

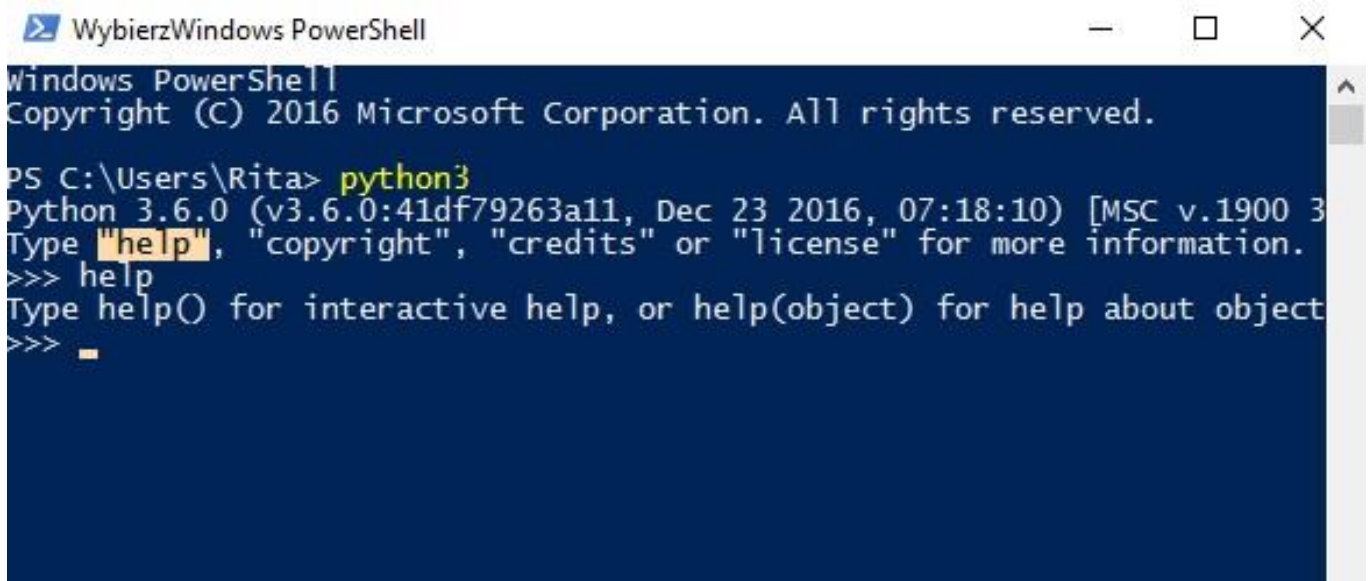
## Python #2: pyta i odpowiada

Najprościej mówiąc: **coś ma swoją funkcję** – tzn, że spełnia określone zadanie. To jest bardzo dobre wytłumaczenie i można je przenieść do Pythona.

- **funkcja** – to fragment kodu, który wykonuje jakąś sekwencję poleceń. Może przyjmować *argumenty*.
- **argumenty** – dane niezbędne do wykonania funkcji

### Funkcje wbudowane w Pythonie

Do tej pory na pewno znasz już funkcję **print()**, która wyświetla treści oraz funkcję **str()**, która przekształca inne typy np. liczby w napisy czyli stringi. Czy można ten proces odwrócić? Oczywiście do zmiany napisu w liczbę wystarczy funkcja **int()**

A screenshot of a Windows PowerShell window titled "WybierzWindows PowerShell". The window shows the Python 3.6.0 help screen. The text in the window is: "Windows PowerShell\nCopyright (C) 2016 Microsoft Corporation. All rights reserved.\n\nPS C:\\Users\\Rita> python3\nPython 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32-bit AMD64]\nType \"help\", \"copyright\", \"credits\" or \"license\" for more information.\n>>> help\nType help() for interactive help, or help(object) for help about object\n>>> \_"

Python zachęca, by skorzystać z funkcji **help()** wyświetlającej pomoc do Pythona. Jeśli chcecie uzyskać więcej szczegółowych informacji o konkretnej funkcji to możecie zapytać – np. **help(print)**. By zamknąć Pythona w konsoli wystarczy użyć funkcji **quit()**.

