27.0000	64.0000
5 (G58)	base 5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6 (G59)	base 6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7 (G59.1)	base 7	-27.0000	64.0000	29.0000	-19.0000	48.0000	99.0000
8 (G59.2)	base 8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
9 (G59.3)	base 9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000