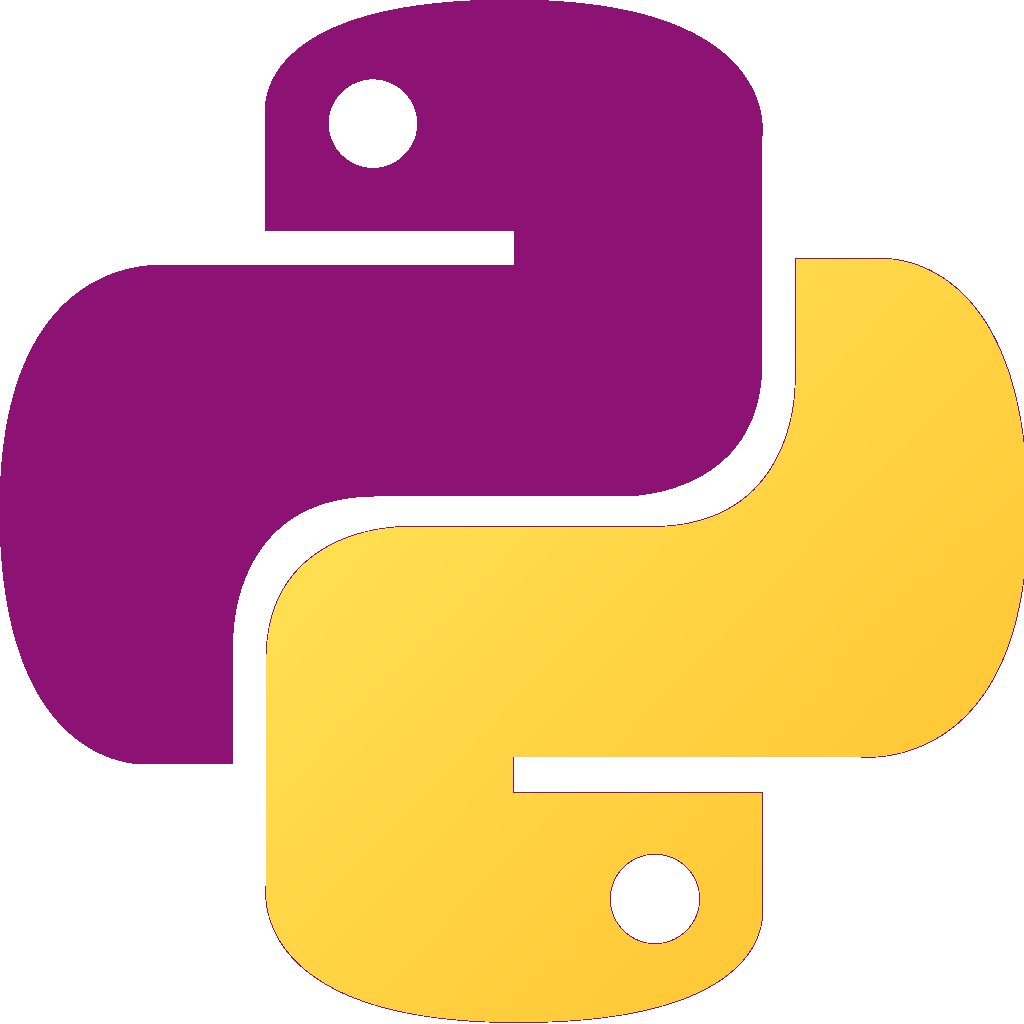
**Učebnice Python pro středně pokročilé**

**k videím na YT Hackni svou budoucnost**

**David Šetek**

**Odkaz na učebnici**

[**bit.ly/hackni-python-2**](https://bit.ly/hackni-python-2)



**Seznam videí na kanále YouTube**

**David Šetek - Hackni svou budoucnost**

<https://www.youtube.com/playlist?list=PLQ8x_VWW6AkuPXuUJ-lGH2aoL4vuoQg62>

**1. Python - Úvodní video o kurzu Python pro středně pokročilé**

**Video:** [**https://youtu.be/vtKETizdfX0**](https://youtu.be/vtKETizdfX0)

Odkaz na učebnici:

[**bit.ly/hackni-python-2**](https://bit.ly/hackni-python-2)

**2. Python - Pokročilý automat na kávu (opakování základů)**

**Video:** [**https://youtu.be/ZjQWInAsOyY**](https://youtu.be/ZjQWInAsOyY)

Zdrojová data ke zkopírování

MENU = {

"espresso": {

"ingredients": {

"water": 50,

"milk": 0,

"coffee": 18,

},

"cost": 40,

},

"latte": {

"ingredients": {

"water": 200,

"milk": 150,

"coffee": 24,

},

"cost": 50,

},

"cappuccino": {

"ingredients": {

"water": 250,

"milk": 100,

"coffee": 24,

},

"cost": 60,

}

}

resources = {

"water": 400,

"milk": 300,

"coffee": 150,

}

Výsledný kód

from source\_data import MENU

from source\_data import resources

# ===Základní nastavení===

espresso\_price = MENU["espresso"]["cost"]

latte\_price = MENU["latte"]["cost"]

cappuccino\_price = MENU["cappuccino"]["cost"]