Funkcją, która będzie działała odwrotnie od wyświetlania będzie odczytywanie **input()**. Funkcja **input()** wprowadzone przez użytkownika znaki z klawiatury **zapisuje jako string**.

```
>>> input()
Przykładowe zdanie z klawiatury
Przykładowe, zdanie z klawiatury
```

Python czekał na wpisanie „Przykładowe zdanie z klawiatury”, a następnie je wyświetlił – funkcja wykonała się, a efektem było zapisanie tego zdania.

```
>>> herbata = input()
zielona z jaśminem
>>> print("Moja ulubiona herbata to", herbata)
Moja ulubiona herbata to zielona z jaśminem
```

Po sczytaniu przez funkcję **input()** to co zostało wprowadzone z klawiatury zamiast od razu wyświetlić, Python zapamiętał to w zmiennej o nazwie **herbata**.

Dopiero przez użycie **print()** wyświetlił zawartość zmiennej.

Mam nadzieję, że do tej pory wszystko wydaje się jasne?

Spróbujmy z liczbami:

```
>>> input()
365
,365'
>>> liczba = input()
45
>>> print(liczba)
'45'
>>> print(liczba + 3)
File „”, line 1, in
TypeError: must be str, not int
```

Ten błąd nie powinien nikogo zaskoczyć – skoro **input()** czytuje dane z klawiatury jako napisy, nic dziwnego, że do napisu „45” nie możemy dodać 3. Wspomniana wcześniej funkcja **int()** pomoże wykonując **konwersję string do int**.

```
>>> print(int(liczba) + 3)
48
```

Funkcję tę można by było umieścić też wcześniej zaraz po zczytaniu liczby:

```
>>> liczba = input()
45
>>> print(liczba)
'45'
>>> liczba = int(liczba)
>>> print(liczba)
45
```

*Widzimy różnicę? Liczba przestała wyświetlać się w apostrofie, więc wiemy, że nie mamy już do czynienia z napisem.*

Lub **od razu traktować jako liczbę**:

```
>>> liczba = int(input())
45
>>> print(liczba)
45
```

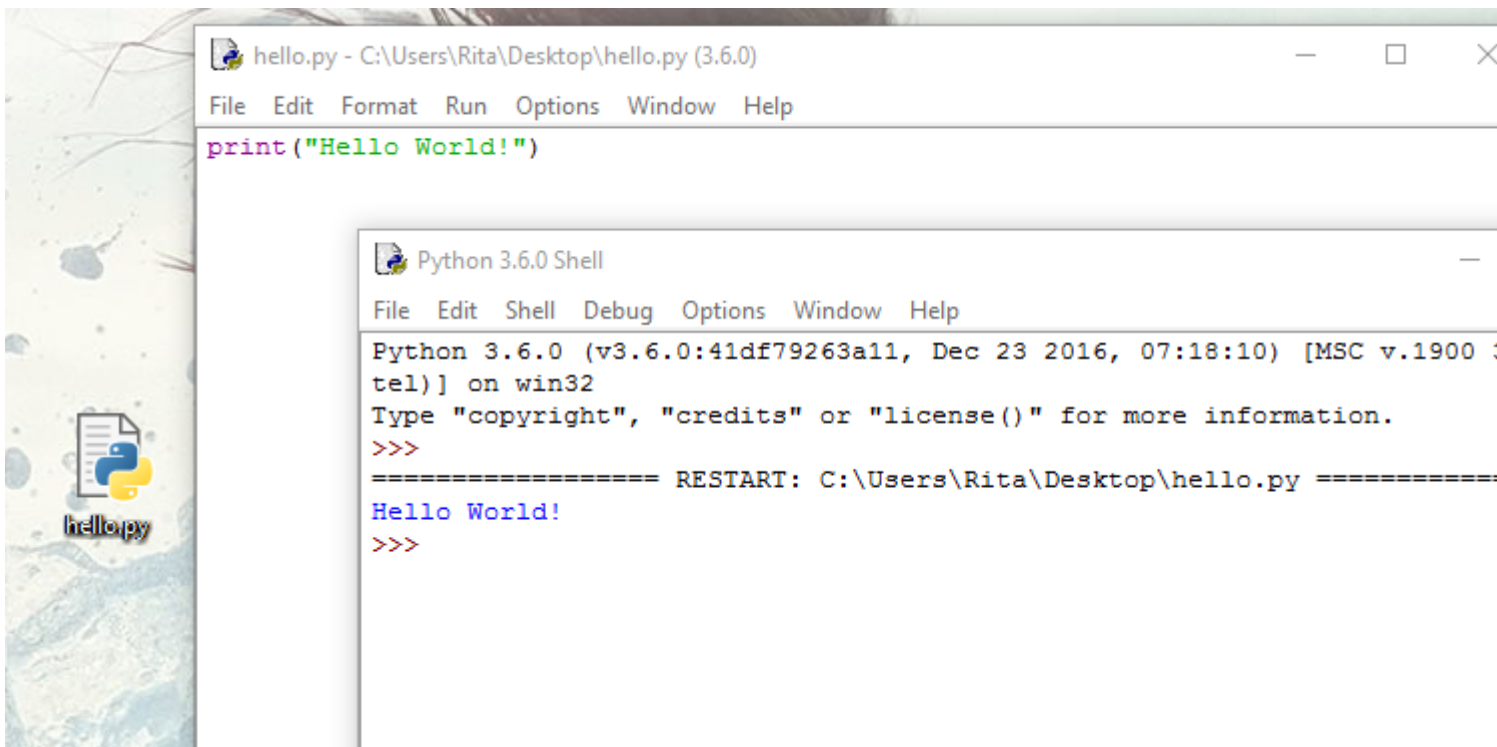
W typie liczbowym znajdują się również liczby zmiennoprzecinkowe – float. Jak się można domyśleć dla nich funkcja będzie wyglądać **float()** i działać analogicznie do **int()**.

