



AKADEMIA GÓRNICZO-HUTNICZA
im. Stanisława Staszica w Krakowie



WYDZIAŁ ZARZĄDZANIA

STUDIA STACJONARNE II stopnia

KIERUNEK: Informatyka i Ekonometria

PRZEDMIOT: Przetwarzanie i analiza danych w języku Python

PROWADZĄCY: prof. dr hab. inż. Oleksandr Petrov

Sprawozdanie 1

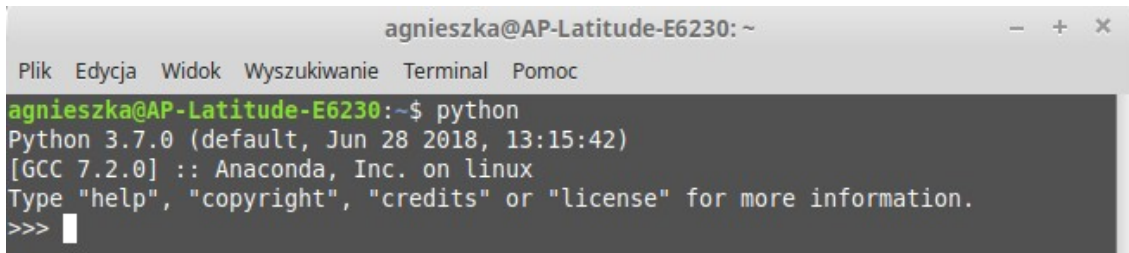
Agnieszka Pytel

Kraków, 2018/2019

1) Zadanie 1.1.

Zainstaluj środowisko Python w wersji 3.x zawarte w dystrybucji Anaconda.

Instalacja została przeprowadzona przed rozpoczęciem zajęć z przedmiotu. Sprawdzono wersję środowiska Python zainstalowanego na komputerze:



```
agnieszka@AP-Latitude-E6230: ~  
Plik Edycja Widok Wyszukiwanie Terminal Pomoc  
agnieszka@AP-Latitude-E6230:~$ python  
Python 3.7.0 (default, Jun 28 2018, 13:15:42)  
[GCC 7.2.0] :: Anaconda, Inc. on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

Na komputerze działa Python w wersji 3.7.0 w dystrybucji Anaconda.

Program Anaconda-Navigator można uruchomić z wiersza poleceń komendą *anaconda-navigator*.

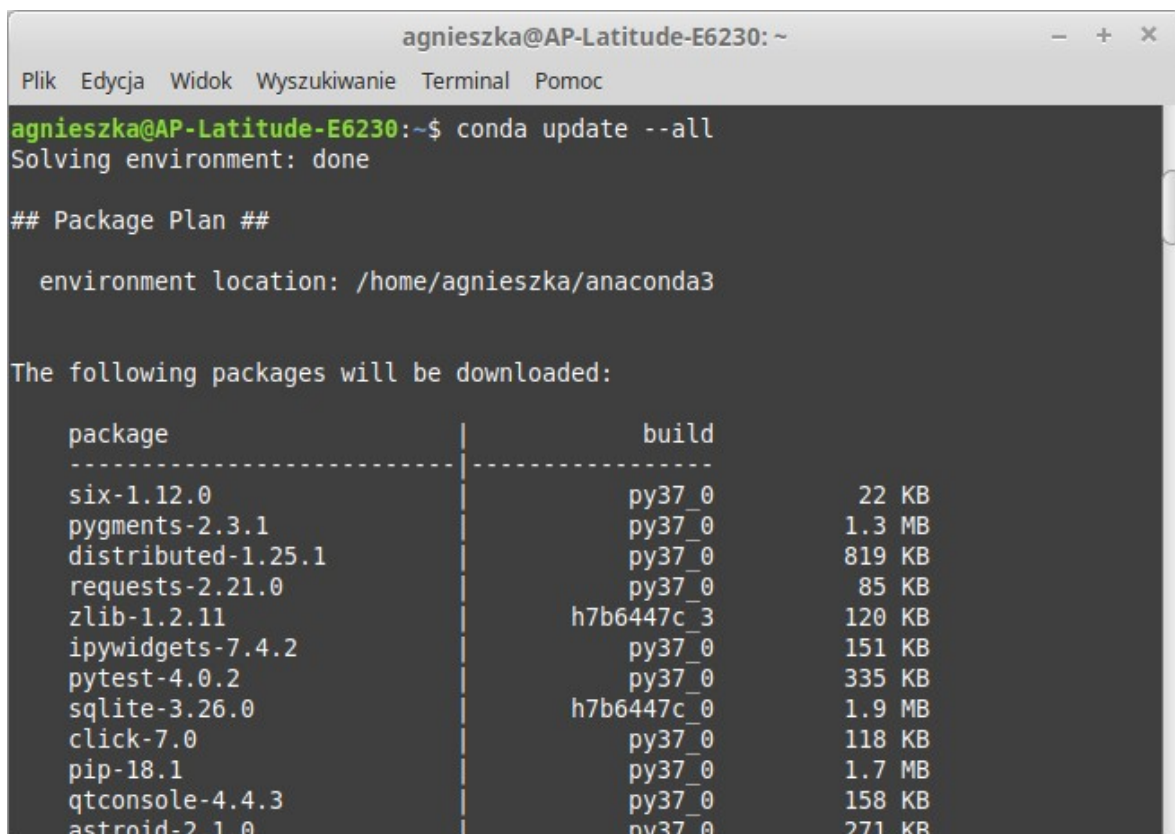
2) Zadanie 1.2.

Zaktualizuj wszystkie zainstalowane pakiety, wywołując w powłoce:

\$ sudo /opt/anaconda3/bin/conda update —all

lub w wierszu poleceń: [anaconda3] > conda update —all

W wierszu poleceń wykonano komendę *conda update -all* w celu aktualizacji oprogramowania i paczek do najnowszej wersji:



```
agnieszka@AP-Latitude-E6230: ~  
Plik Edycja Widok Wyszukiwanie Terminal Pomoc  
agnieszka@AP-Latitude-E6230:~$ conda update --all  
Solving environment: done  
  
## Package Plan ##  
  
environment location: /home/agnieszka/anaconda3  
  
The following packages will be downloaded:  
  
package | build | size  
-----|-----|-----  
six-1.12.0 | py37_0 | 22 KB  
pygments-2.3.1 | py37_0 | 1.3 MB  
distributed-1.25.1 | py37_0 | 819 KB  
requests-2.21.0 | py37_0 | 85 KB  
zlib-1.2.11 | h7b6447c_3 | 120 KB  
ipywidgets-7.4.2 | py37_0 | 151 KB  
pytest-4.0.2 | py37_0 | 335 KB  
sqlite-3.26.0 | h7b6447c_0 | 1.9 MB  
click-7.0 | py37_0 | 118 KB  
pip-18.1 | py37_0 | 1.7 MB  
qtconsole-4.4.3 | py37_0 | 158 KB  
astroid-2.1.0 | py37_0 | 271 KB
```

3) Zadanie 1.3.

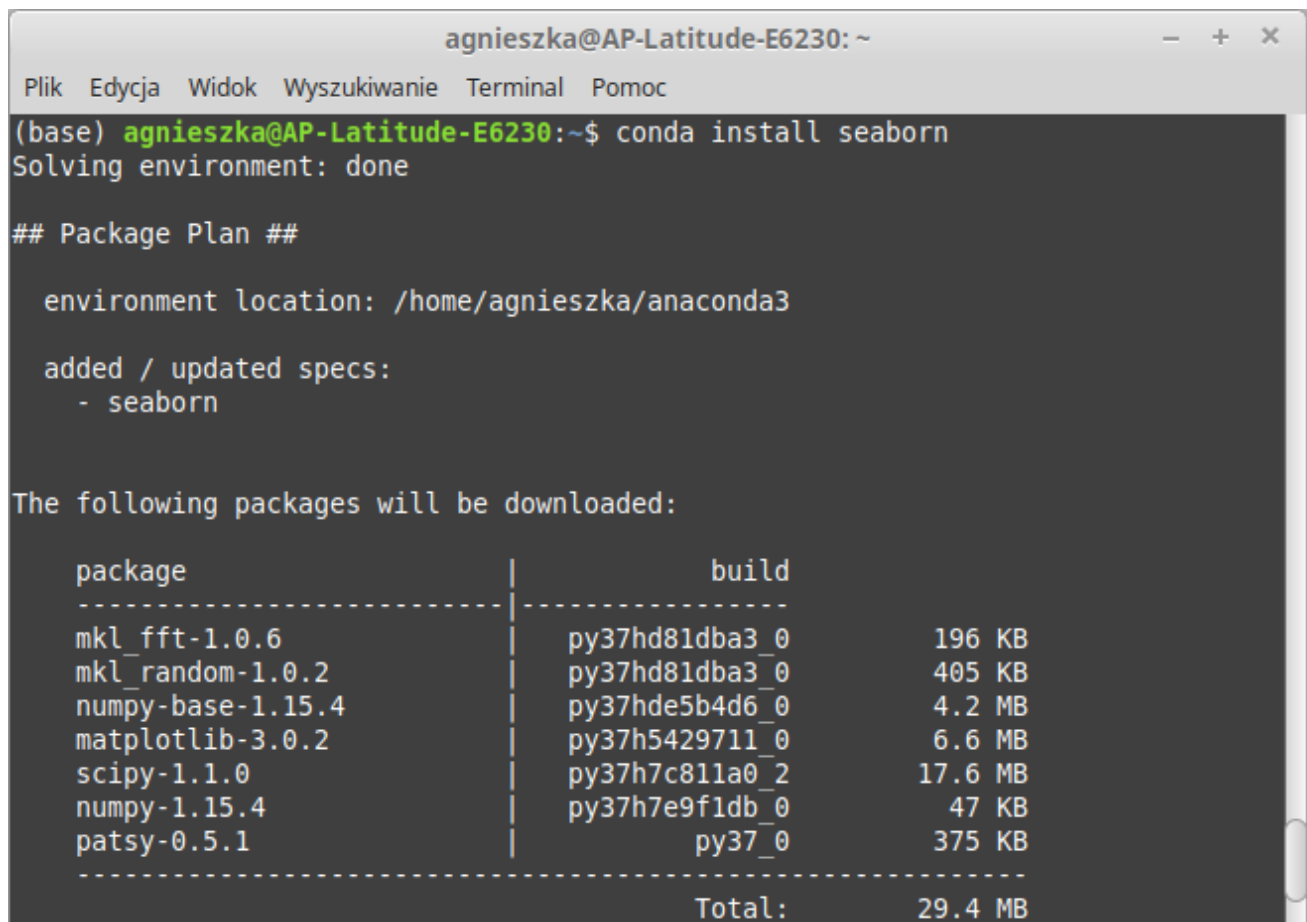
Zainstaluj pakiet seaborn, wykonując polecenie w powłoce:

```
$ sudo /opt/anaconda3/bin/conda install seaborn
```

lub w wierszu poleceń:

```
[anaconda3] > conda install seaborn
```

Zainstalowano pakiet **seaborn** poleceniem *conda install seaborn*:



```
agnieszka@AP-Latitude-E6230: ~
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
(base) agnieszka@AP-Latitude-E6230:~$ conda install seaborn
Solving environment: done

## Package Plan ##

environment location: /home/agnieszka/anaconda3

added / updated specs:
- seaborn

The following packages will be downloaded:

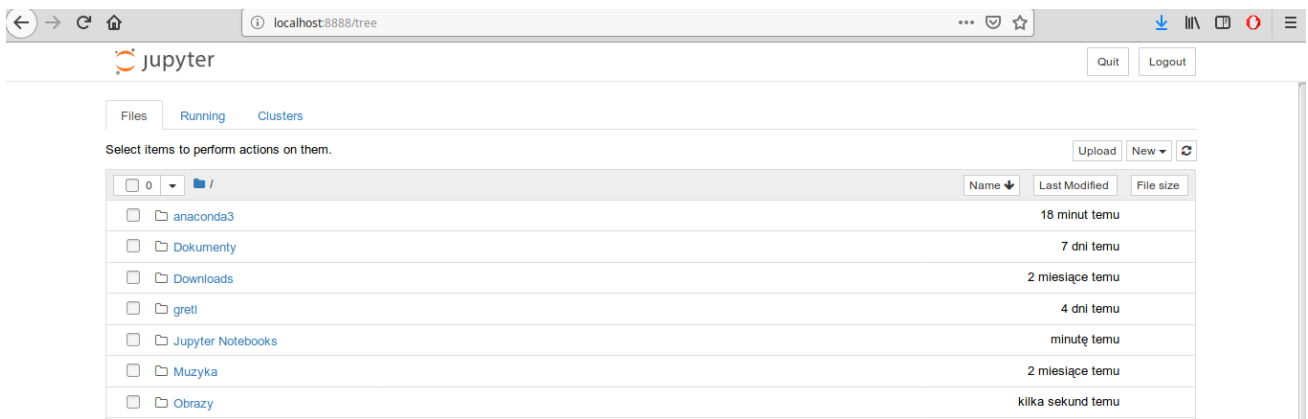
package | build | size
-----|-----|-----
mkl_fft-1.0.6 | py37hd81dba3_0 | 196 KB
mkl_random-1.0.2 | py37hd81dba3_0 | 405 KB
numpy-base-1.15.4 | py37hde5b4d6_0 | 4.2 MB
matplotlib-3.0.2 | py37h5429711_0 | 6.6 MB
scipy-1.1.0 | py37h7c811a0_2 | 17.6 MB
numpy-1.15.4 | py37h7e9f1db_0 | 47 KB
patsy-0.5.1 | py37_0 | 375 KB
-----|-----|-----
Total: | 29.4 MB
```

4) Zadanie 1.4.

Uruchom serwer Jupyter, wywołując:

klikając Menu Start -> Anaconda3 -> Jupyter Notebook pod Windows.

Uruchomiono serwer Jupyter pod Linuxem wywołując w terminalu polecenie *jupyter notebook*. Po wykonaniu polecenia użytkownik został automatycznie przekierowany na kartę w przeglądarce z otwartym serwerem Jupyter:

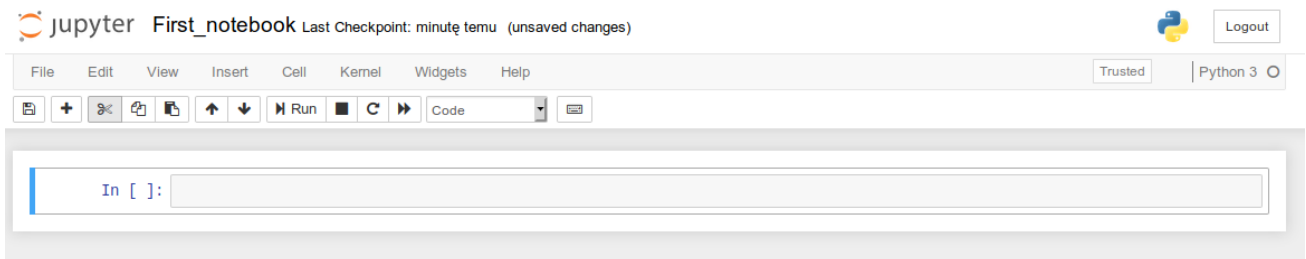


5) Zadanie 1.5.

Zapoznaj się ze strukturą menu w oknie notatnika. Zwroć uwagę m.in. na opcję

Kernel —> Interrupt, która pozwala przerywać zbyt długo wykonujące się obliczenia.

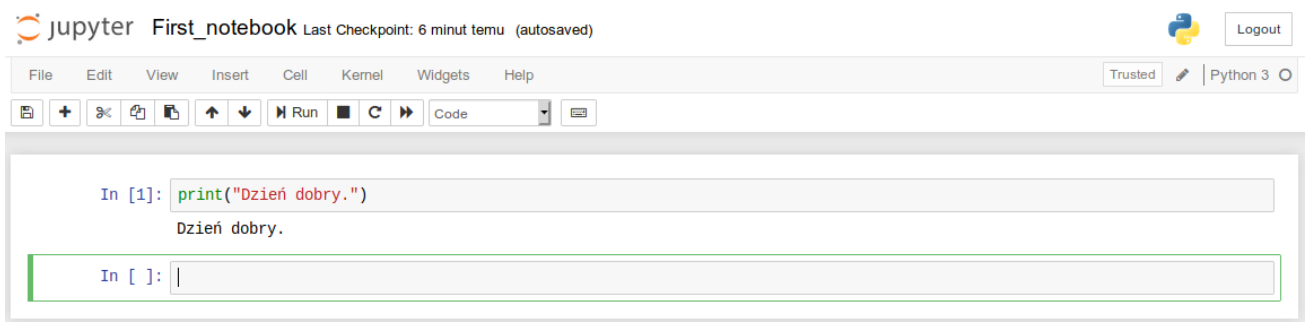
W menu notatnika znajdują się przyciski pozwalające na szybki zapis pliku, dodanie nowej linii, wycięcie, skopiowanie i wklejenie komórek, przeniesienie komórek wyżej lub niżej, uruchomienie kodu, zatrzymanie wykonywania kodu, ponowne uruchomienie kodu, ponowne uruchomienie kodu oraz całego notatnika, a także okno wyboru trybu edycji notatnika oraz przycisk otwierający paletę komend.



6) Zadanie 1.6.

Wprowadź w komórce notatnika polecenie `print("Dzień dobry.")`