# ===Funkce===

def report(data):

print(f"Voda: {data['water']}")

print(f"Mléko: {data['milk']}")

print(f"Káva: {data['coffee']}")

def coins():

print("Prosím vložte mince 1, 2, 5, 10, 20, 50")

kc1 = int(input("Kolik 1 Kč chcete vložit?: ")) \* 1

kc2 = int(input("Kolik 2 kč chcete vložit?: ")) \* 2

kc5 = int(input("Kolik 5 kč chcete vložit?: ")) \* 5

kc10 = int(input("Kolik 10 kč chcete vložit?: ")) \* 10

kc20 = int(input("Kolik 20 kč chcete vložit?: ")) \* 20

kc50 = int(input("Kolik 50 kč chcete vložit?: ")) \* 50

suma = kc1 + kc2 + kc5 + kc10 + kc20 + kc50

print(f"Celkem jste vložili: {suma} Kč")

return suma

def calculate\_change(user\_sum\_coins, price):

refund = user\_sum\_coins - price

if refund >= 0:

print("Nápoj se připravuje.")

if refund > 0:

print(f"Zde jsou peníze zpět: {refund} Kč")

else:

print(f"Nevhodili jste dostatek peněz. Ještě je zapotřebí vložit {price - user\_sum\_coins} Kč")

def fill\_in\_ingredience():

return resources

def consumption\_ingredience(name\_of\_drink, ingredience):

ingredience["water"] = ingredience["water"] - MENU[name\_of\_drink]["ingredients"]["water"]

ingredience["milk"] = ingredience["milk"] - MENU[name\_of\_drink]["ingredients"]["milk"]

ingredience["coffee"] = ingredience["coffee"] - MENU[name\_of\_drink]["ingredients"]["coffee"]

print(f"Zbylé ingredience: {ingredience}")

def calculate\_ingredients(drink\_name):

if drink\_name == "espresso":

consumption\_ingredience(drink\_name, rest\_of\_ingredience)

elif drink\_name == "latte":

consumption\_ingredience(drink\_name, rest\_of\_ingredience)

elif drink\_name == "cappuccino":

consumption\_ingredience(drink\_name, rest\_of\_ingredience)

def ingredience\_checker(in\_water, in\_milk, in\_coffee):

if in\_water < 0:

print("Nemáme dostatek ingrediencí na tento nápoj")

return False

elif in\_milk < 0:

print("Nemáme dostatek ingrediencí na tento nápoj")

return False

elif in\_coffee < 0:

print("Nemáme dostatek ingrediencí na tento nápoj")

return False

else:

print("Na váš nápoj máme dostatek ingrediencí.")

return True

# ===Kód automatu===

# Načítáme původní množství ingrediencí

rest\_of\_ingredience = fill\_in\_ingredience()

lets\_continue = True

while(lets\_continue):

# Volba uživatele - jaký chce nápoj

user\_choice = input("Co byste si dal/a? (espresso/latte/cappuccino): ")

# Vypočítá kolik zbývá ingrediencí

calculate\_ingredients(user\_choice)

# Kontrola, zda máme dostatek ingrediencí

if user\_choice != "report":

lets\_continue = ingredience\_checker(rest\_of\_ingredience["water"], rest\_of\_ingredience["milk"], rest\_of\_ingredience["coffee"])

# Má kód dále pokračovat?

if lets\_continue == False:

break

# Kontrolní report

if user\_choice == "report":

report(rest\_of\_ingredience)

# Hlavní kód automatu

if user\_choice == "espresso":

sum = coins()

print(f"Cena espressa je: {espresso\_price} Kč")

calculate\_change(sum, espresso\_price)

elif user\_choice == "latte":

sum = coins()

print(f"Cena espressa je: {latte\_price} Kč")

calculate\_change(sum, latte\_price)

elif user\_choice == "cappuccino":

sum = coins()

print(f"Cena espressa je: {cappuccino\_price} Kč")

calculate\_change(sum, cappuccino\_price)

**3. Python - Úvod do objektově orientovaného programování**

**Video:** [**https://youtu.be/211mnm\_gSws**](https://youtu.be/211mnm_gSws)

**4. Python - Tvorba první classy a objektů v OOP**

**Video:** [**https://youtu.be/jSAbp0OL7SU**](https://youtu.be/jSAbp0OL7SU)

class Robot:

pass

robot\_1 = Robot()

robot\_1.bateire = 24

robot\_1.delka\_rukou = 0.6

robot\_2 = Robot()

robot\_2.bateire = 48

robot\_2.delka\_rukou = 0.5

print(f"Výdrž baterie: {robot\_1.bateire}")

print(f"Délka rukou: {robot\_1.delka\_rukou}")

print(f"Výdrž baterie: {robot\_2.bateire}")

print(f"Délka rukou: {robot\_2.delka\_rukou}")

**5. Python - Používáme constructor u robota**

**Video:** [**https://youtu.be/jB758Dw-eN4**](https://youtu.be/jB758Dw-eN4)

class Robot:

# constructor

def \_\_init\_\_(self, baterie, delka\_rukou):

self.baterie = baterie

self.delka\_rukou = delka\_rukou

# Tvoříme objekty podle classy

robot\_1 = Robot(24, 0.6)

robot\_2 = Robot(48, 0.5)

robot\_3 = Robot(50, 0.6)

robot\_4 = Robot(38, 0.4)

print(robot\_1.baterie)

print(robot\_1.delka\_rukou)

print(robot\_2.baterie)

print(robot\_2.delka\_rukou)

print(robot\_3.baterie)

print(robot\_3.delka\_rukou)

print(robot\_4.baterie)

print(robot\_4.delka\_rukou)