Funkcji wbudowanych w Pythonie jest znacznie więcej, ale nie ma co się na początek przemęczać 😊

Mamy co ćwiczyć na dziś.

Zamknij Pythona w konsoli poleceniem **quit()**. Konsolę możesz zminimalizować.

W dowolnym folderze utwórz nowy plik o dowolnej nazwie i rozszerzeniu .py. Możesz użyć utworzonego po instalacji hello.py.



W treści umieść:

```
>>> print("Hej!")
```

Wiemy już jak uruchomić skrypt z poziomu IDLE.

Teraz zrobimy to z konsoli.

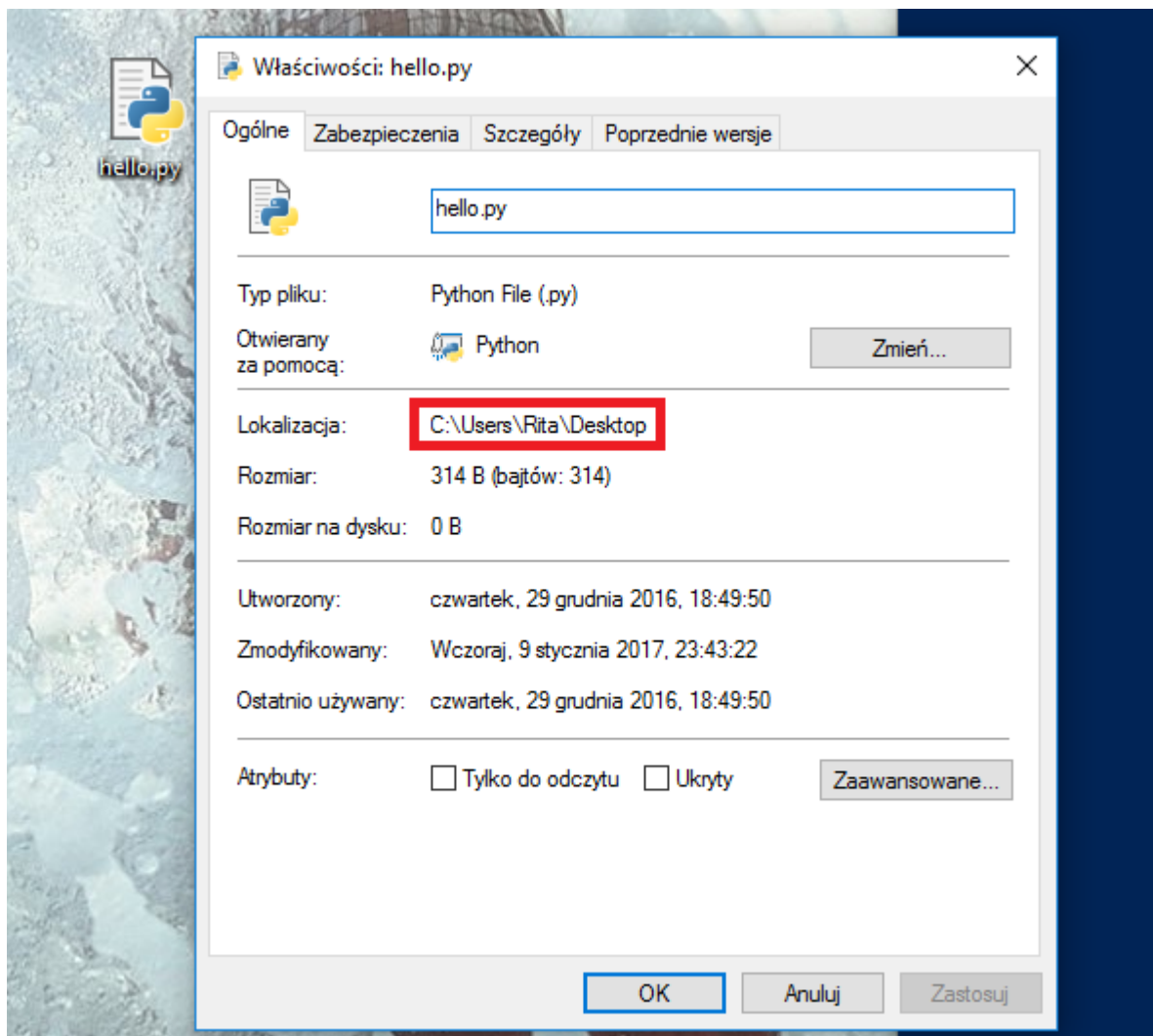
## Uruchamianie skryptu z konsoli

Możemy to zrobić na dwa sposoby.

### 1) python \ścieżka-do-pliku

Ścieżkę można zobaczyć w nazwie okna w IDLE, ale nie o to chodzi

Ścieżka do pliku to miejsce, gdzie dokładnie znajduje się nasz plik. W Windowsie czy w Linuxie wystarczy na wybrany plik kliknąć prawym przyciskiem i zobaczyć właściwości. W Mac'u należy wybrać Plik > Informacje.



Z tego wynika, że mój plik znajduje się na pulpicie, a jego ścieżka to `C:\Users\Rita\Desktop\hello.py`.

W Mac'ach trzeba zwrócić uwagę na kierunek slashy, są odwrócone „/” względem windowsowych „\”.

Może tłumaczyć rzeczy zupełnie banalne, ale lepiej, żeby nie było wątpliwości. 😊