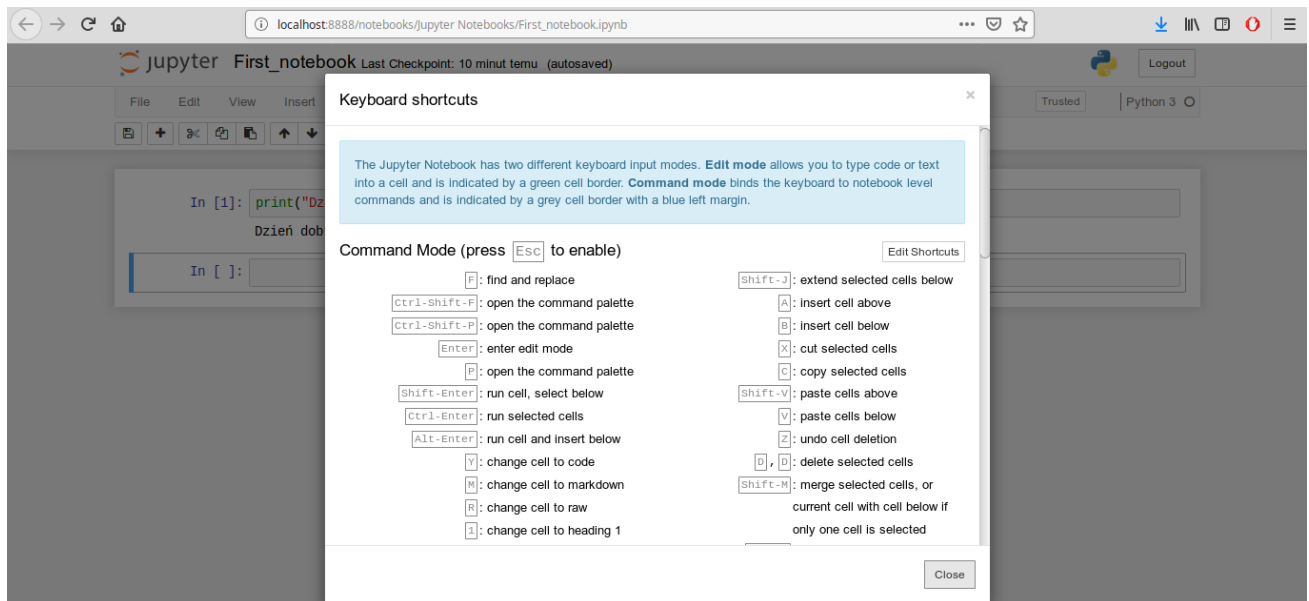
Wprowadzono polecenie `print(„Dzień dobry.”)` oraz uruchomiono notatnik:



7) Zadanie 1.7.

Wciśnij (H) w trybie poleceń i przeczytaj opis dostępnych skrótów klawiszowych.

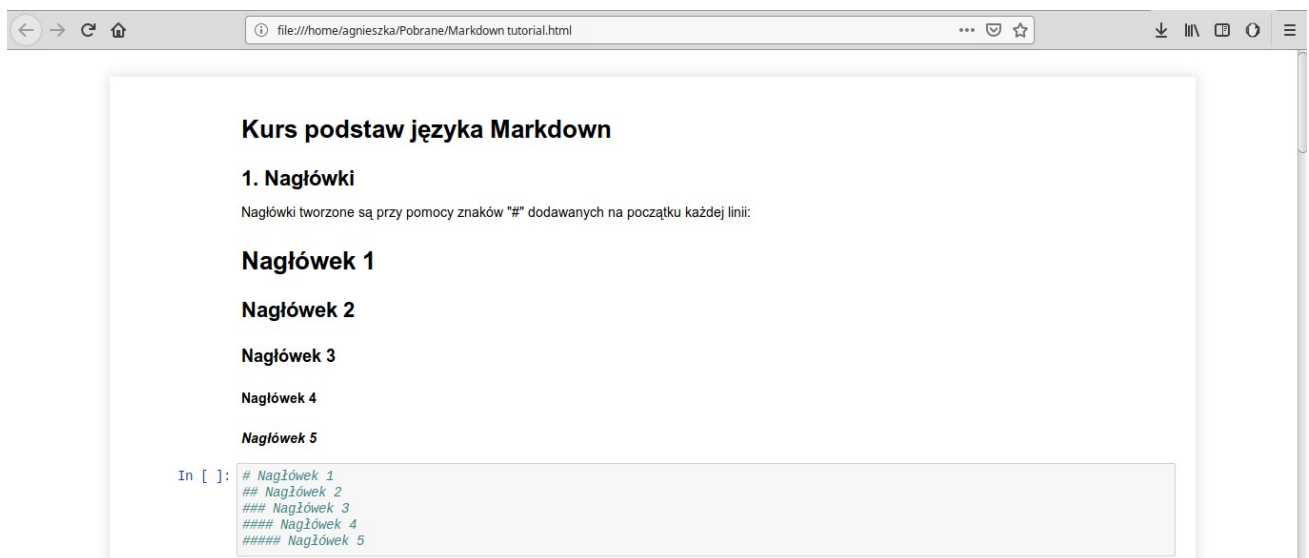
Zapoznano się z dostępnymi skrótami klawiszowymi:



8) Zadanie 1.8.

W notatniku Jupyter utwórz kurs podstaw języka Markdown (podobny do niniejszego) przy użyciu poleceń tego języka oraz wyeksportuj go do pliku w formacie HTML.

Poniżej przedstawiono screeny pliku html z utworzonym kursem podstaw języka Markdown:



2. Akapity, sekcje

Aby stworzyć nowy akapit, wystarczy dodać w odpowiednim miejscu dwie puste komórki.

3. Formatowanie tekstu

1. *Kursywa* - tworzona przy pomocy pojedynczych znaków `"~"`
2. **Pogrubienie** - tworzone przy pomocy podwójnych znaków `"=="`
3. ~~Przekreślenie~~ - tworzone przy pomocy podwójnych znaków `"~"`
4. Kod - tworzony przy pomocy znaków `"`"`

Kod:

```
print ("Dzień dobry")
```

1. Blok kodu - tworzony przy pomocy potrójnych znaków `"`"`

Blok kodu:

```
function add(a, b) {  
    int suma;  
    suma = a + b;  
    return suma;  
}
```

```
In [ ]: # *Kursywa*  
# **Pogrubienie**  
# ~~Przekreślenie~~  
# Kod:  
# `print ("Dzień dobry")`  
# Blok kodu:  
# ```  
#     function add(a, b) {  
#         int suma;  
#         suma = a + b;  
#         return suma;  
#     }  
# ```
```

4. Lista numerowana

Listę numerowaną tworzy się poprzez wstawienie cyfry na początku wiersza:

1. Punkt 1
2. Punkt 2
3. Punkt 3

Kolejne poziomy listy tworzy się analogicznie, dodatkowo dodając tabulator na początku linii:

1. Punkt 1
 - A. Punkt 1.1
 - a. Punkt 1.1.1
 - i. Punkt 1.1.1.1
 1. Punkt 1.1.1.1.1
2. Punkt 2

```
In [4]: # 1. Punkt 1  
# 2. Punkt 2  
# 3. Punkt 3  
  
#1. Punkt 1  
#     1. Punkt 1.1  
#         1. Punkt 1.1.1  
#             1. Punkt 1.1.1.1  
#                 1. Punkt 1.1.1.1.1  
#2. Punkt 2
```

5. Lista nienumerowana

Listę nienumerowaną tworzy się poprzez wstawienie znaku `"*"` lub `"-"` na początku wiersza:

- Punkt 1
- Punkt 2
- Punkt 3

- Punkt 1
- Punkt 2
- Punkt 3

```
In [ ]: # * Punkt 1  
# * Punkt 2  
# * Punkt 3  
  
# - Punkt 1  
# - Punkt 2  
# - Punkt 3
```

6. Wzory i formuły matematyczne

Można wprowadzać wzory i formuły matematyczne używając składni języka TEX:

Wzór: $y = f(x)$

Tryb blokowy:

$$f(x) = \sqrt{\frac{2a}{b}}$$

```
In [ ]: # Wzór: $y = f(x)$
# Tryb blokowy:
# $$ f(x) = \sqrt{\frac{2a}{b}} $$
```

7. Tabele

Tabele tworzymy używając znaków "|" oraz "-"

Lp.	Produkt	Cena
1.	Krzesło	100
2.	Stół	1000
3.	Szafka	300

```
In [6]: # | Lp. | Produkt | Cena |
# |-----|-----|-----|
# | 1. | Krzesło | 100 |
# | 2. | Stół | 1000 |
# | 3. | Szafka | 300 |
```

8. Linki

Linki wstawiamy używając nawiasów kwadratowych dla klikanego tekstu oraz nawiasów okrągłych dla linku:

[Google](www.google.com)

[Google](https://www.google.com)

9. Obrazki

Obrazki wstawiamy używając wykrzyknika, nawiasów kwadratowych dla nazwy alternatywnej oraz nawiasów okrągłych dla linku:

![Piłka](https://upload.wikimedia.org/wikipedia/commons/a/a5/Ball_icon.png)



9) Zadanie 2.1.

Wiedząc, że pierwiastek n -tego stopnia z x równa się x do potęgi $1/n$ i wykorzystując wiedzę o użyciu liczb zespolonych w Pythonie, wylicz wartość pierwiastka drugiego stopnia z liczby -17 .

```
In [7]: result = complex((-17)**(1/2))
print(result)
(2.5246740534795566e-16+4.123105625617661j)
```

10) Zadanie 2.2.

Używając instrukcji Pythona oblicz resztę z dzielenia 17 przez 7 i zapamiętaj wynik w zmiennej o nazwie Z . Następnie, pojedynczym poleceniem Pythona i bez użycia nawiasów, przemnoż zmienną Z przez $Z+3$.

```
In [8]: z = 17%7
print(z)
z *= z+3
print(z)
3
18
```

11) Zadanie 2.3.

Spowoduj pojedynczym poleceniem Pythona, by na ekranie 20-krotnie wyświetliła się wartość wyrażenia $1.2e+3+34.5$ każdorazowo rozdzielona średnikiem.