**6. Python - Dny do opravy robota (defaultní hodnoty v OOP)**

**Video:** [**https://youtu.be/f-Fe9yAOL\_E**](https://youtu.be/f-Fe9yAOL_E)

class Robot:

# constructor

def \_\_init\_\_(self, baterie, delka\_rukou):

self.baterie = baterie

self.delka\_rukou = delka\_rukou

self.dny\_do\_opravy = 365

# Tvoříme objekty podle classy

robot\_1 = Robot(24, 0.6)

robot\_2 = Robot(48, 0.5)

robot\_3 = Robot(50, 0.6)

robot\_4 = Robot(38, 0.4)

print(robot\_1.baterie)

print(robot\_1.delka\_rukou)

print(robot\_1.dny\_do\_opravy)

print(robot\_2.baterie)

print(robot\_2.delka\_rukou)

print(robot\_2.dny\_do\_opravy)

print(robot\_3.baterie)

print(robot\_3.delka\_rukou)

print(robot\_3.dny\_do\_opravy)

print(robot\_4.baterie)

print(robot\_4.delka\_rukou)

print(robot\_4.dny\_do\_opravy)

**7. Python - Robot dělá krok vpřed a vzad (metody v OOP)**

**Video:** [**https://youtu.be/OadfGruLzUw**](https://youtu.be/OadfGruLzUw)

class Robot:

# constructor

def \_\_init\_\_(self, baterie, delka\_rukou):

self.baterie = baterie

self.delka\_rukou = delka\_rukou

self.ukony\_do\_kontroly = 1000

def krok\_vpred(self):

print("Robot udělal krok vpřed")

self.ukony\_do\_kontroly -= 1

def krok\_vzad(self):

print("Robot udělal krok vzad")

self.ukony\_do\_kontroly -= 1

# Tvoříme objekty podle classy

robot\_1 = Robot(24, 0.6)

robot\_2 = Robot(48, 0.5)

robot\_3 = Robot(50, 0.6)

robot\_4 = Robot(38, 0.4)

print(robot\_1.baterie)

print(robot\_1.delka\_rukou)

print(robot\_1.ukony\_do\_kontroly)

robot\_1.krok\_vpred()

robot\_1.krok\_vzad()

robot\_1.krok\_vpred()

robot\_1.krok\_vzad()

robot\_1.krok\_vpred()

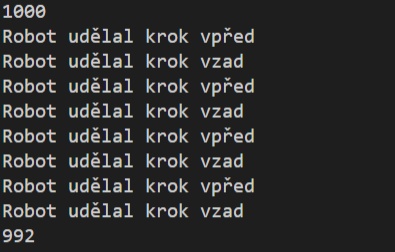
robot\_1.krok\_vzad()

robot\_1.krok\_vpred()

robot\_1.krok\_vzad()

print(robot\_1.ukony\_do\_kontroly)

**Terminál:**



**8. Python - Log robota pro servisáka (použití atributu v metodě)**

**Video:** [**https://youtu.be/SIkU-Ya7I2s**](https://youtu.be/SIkU-Ya7I2s)

Přidali jsme žluté části kódu

class Robot:

# constructor

def \_\_init\_\_(self, baterie, delka\_rukou):

self.baterie = baterie

self.delka\_rukou = delka\_rukou

self.ukony\_do\_kontroly = 1000

def krok\_vpred(self):

print("Robot udělal krok vpřed")

self.ukony\_do\_kontroly -= 1

print(f"Úkonů do kontroly: {self.ukony\_do\_kontroly}")

def krok\_vzad(self):

print("Robot udělal krok vzad")

self.ukony\_do\_kontroly -= 1

print(f"Ukonů do kontroly: {self.ukony\_do\_kontroly}")

# Tvoříme objekty podle classy

robot\_1 = Robot(24, 0.6)

robot\_2 = Robot(48, 0.5)

robot\_3 = Robot(50, 0.6)

robot\_4 = Robot(38, 0.4)

robot\_1.krok\_vpred()

robot\_1.krok\_vzad()

robot\_1.krok\_vpred()

robot\_1.krok\_vzad()

**9. Python - Tvoříme kvíz (1. část)**

**Video:** [**https://youtu.be/ghEfPv9kSC0**](https://youtu.be/ghEfPv9kSC0)

**Data ke zkopírování**

question\_data = [

{"text": "Operační systém Linux byl založen Linusem Torvaldem", "answer": "True"},

{"text": "HTML jazyk je také nazýván značkovacím jazykem", "answer": "True"},

{"text": "JavaScript patří mezi frontendové jazyky", "answer": "True"},

{"text": "CSS je programovací jazyk", "answer": "False"}

]

**question\_model.py**

class Question:

def \_\_init\_\_(self, question\_text, question\_answer):

self.text = question\_text

self.answer = question\_answer

# q\_1 = Question("Python vznikl v roce 1991", "True")

# q\_2 = Question("Operační systém Linux byl založen Linusem Torvaldem", "True")

**data.py**

question\_data = [

{"text": "Operační systém Linux byl založen Linusem Torvaldem", "answer": "True"},