Mamy ścieżkę, więc możemy uruchomić plik poleceniem:

```
python C:\Users\Rita\Desktop\hello.py
```

*(ścieżka bezwzględna do pliku)*

W uproszczeniu

```
python .\Desktop\hello.py
```

*(ścieżka względna do pliku)*

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Rita> python C:\Users\Rita\Desktop\hello.py
Hej!
PS C:\Users\Rita> python .\Desktop\hello.py
Hej!
PS C:\Users\Rita>
```

Ponieważ w momencie uruchamiania znajduję się w katalogu **C:\Users\Rita**, to nie muszę podawać całej ścieżki, tylko **ścieżkę względem tego katalogu**, „w którym jestem”.

Kropka przed `\Desktop` oznacza dosłownie „tutaj \wybierz Desktop\ wybierz plik hello.py”.

## 2) python nazwa\_pliku.py

Żeby uruchomić plik bez podawania ścieżki musisz znajdować się w tej samej lokalizacji co plik.

Jak to zrobić?

W konsoli wpisujesz `cd \ścieżka-do-folderu`, w którym znajduje się plik i klikasz enter.

W moim przypadku:

```
cd .\Desktop
```

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Rita>
PS C:\Users\Rita> cd .\Desktop
PS C:\Users\Rita\Desktop>
```

Jestem teraz w katalogu Pulpitu, a w takim razie wystarczy wpisać **python hello.py**, aby skrypt zadziałał.