```
In [17]: result = 20*(str(1.2e+3+34.5)+";")
print(result)
1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;1234.5;
```

12) Zadanie 3.1

Napisz program „powiel.py”, który wczyta od użytkownika pewien napis, a następnie wyświetli 30 kopii tego napisu, każda w osobnej linii.

Zadanie 3.1

Napisz program „powiel.py”, który wczyta od użytkownika pewien napis, a następnie wyświetli 30 kopii tego napisu, każda w osobnej linii.

```
In [*]: text = input("Podaj napis:")
for i in range(1,20):
    print (text)
```

Podaj napis:

```
In [6]: text = input("Podaj napis:")
for i in range(1,20):
    print (text)
```

```
Podaj napis:Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
Napis testowy
```

13) Zadanie 3.2.

Napisz program „pole_tr.py” który obliczy pole trójkąta, pod warunkiem że użytkownik poda wysokość i długość podstawy tego trójkąta. Uwzględnij, że wysokość i długość podstawy mogą być liczbami niecałkowitymi.

```
In [4]: a = input("Podaj długość podstawy trójkąta: ")
h = input("Podaj wysokość trójkąta:")
result = float(a)*float(h)
print(result)
```

```
Podaj długość podstawy trójkąta: 3.4
Podaj wysokość trójkąta:2.3
7.819999999999999
```


14) Zadanie 3.3.

Napisz program "odsetki.py", który obliczy stan konta za N lat, gdzie stan początkowy konta wynosi SPK, a stopa oprocentowania P % rocznie (obowiązuje miesięczna kapitalizacja odsetek). N, SPK i P podaje użytkownik programu.

```
In [9]: N = input("Podaj liczbę lat: ")
        SPK = input("Podaj stan początkowy konta: ")
        P = input("Podaj roczną stopę oprocentowania w procentach: ")
        result = float(SPK)*(1+float(P)/100/12)**(12*float(N))
        print("Stan konta za",N,"lat przy stopie oprocentowania",P+"%", " i początkowym stanie konta", SPK, " :", result)
```

```
Podaj liczbę lat: 2
Podaj stan początkowy konta: 1000
Podaj roczną stopę oprocentowania w procentach: 4
Stan konta za 2 lat przy stopie oprocentowania 4% i początkowym stanie konta 1000 : 1083.1429591590663
```

15) Zadanie 4.1.

Napisz program "parzyste.py", który wczyta od użytkownika liczbę całkowitą i wyświetli informację, czy jest to liczba parzysta, czy nieparzysta.

```
In [5]: liczba = input("Podaj liczbę całkowitą: ")
        if int(liczba) % 2 == 0:
            print("Liczba",liczba,"jest parzysta")
        else:
            print("Liczba",liczba,"jest nieparzysta")
```

```
Podaj liczbę całkowitą: 654
Liczba 654 jest parzysta
```

```
In [1]: liczba = input("Podaj liczbę całkowitą: ")
        if int(liczba) % 2 == 0:
            print("Liczba",liczba,"jest parzysta")
        else:
            print("Liczba",liczba,"jest nieparzysta")
```

```
Podaj liczbę całkowitą: 17
Liczba 17 jest nieparzysta
```

16) Zadanie 4.2.

Napisz program "całkowite.py", który wczyta od użytkownika liczbę i wyświetli informację, czy jest to liczba całkowita, czy niecałkowita.

```
In [6]: from math import floor

        liczba = input("Podaj liczbę: ")
        if float(liczba) - floor(float(liczba)) == 0:
            print("Liczba",liczba,"jest całkowita")
        else:
            print("Liczba",liczba,"nie jest całkowita")
```

```
Podaj liczbę: 234.45
Liczba 234.45 nie jest całkowita
```

```
In [2]: from math import floor

        liczba = input("Podaj liczbę: ")
        if float(liczba) - floor(float(liczba)) == 0:
            print("Liczba",liczba,"jest całkowita")
        else:
            print("Liczba",liczba,"nie jest całkowita")
```

```
Podaj liczbę: 2
Liczba 2 jest całkowita
```

17) Zadanie 4.3.

Napisz program "prk.py", który obliczy wszystkie pierwiastki rzeczywiste równania kwadratowego o postaci $ax^2+bx+c=0$, gdzie a, b i c podaje użytkownik. Program powinien na początku sprawdzić, czy wprowadzone równanie jest rzeczywiście kwadratowe.

Na potrzeby wyliczenia pierwiastków równania zaimportowano funkcję sqrt z biblioteki math.