{"text": "HTML jazyk je také nazýván značkovacím jazykem", "answer": "True"},

{"text": "JavaScript patří mezi frontendové jazyky", "answer": "True"},

{"text": "CSS je programovací jazyk", "answer": "False"}

]

**10. Python - Tvoříme kvíz (2. část)**

**Video:** [**https://youtu.be/zacm1fFgHfY**](https://youtu.be/zacm1fFgHfY)

from question\_model import Question

from data import question\_data

question\_list = []

for one\_question in question\_data:

question\_t = one\_question["text"]

question\_a = one\_question["answer"]

new\_question = Question(question\_t, question\_a)

question\_list.append(new\_question)

print(question\_list)

**11. Python - Tvoříme kvíz (3. část)**

**Video:** [**https://youtu.be/24uEJ1IcCYQ**](https://youtu.be/24uEJ1IcCYQ)

**quiz\_brain.py**

class QuizBrain:

def \_\_init\_\_(self, q\_list):

self.question\_number = 0

self.question\_li = q\_list

def next\_question(self):

current\_question = self.question\_li[self.question\_number]

self.question\_number += 1

input(f"Otázka č. {self.question\_number}: {current\_question.text} (True/False): ")

**main.py**

from question\_model import Question

from data import question\_data

from quiz\_brain import QuizBrain

question\_list = []

for one\_question in question\_data:

question\_t = one\_question["text"]

question\_a = one\_question["answer"]

new\_question = Question(question\_t, question\_a)

question\_list.append(new\_question)

# print(question\_list[0].text)

# print(question\_list[0].answer)

quiz = QuizBrain(question\_list)

**12. Python - Tvoříme kvíz (4. část)**

**Video:** [**https://youtu.be/onv0ElYnKM4**](https://youtu.be/onv0ElYnKM4)

**quiz\_brain.py**

def has\_questions(self):

if self.question\_number < len(self.question\_li):

return True

else:

return False

**main.py**

while quiz.has\_questions() == True:

quiz.next\_question()

**13. Python - Tvoříme kvíz (5. část)**

**Video:** [**https://youtu.be/s1mFVAWppK4**](https://youtu.be/s1mFVAWppK4)

**quiz\_brain.py**

def next\_question(self):

current\_question = self.question\_li[self.question\_number]

self.question\_number += 1

user\_answer = input(f"Otázka č. {self.question\_number}: {current\_question.text} (True/False): ")

self.check\_answer(user\_answer, current\_question.answer)

def check\_answer(self, u\_answer, correct\_answer):

if u\_answer.lower() == correct\_answer.lower():

print("Uhádli jste!")

else:

print("Špatná odpověď")

print(f"Správná odpověď je: {correct\_answer}.")

**14. Python - Tvoříme kvíz (6. část)**

**Video:** [**https://youtu.be/v-fic3\_bBRs**](https://youtu.be/v-fic3_bBRs)

**quiz\_brain.py**

def \_\_init\_\_(self, q\_list):

self.question\_number = 0

self.score = 0

self.question\_li = q\_list

**quiz\_brain.py**

def check\_answer(self, u\_answer, correct\_answer):

if u\_answer.lower() == correct\_answer.lower():

print("Uhádli jste!")

self.score += 1

else:

print("Špatná odpověď")

print(f"Správná odpověď je: {correct\_answer}.")

print(f"Vaše skóre je: {self.score} / {self.question\_number}")

**15. Python - Celý kód na GitHubu (jak kód stáhnout)**

**Video:** [**https://youtu.be/c2juIFs6BJE**](https://youtu.be/c2juIFs6BJE)

Kód ke stažení najdete zde:

<https://github.com/DavidSetek/python-quiz.git>

**16. Python - Vkládáme otázky z Open database a ukazujeme si výhody OOP**

**Video:** [**https://youtu.be/a8OgoVjB9Ag**](https://youtu.be/a8OgoVjB9Ag)

<https://opentdb.com/>

Data z Open database - vy si vygenerujte svoje

question\_data = [

{

"category": "Science: Computers",

"type": "boolean",

"difficulty": "medium",

"question": "The HTML5 standard was published in 2014.",

"correct\_answer": "True",

"incorrect\_answers": [

"False"

]

},

{

"category": "Science: Computers",

"type": "boolean",

"difficulty": "medium",

"question": "The very first recorded computer &quot;bug&quot; was a moth found inside a Harvard Mark II computer.",

"correct\_answer": "True",

"incorrect\_answers": [

"False"

]

},

{

"category": "Science: Computers",

"type": "boolean",

"difficulty": "easy",

"question": "The programming language &quot;Python&quot; is based off a modified version of &quot;JavaScript&quot;.",

"correct\_answer": "False",

"incorrect\_answers": [

"True"

]

}

]

main.py - změníme pouze tyto dva žluté názvy

for one\_question in question\_data:

question\_t = one\_question["question"]

question\_a = one\_question["correct\_answer"]

new\_question = Question(question\_t, question\_a)

question\_list.append(new\_question)

**17. Python - Jak na importy modulů a souborů (4 možnosti importu)**

**Video:** [**https://youtu.be/G2bKXkR8VFk**](https://youtu.be/G2bKXkR8VFk)

# 1. možnost - zdlouhavější zápis

import data

print(data.my\_data)

# 2. možnost - doporučovaná

from data import my\_data

print(my\_data)

