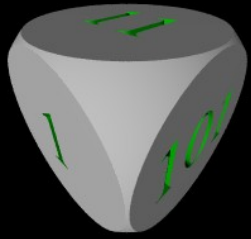




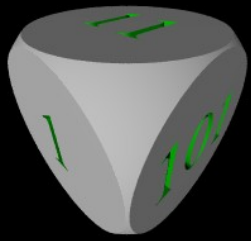
KOŁO NAUKOWE TWÓRCÓW GIER

POLYGON

POLITECHNIKA WARSZAWSKA



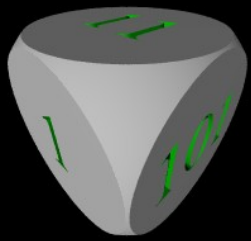
# Wstęp do programowania w pythonie.



# Obowiązkowa literatura



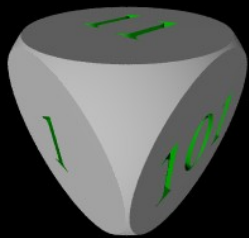
- [www.python.org](http://www.python.org) -> documentation
- [www.python.org.pl](http://www.python.org.pl)



# Python – język skryptowy



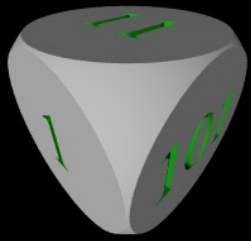
- Konsola języka python
- Skrypty w języku python
- Nie ma **ŻADNEJ** praktycznej różnicy między pythonem – językiem programowania, a pythonem – językiem powłoki



## Instalacja pythona (Windows)



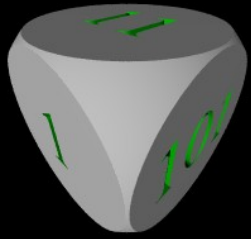
- <http://www.python.org/download/>
- Mój komputer (prawym klawiszem myszy) -> Właściwości -> zaawansowane -> zmienne środowiskowe
- Edytujemy zmienną path w sekcji „zmienne systemowe”
- Dodajemy na jej początek folder z binariami pythona (np. „C:\python26\bin;” lub „C:\python31\bin;”)



# Instalacja pythona (Linux)



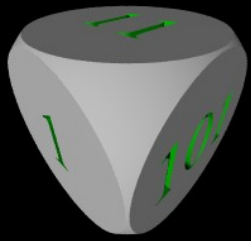
- Instalujemy pakiet „python”:
  - `sudo apt-get install python`



# Pliki programu w pythonie



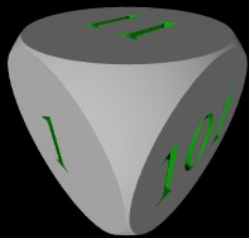
- \*.py – pliki źródłowe
- \*.pyw – pliki źródłowe programów okienkowych
- \*.pyc – pliki przechowujące skompilowane wersje programów



# Założenia pythona



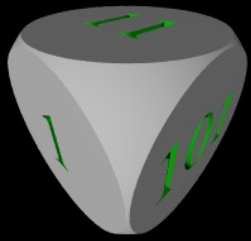
- W pełni obiektowy
- W pełni dynamiczny
- Język interpretowany (możliwość uruchamiania na wielu platformach)



## Wersje pythona



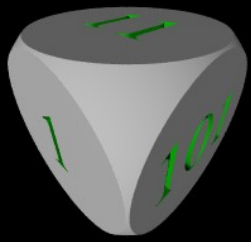
- Gałąź 2 (najnowsza wersja: 2.6.5)
  - Kompatybilna wstecz
- Gałąź 3 (najnowsza wersja: 3.1.2)
  - Niekompatybilna wstecz
    - Usunięte wszystkie zdeprecjonowane elementy



## Organizacja kodu



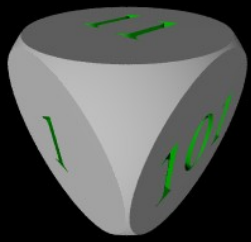
- Brak znaków oznaczających początek / koniec bloku
- Blok zagnieżdżony tworzymy zwiększając poziom wcięcia
- Jeśli piszemy każdą instrukcję w osobnej linijce, nie musimy stosować średników
- Według konwencji jedno wcięcie to cztery znaki spacji



# Zmienne w pythonie



- Brak zmiennych lokalnych
- Wszystko jest obiektem
- Nowe obiekty tworzymy wywołując konstruktor ich klasy
- Znak „=” „dowiązuje” obiekt po prawej do nazwy po lewej
- Dowiązanie puste jest osiąganane przez dowiązanie obiektu None