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

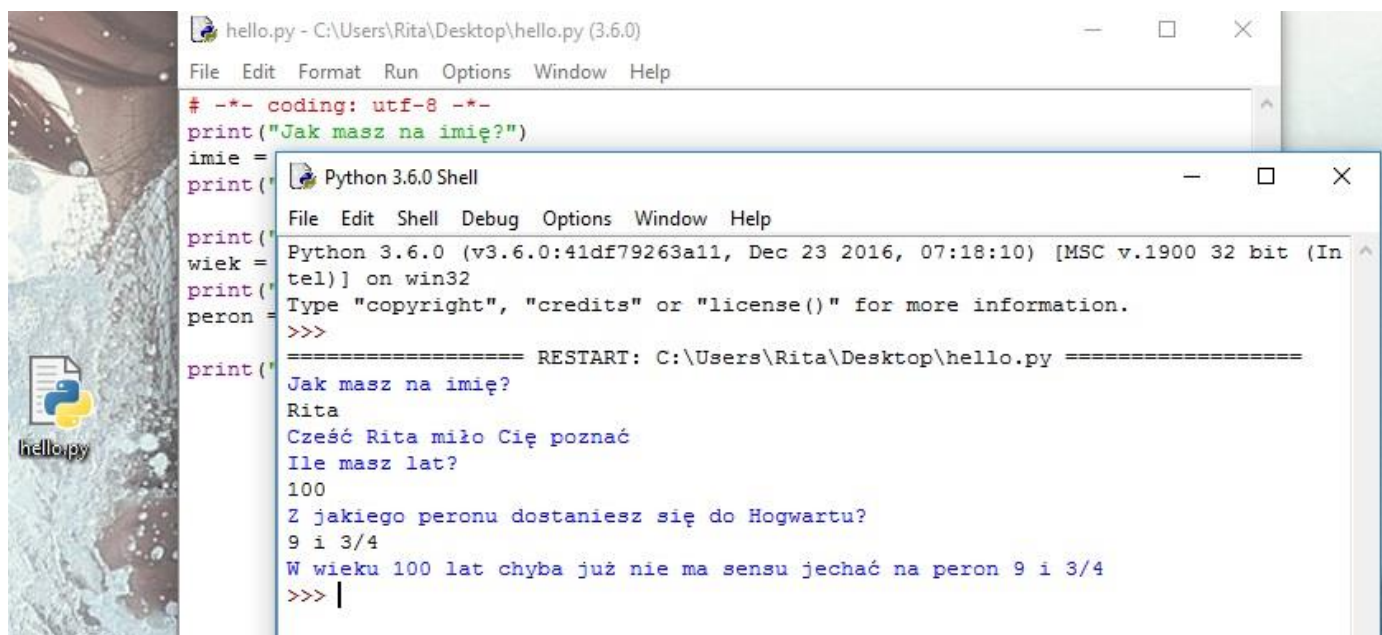
PS C:\Users\Rita> cd .\Desktop
PS C:\Users\Rita\Desktop> python hello.py
Hej!
PS C:\Users\Rita\Desktop>
```

## Zadanie 1

Teraz chcemy, tak aby skrypt:

- Zapytał o Twoje imię
- Powitał Cię po imieniu
- Zapytał o Twój wiek
- Zapytał o peron z Harrego Potter'a
- Odpowiedział (dowolnie łącząc wiek i peron), czy jest sens jechać do Hogwartu

Korzystając tylko z tego co już wiesz jesteś w stanie napisać taki skrypt:



```
hello.py - C:\Users\Rita\Desktop\hello.py (3.6.0)
File Edit Format Run Options Window Help

# -*- coding: utf-8 -*-
print("Jak masz na imię?")
imie = input()
print(imie)

print("Ile masz lat?")
wiek = input()
print(wiek)
peron = input("Z jakiego peronu dostaniesz się do Hogwartu?")
print(peron)

Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Rita\Desktop\hello.py =====
Jak masz na imię?
Rita
Cześć Rita miło Cię poznać
Ile masz lat?
100
Z jakiego peronu dostaniesz się do Hogwartu?
9 i 3/4
W wieku 100 lat chyba już nie ma sensu jechać na peron 9 i 3/4
>>>
```

Spróbuj napisać własny kod.

Aby zapobiec błędom związanym z różnym kodowaniem znaków ASCII, na początku skryptu dodajemy nagłówek:

```
# -*- coding: utf-8 -*-
```

Normalnie znak # rozpoczyna linię komentarza, czyli nasze uwagi do kodu, które nie wpływają sam kod i nie są nigdzie wyświetlane. W tym wypadku jednak nagłówek informuje interpreter, że kodowanie znaków jest ustawione na **utf-8** – czyli kodowanie zawierające również *polskie znaki* takie jak *ą, ę, ć* etc. Mimo to nie używamy polskich znaków w nazwach zmiennych, a jedynie stosujemy je w wyświetlanych napisach.

## Zadanie 2 – kalkulator BMI

Zapisz kalkulator BMI, który wcześniej był w konsoli do pliku, tak aby pytał użytkownika o potrzebne do obliczeń dane.

## Zadanie 3

Spróbuj to samo z kalkulatorem kalorii. Dla uproszczenia przyjmij, że jest to kalkulator dla kobiet lub dla mężczyzn.