# 3. možnost - moc se nepoužívá, matoucí

from data import \*

# 4. možnost - alias (jiný název)

import data as d

print(d.my\_data)

**18. Python - Grafické prostředí Turtle Graphics**

**Video:** [**https://youtu.be/exKPWi9cATE**](https://youtu.be/exKPWi9cATE)

Dokumentace k modulu **turtle**

<https://docs.python.org/3/library/turtle.html>

# Turtle graphics

from turtle import Turtle, Screen

tommy = Turtle()

my\_screen = Screen()

# print(f"šířka: {my\_screen.canvwidth}")

# print(f"výška: {my\_screen.canvheight}")

my\_screen.exitonclick()

**19. Python - Turtle Graphics - měníme tvar želvy, barvu a barvu pozadí**

**Video:** [**https://youtu.be/03\_-EycPUgM**](https://youtu.be/03_-EycPUgM)

Seznam barev:

<https://www.tcl.tk/man/tcl8.4/TkCmd/colors.html>

# Turtle graphics

from turtle import Turtle, Screen

tommy = Turtle()

tommy.shape("turtle")

tommy.color("green")

my\_screen = Screen()

my\_screen.exitonclick()

**20. Python - Turtle Graphics - rozhýbeme želvu (forward, backward, right, left)**

**Video:** [**https://youtu.be/fzw0\_I5cVnY**](https://youtu.be/fzw0_I5cVnY)

# Turtle graphics

from turtle import Turtle, Screen

tommy = Turtle()

tommy.shape("turtle")

tommy.forward(50)

tommy.right(90)

tommy.forward(50)

my\_screen = Screen()

my\_screen.exitonclick()

**21. Python - Nakreslete čtverec (procvičování)**

**Video:** [**https://youtu.be/XwLEQMSrVWs**](https://youtu.be/XwLEQMSrVWs)

Založení zcela od začátku + zadání úkolu

# Turtle graphics

from turtle import Turtle, Screen

tommy = Turtle()

tommy.shape("turtle")

# Vaším úkolem je, aby želva nakreslila čtverec. Zkuste nejdříve kód napsat klasicky v příkazech za sebou a poté použít cyklus.

my\_screen = Screen()

my\_screen.exitonclick()

**Řešení**

from turtle import Turtle, Screen

tommy = Turtle()

tommy.shape("turtle")

# Vaším úkolem je, aby želva nakreslila čtverec. Zkuste nejdříve kód napsat klasicky v příkazech za sebou a poté použít cyklus.

# 1. možnost

# tommy.forward(100)

# tommy.right(90)

# tommy.forward(100)

# tommy.right(90)

# tommy.forward(100)

# tommy.right(90)

# tommy.forward(100)

# tommy.right(90)

# 2. možnost

for \_ in range(0, 4):

tommy.forward(100)

tommy.right(90)

my\_screen = Screen()

my\_screen.exitonclick()

**22. Python - Nakreslete čárkovanou čáru (procvičování)**

**Video:** [**https://youtu.be/ZGjSNEGUpUA**](https://youtu.be/ZGjSNEGUpUA)

Dokumentace Turtle Graphics

<https://docs.python.org/3/library/turtle.html>

**Zadání**

from turtle import Turtle, Screen

tommy = Turtle()

tommy.shape("turtle")

# Vaším úkolem je, aby želva nakreslila čárkovanou čáru a v libovolné délce

my\_screen = Screen()

my\_screen.exitonclick()

**Řešení**

from turtle import Turtle, Screen

tommy = Turtle()

tommy.shape("turtle")

# Vaším úkolem je, aby želva nakreslila čárkovanou čáru a v libovolné délce

for \_ in range(10):

tommy.pendown()

tommy.forward(20)

tommy.penup()

tommy.forward(20)

my\_screen = Screen()

my\_screen.exitonclick()

**23. Python - Tvoříme obrazce vždy s jedním úhlem navíc (procvičování)**

**Video:** [**https://youtu.be/JGKHErHz6lo**](https://youtu.be/JGKHErHz6lo)

<https://www.tcl.tk/man/tcl8.4/TkCmd/colors.html>

**Řešení**

from turtle import Turtle, Screen

import random

tommy = Turtle()

tommy.shape("turtle")

tommy.pensize(2)

colors = ["azure2", "brown4", "chartreuse", "coral1", "cornsilk2", "DarkMagenta", "DarkSeaGreen3", "DeepSkyBlue4"]

moves = 3

while moves != 9:

random\_color = random.choice(colors)

tommy.pencolor(random\_color)

for \_ in range(moves):

tommy.forward(100)

tommy.right(360/moves)

moves += 1

# for \_ in range(3):

# tommy.forward(100)

# tommy.right(120) # 360 : 3

# for \_ in range(4):

# tommy.forward(100)

# tommy.right(90) # 360 : 4

# for \_ in range(5):

# tommy.forward(100)

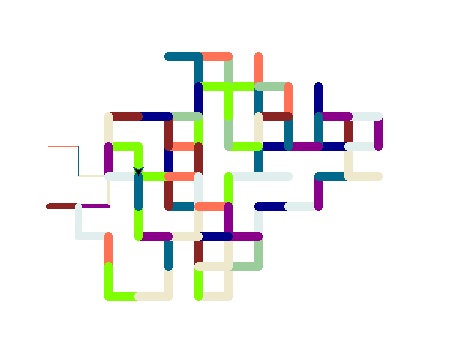
# tommy.right(72) # 360 : 5

my\_screen = Screen()

my\_screen.exitonclick()