# Instrukcja warunkowa



**if warunek:**

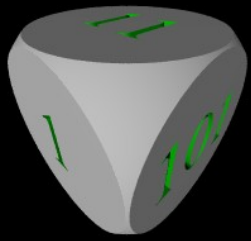
Wcięcie!     **instrukcja1**

Wcięcie!     **instrukcja2; instrukcja 3**

**else:**

Wcięcie!     **instrukcjaA**

Wcięcie!     **instrukcjaB**



## Pętla while



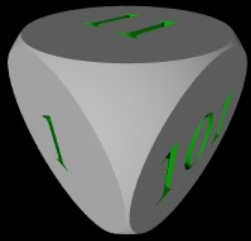
while warunek:

instrukcja1

instrukcja2

instrukcja3

- Brak instrukcji do-while



## Pętla for



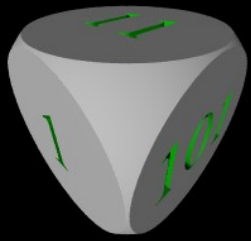
- Istnieje wyłącznie pętla for-each

for elem in elements:

    elem.do\_sth()

    instrukcja2

    instrukcja3



# Funkcje w pythonie



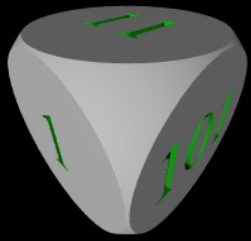
- Brak możliwości przeciążania
- Nie trzeba określać typu danych wejściowych
- Można stosować wartości domyślne

```
def nazwa_funkcji(arg1, msg="Hello world", arg3 = None):
```

```
    instrukcja1
```

```
    instrukcja2
```

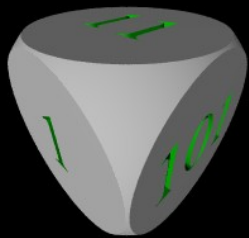
```
    instrukcja3
```



## Wywoływanie funkcji



- Przekazywanie argumentów zawsze przez referencję
- Argumenty przekazywane w.g. Kolejności
- Argumenty przekazywane w.g. Nazwy („keyword arguments”, „kwargs”)



# Klasy w pythonie



```
class nazwa_klasy:
```

```
    zmienna_klasowa_1 = 1;
```

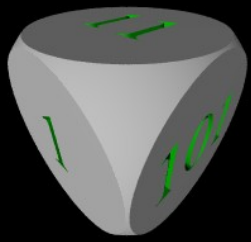
```
    def metoda(self, arg1, arg2):
```

```
        self.zmienna_publiczna = None;
```

```
        self._zmienna_chroniona = 3;
```

```
        self.__zmienna_prywatna = 127;
```

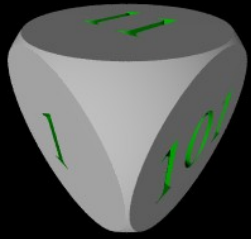
```
    zmienna_klasowa_2 = 2;
```



## Standardowe typy



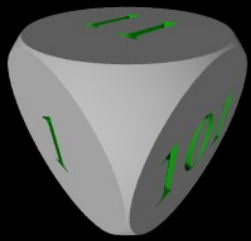
- NoneType
- str
- int
- tuple
- list
- dict
- class
- callable
- double



# Definiowanie NoneType



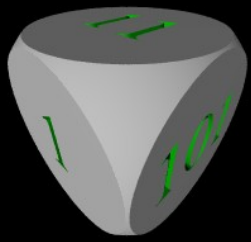
- W programie istnieje dokładnie jeden obiekt typu NoneType
- Nazywa się None



## Definiowanie liczb



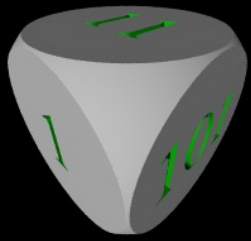
- `liczba_typu_int = 1`
- `liczba_typu_double = 3.0`
- `2/3` – dzielenie całkowite
- `2/3.0` – dzielenie zmiennoprzecinkowe
- `2.0/3` – dzielenie zmiennoprzecinkowe
- `2.0/3.0` – dzielenie zmiennoprzecinkowe



## Definiowanie stringów



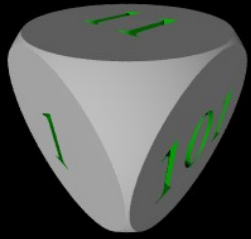
- "Hello World!!!"
- "'Hello World!!!" I shout'
- """Hello world  
in multiple lines"""
- """"And again...  
and again...""""
- 'And again\n\  
:P'



# Tuple'e



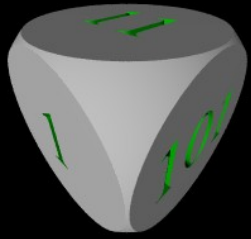
- Obiekty typu tuple są „immutable”
- (elem1, elem2, elemN)
- elem1, elem2, elemN
- (elem1,)
- Przypisanie do Tuple'i:  $a, b = 2, 3$
- Umożliwiają zwrócenie wielu wartości z funkcji



## list



- [elem1, elem2, elemN]
- Obiekty typu list są „mutable”



dict



- $a = \{1:\text{"Jeden"}, 2:\text{"Dwa"}, \text{"Trzy"}:3\}$
- Klucze muszą być immutable, więc mogą to być tylko liczby typu int i obiekty typu string