```
In [6]: from math import sqrt
```

```
In [7]: print("Postać równania kwadratowego: ax2 + bx + c = 0")

a = float(input("Podaj parametr a: "))
b = float(input("Podaj parametr b: "))
c = float(input("Podaj parametr c: "))

delta = b**2 - 4*a*c

if delta < 0:
    print ("Brak pierwiastków rzeczywistych")
elif delta == 0:
    x = -b /( 2*a )
    print("Podwójny pierwiastek równania to x =",x)
else:
    x1 = ( -b - sqrt(delta) )/( 2*a )
    x2 = ( -b + sqrt(delta) )/( 2*a )
    print("Pierwiastki równania to: x1 =",x1,"oraz x2 =",x2)
```

Postać równania kwadratowego: ax2 + bx + c = 0
Podaj parametr a: 3
Podaj parametr b: 5
Podaj parametr c: 2
Pierwiastki równania to: x1 = -1.0 oraz x2 = -0.6666666666666666

```
In [1]: print("Postać równania kwadratowego: ax2 + bx + c = 0")

a = float(input("Podaj parametr a: "))
b = float(input("Podaj parametr b: "))
c = float(input("Podaj parametr c: "))

delta = b**2 - 4*a*c

if delta < 0:
    print ("Brak pierwiastków rzeczywistych")
elif delta == 0:
    x = -b /( 2*a )
    print("Podwójny pierwiastek równania to x =",x)
else:
    x1 = ( -b - sqrt(delta) )/( 2*a )
    x2 = ( -b + sqrt(delta) )/( 2*a )
    print("Pierwiastki równania to: x1 =",x1,"oraz x2 =",x2)
```

Postać równania kwadratowego: ax2 + bx + c = 0
Podaj parametr a: 1
Podaj parametr b: 4
Podaj parametr c: 4
Podwójny pierwiastek równania to x = -2.0

```
In [2]: print("Postać równania kwadratowego: ax2 + bx + c = 0")

a = float(input("Podaj parametr a: "))
b = float(input("Podaj parametr b: "))
c = float(input("Podaj parametr c: "))

delta = b**2 - 4*a*c

if delta < 0:
    print ("Brak pierwiastków rzeczywistych")
elif delta == 0:
    x = -b /( 2*a )
    print("Podwójny pierwiastek równania to x =",x)
else:
    x1 = ( -b - sqrt(delta) )/( 2*a )
    x2 = ( -b + sqrt(delta) )/( 2*a )
    print("Pierwiastki równania to: x1 =",x1,"oraz x2 =",x2)
```

Postać równania kwadratowego: ax2 + bx + c = 0
Podaj parametr a: 23
Podaj parametr b: 3
Podaj parametr c: 2
Brak pierwiastków rzeczywistych

18) Zadanie 5.1.

Napisz program "parzyste2.py", który wczyta od użytkownika liczbę całkowitą i bez użycia instrukcji if wyświetli informację, czy jest to liczba parzysta, czy nieparzysta.

```
In [4]: liczba = int(input("Podaj liczbę całkowitą: "))
print( {True: "Liczba jest parzysta", False: "Liczba jest nieparzysta"}[liczba % 2 == 0] )

Podaj liczbę całkowitą: 25
Liczba jest nieparzysta
```

```
In [5]: liczba = int(input("Podaj liczbę całkowitą: "))
print( {True: "Liczba jest parzysta", False: "Liczba jest nieparzysta"}[liczba % 2 == 0] )

Podaj liczbę całkowitą: 10
Liczba jest parzysta
```

19) Zadanie 5.2.

Napisz program "numer.py", który zamieni wprowadzony przez użytkownika ciąg cyfr na formę tekstową:

a. znaki nie będące cyframi mają być ignorowane

b. konwertujemy cyfry, nie liczby, a zatem:

i. 911 to "dziewięć jeden jeden"

ii. 1100 to "jeden jeden zero zero"

Pierwszym krokiem jest stworzenie słownika zawierającego klucze (poszczególne cyfry) i wartości (odpowiadające cyfrom nazwy).

```
In [11]: dict = {
    0 : "zero", 1 : "jeden", 2 : "dwa", 3 : "trzy", 4 : "cztery", 5 : "pięć",
    6 : "sześć", 7 : "siedem", 8 : "osiem", 9 : "dziewięć"
}
```

Następnie należy pobrać od użytkownika ciąg cyfr i przekształcić je na słowa:

```
In [12]: liczba = input("Podaj ciąg cyfr. Znaki inne niż cyfry będą ignorowane: ")

Podaj ciąg cyfr. Znaki inne niż cyfry będą ignorowane: krfmju347yewgsdbh cxbw2
```

W celu sprawdzenia, czy dany znak jest cyfrą, sprawdza się jego kod ASCII. Powinien należeć do zakresu < 48, 57 >

```
In [13]: wynik = ""
for znak in liczba:
    if ( 48 <= ord(znak) <= 57 ):
        wynik += dict[int(znak)] + " "
print(wynik)

trzy cztery siedem dwa
```