**24. Python - Náhodný pohyb (procvičování)**

**Video:** [**https://youtu.be/ZMYHyfL0xOo**](https://youtu.be/ZMYHyfL0xOo)



from turtle import Turtle, Screen

import random

tommy = Turtle()

tommy.shape("turtle")

colors = ["azure2", "brown4", "chartreuse", "coral1", "cornsilk2", "DarkMagenta", "DarkSeaGreen3", "DeepSkyBlue4", "blue4"]

rotation = [0, 90, 180, 270]

speed = 1

for number in range(200):

# Náhodný výběr barvy

random\_color = random.choice(colors)

tommy.pencolor(random\_color)

# Tloušťka čáry se zvyšuje

if number <= 10:

tommy.pensize(number)

# Pohyb a náhodné otočení

tommy.forward(30)

tommy.right(random.choice(rotation))

# Zvyšujeme rychlost

tommy.speed(speed)

speed += 1

my\_screen = Screen()

my\_screen.exitonclick()

**25. Python - Datový typ Tuple, co to je a jak funguje**

**Video:** [**https://youtu.be/MBup-G5aPoY**](https://youtu.be/MBup-G5aPoY)

# String

# Integer

# Float

# Boolean

# List

# Dictionary

# Tuple

my\_tuple = (1, 5, 8)

print(my\_tuple[0])

print(my\_tuple[1])

print(my\_tuple[2])

# Vyhodí chybu

# my\_tuple[0] = 12

# Tuple změníme na list

tuple\_to\_list = list(my\_tuple)

print(tuple\_to\_list)

tuple\_to\_list[0] = 12

print(tuple\_to\_list)

**26. Python - Tuple v praxi, vylepšujeme random-walk náhodným generováním barvy**

**Video:** [**https://youtu.be/zNCuMGbpuQA**](https://youtu.be/zNCuMGbpuQA)

from turtle import Turtle, Screen

import random

import turtle

# Změna barevného módu

turtle.colormode(255)

tommy = Turtle()

tommy.shape("turtle")

def random\_color():

r = random.randint(0, 255)

g = random.randint(0, 255)

b = random.randint(0, 255)

random\_color = (r, g, b)

return random\_color

# colors = ["azure2", "brown4", "chartreuse", "coral1", "cornsilk2", "DarkMagenta", "DarkSeaGreen3", "DeepSkyBlue4", "blue4"]

rotation = [0, 90, 180, 270]

speed = 1

for number in range(200):

# Náhodný výběr barvy

tommy.pencolor(random\_color())

# Tloušťka čáry se zvyšuje

if number <= 10:

tommy.pensize(number)

# Pohyb a náhodné otočení

tommy.forward(30)

tommy.right(random.choice(rotation))

# Zvyšujeme rychlost

tommy.speed(speed)

speed += 1

my\_screen = Screen()

my\_screen.exitonclick()

**27. Python - Spirograf v pythonu (procvičování)**

**Video:** [**https://youtu.be/uLoW9jSBJgo**](https://youtu.be/uLoW9jSBJgo)

Začátek souboru

from turtle import Turtle, Screen

tommy = Turtle()

tommy.shape("turtle")

my\_screen = Screen()

my\_screen.exitonclick()

Tvorba spirografu

# Importy

from turtle import Turtle, Screen

import random

import turtle

# Změna barevného módu

turtle.colormode(255)

# Generování a základní nastavení objektu

tommy = Turtle()

tommy.shape("turtle")

tommy.speed(20)

# Funkce na generování barvy

def random\_color():

r = random.randint(0, 255)

g = random.randint(0, 255)

b = random.randint(0, 255)

color = (r, g, b)

return color

for number in range(36):

tommy.pencolor(random\_color())

tommy.circle(80)

tommy.left(10)

my\_screen = Screen()

my\_screen.exitonclick()

**28. Python - Vylepšujeme Spirograf**

**Video:** [**https://youtu.be/cHJQ3fj7tps**](https://youtu.be/cHJQ3fj7tps)

# Importy

from turtle import Turtle, Screen

import random

import turtle

# Změna barevného módu

turtle.colormode(255)

# Generování a základní nastavení objektu

tommy = Turtle()

tommy.shape("turtle")

tommy.speed(20)

# Funkce na generování barvy

def random\_color():

r = random.randint(0, 255)

g = random.randint(0, 255)

b = random.randint(0, 255)

color = (r, g, b)

return color

def spirograph(gap):

for number in range(int(360/gap)):

tommy.pencolor(random\_color())

tommy.circle(80)

tommy.left(gap)

spirograph(1)

my\_screen = Screen()

my\_screen.exitonclick()

**29. Python - Tvoříme diagram (procvičování)**

**Video:** [**https://youtu.be/KOGHv27lveI**](https://youtu.be/KOGHv27lveI)

from turtle import Turtle, Screen

colors = ["violet", "yellow", "red", "green", "blue", "pink"]

arrow = Turtle("arrow")

for x in range(100):

arrow.pencolor(colors[x%6])

arrow.forward(x)

arrow.left(60)

screen = Screen()

screen.exitonclick()

**30. Python - Vyplňujeme objekt v Turtle graphics**

**Video:** [**https://youtu.be/UolW7O-akrU**](https://youtu.be/UolW7O-akrU)

from turtle import Turtle, Screen

arrow = Turtle("arrow")

arrow.pencolor("red")

arrow.pen(fillcolor="red")

arrow.begin\_fill()

for \_ in range(4):

arrow.forward(80)

arrow.left(90)

arrow.end\_fill()

screen = Screen()

screen.exitonclick()

**31. Python - Kruh v kruhu pomocí Turtle Graphics**

**Video:** [**https://youtu.be/a8Z80XwfuX4**](https://youtu.be/a8Z80XwfuX4)

from turtle import Turtle, Screen

arrow1 = Turtle("arrow")

arrow2 = Turtle("arrow")

arrow1.color("red")

arrow2.color("green")

arrow1.pensize(2)

arrow2.pensize(2)

arrow1.circle(20)

for i in range(30, 100, 10):

arrow2.circle(i)

screen = Screen()

screen.exitonclick()

**32. Python - Úvodní video ke Snake game**

**Video:** [**https://youtu.be/v\_GzR6VPmmI**](https://youtu.be/v_GzR6VPmmI)

**33. Python - Základní nastavení plátna a co je to tracer a update**

**Video:** [**https://youtu.be/8ot9jCZTXKs**](https://youtu.be/8ot9jCZTXKs)

main.py

from turtle import Turtle, Screen

screen = Screen()

screen.bgcolor("green")

screen.title("Vítejte v Hadí hře")

screen.setup(width=600, height=600)

screen.tracer(False)

screen.exitonclick()

tracer.py

# Testovací soubor

from turtle import Turtle, Screen

import time

screen = Screen()

screen.bgcolor("green")

screen.title("Vítejte v Hadí hře")

screen.setup(width=600, height=600)

screen.tracer(False)

square1 = Turtle("square")

square1.penup()

square1.goto(0, 0)

square2 = Turtle("square")

square2.penup()

square2.goto(-20, 0)

for \_ in range(80):

square1.forward(10)

square2.forward(10)

time.sleep(0.1)

screen.update()

screen.exitonclick()

**34. Python - Tvoříme hadí hlavu a začínáme řešit pohyb**

**Video:** [**https://youtu.be/pNfF-H9qlFw**](https://youtu.be/pNfF-H9qlFw)

main.py

screen.tracer(False)

# Hadí hlava

head = Turtle("square")

head.color("black")

head.speed(0)

head.penup()

head.goto(0, 0)

head.direction = "up"

def move():

if head.direction == "up":

y = head.ycor()

head.sety(y + 20)

while True:

move()

time.sleep(0.1)

screen.update()

screen.exitonclick()

**35. Python - Další směry pohybu hadí hlavy (procvičování)**

**Video:** [**https://youtu.be/rsqjFzSCH08**](https://youtu.be/rsqjFzSCH08)

main.py

def move():

if head.direction == "up":

y = head.ycor()

head.sety(y + 20)

if head.direction == "down":

y = head.ycor()

head.sety(y - 20)

if head.direction == "left":

x = head.xcor()

head.setx(x - 20)

if head.direction == "right":

x = head.xcor()

head.setx(x + 20)

**36. Python - Pohybujeme hlavou hada do všech stran stisknutím kláves (+ procvičování)**

**Video:** [**https://youtu.be/4cnIVzVg9AE**](https://youtu.be/4cnIVzVg9AE)

main.py

def move\_up():

head.direction = "up"

def move\_down():

head.direction = "down"

def move\_left():

head.direction = "left"

def move\_right():

head.direction = "right"

# Kliknutí na klávesy

screen.listen()

screen.onkeypress(move\_up, "w")

screen.onkeypress(move\_down, "s")

screen.onkeypress(move\_left, "a")

screen.onkeypress(move\_right, "d")

events.py

from turtle import Turtle, Screen

screen = Screen()

tommy = Turtle("turtle")

def move\_forward():

tommy.forward(20)

# Stisknutí klávesy

screen.listen()

screen.onkeypress(move\_forward, "w")

screen.exitonclick()

**37. Python - Potrava pro hada, kolize a posun potravy na náhodnou souřadnici**

**Video:** [**https://youtu.be/BIpBUHSgP64**](https://youtu.be/BIpBUHSgP64)

# Hadí hlava a jablko

head = Turtle("square")

head.color("black")

head.speed(0)

head.penup()

head.goto(0, 0)

head.direction = "stop"

apple = Turtle("circle")

apple.color("red")

apple.penup()

apple.goto(100, 100)

from turtle import Turtle, Screen

import time

import random

while True:

screen.update()

if head.distance(apple) < 20:

x = random.randint(-290, 290)

y = random.randint(-290, 290)

apple.goto(x, y)

move()

time.sleep(0.1)

**38. Python - Tvoříme tělo hada (1. část)**

**Video:** [**https://youtu.be/aeDgKeskVSc**](https://youtu.be/aeDgKeskVSc)

apple = Turtle("circle")

apple.color("red")

apple.penup()

apple.goto(100, 100)

body\_parts = []

# Hlavní cyklus

while True:

if head.distance(apple) < 20:

x = random.randint(-290, 290)

y = random.randint(-290, 290)

apple.goto(x, y)

# Přidání části těla

new\_body\_part = Turtle("square")

new\_body\_part.speed(0)

new\_body\_part.color("grey")

new\_body\_part.penup()

body\_parts.append(new\_body\_part)

if len(body\_parts) > 0:

x = head.xcor()

y = head.ycor()

body\_parts[0].goto(x,y)

move()

time.sleep(0.1)

screen.update()

**39. Python - Tvoříme tělo hada (2. část)**

**Video:** [**https://youtu.be/KuBGrOzXxtI**](https://youtu.be/KuBGrOzXxtI)

cycle.py

parts = ["jedna", "dva", "tři", "čtyři", "pět"]

for index in range(len(parts) - 1, 0, -1):

print(parts[index])

main.py

for index in range(len(body\_parts) - 1, 0, -1):

x = body\_parts[index - 1].xcor()

y = body\_parts[index - 1].ycor()

body\_parts[index].goto(x, y)

if len(body\_parts) > 0:

x = head.xcor()

y = head.ycor()

body\_parts[0].goto(x,y)

**40. Python - Kolize s okrajem plátna**

**Video:** [**https://youtu.be/iYmihuW-1Os**](https://youtu.be/iYmihuW-1Os)

# Hlavní cyklus

while True:

screen.update()

# Kontrola kolize s hranou obrazovky

if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor() < - 290:

time.sleep(2)

head.goto(0, 0)

head.direction = "stop"

# Skryjeme části těla

for one\_body\_part in body\_parts:

one\_body\_part.goto(1500, 1500)

# Vyprázdníme list s částmi těla (šedé čtverečky)

body\_parts.clear()

**41. Python - Kolize hlavy s tělem**

**Video:** [**https://youtu.be/4qGA3WFoLwI**](https://youtu.be/4qGA3WFoLwI)

move()

# Hlava narazila do těla

for one\_body\_part in body\_parts:

if one\_body\_part.distance(head) < 20:

time.sleep(2)

head.goto(0, 0)

head.direction = "stop"

# Skryjeme části těla

for one\_body\_part in body\_parts:

one\_body\_part.goto(1500, 1500)

# Vyprázdníme list s částmi těla (šedé čtverečky)

body\_parts.clear()

time.sleep(0.1)

**42. Python - Upravujeme směr pohybu hlavy (+ procvičování)**

**Video:** [**https://youtu.be/s96WVh03QyY**](https://youtu.be/s96WVh03QyY)

def move\_up():

if head.direction != "down":

head.direction = "up"

def move\_down():

if head.direction != "up":

head.direction = "down"

def move\_left():

if head.direction != "right":

head.direction = "left"

def move\_right():

if head.direction != "left":

head.direction = "right"

**43. Python - Přidáváme skóre a nejvyšší dosažené skóre**

**Video:** [**https://youtu.be/PNGAU4sPLPs**](https://youtu.be/PNGAU4sPLPs)

apple = Turtle("circle")

apple.color("red")

apple.penup()

apple.goto(100, 100)

score\_sign = Turtle("square")

score\_sign.speed(0)

score\_sign.color("white")

score\_sign.penup()

score\_sign.hideturtle()

score\_sign.goto(0, 265)

score\_sign.write("Skóre: 0 Nejvyšší skóre: 0", align="center", font=("Arial", 18))

from turtle import Turtle, Screen

import time

import random

# Proměnné

score = 0

highest\_score = 0

# Přidání části těla

new\_body\_part = Turtle("square")

new\_body\_part.speed(0)

new\_body\_part.color("grey")

new\_body\_part.penup()

body\_parts.append(new\_body\_part)

# Zvýšení skóre

# score = score + 10

score += 10

if score > highest\_score:

highest\_score = score

score\_sign.clear()

score\_sign.write(f"Skóre: {score} Nejvyšší skóre: {highest\_score}", align="center", font=("Arial", 18))

Resetování skóre, když dojde ke kolizi s hranou obrazovky

# Skryjeme části těla

for one\_body\_part in body\_parts:

one\_body\_part.goto(1500, 1500)

# Vyprázdníme list s částmi těla (šedé čtverečky)

body\_parts.clear()

# Resetování skóre

score = 0

score\_sign.clear()

score\_sign.write(f"Skóre: {score} Nejvyšší skóre: {highest\_score}", align="center", font=("Arial", 18))

Resetování skóre, když hlava koliduje se svým tělem - POZOR NA ODSAZENÍ!!!

# Skryjeme části těla

for one\_body\_part in body\_parts:

one\_body\_part.goto(1500, 1500)

# Vyprázdníme list s částmi těla (šedé čtverečky)

body\_parts.clear()

# Resetování skóre

score = 0

score\_sign.clear()

score\_sign.write(f"Skóre: {score} Nejvyšší skóre: {highest\_score}", align="center", font=("Arial", 18))

**44. Python - Celý kód na GitHubu**

**Video:** [**https://youtu.be/pqPqh45yQcg**](https://youtu.be/pqPqh45yQcg)

Celý kód Hadí hry najdete na mém GitHubu:

<https://github.com/DavidSetek/snake-game-yt>

**45. Python - OOP - Vše v Pythonu je objekt, classa**

**Video:** [**https://youtu.be/fiVRow0PQ3Y**](https://youtu.be/fiVRow0PQ3Y)

# Objektově orientované programování

print(type(5))

print(type("david"))

print(type(True))

print(type(()))

print(type([]))

# Atributy a metody

class Car:

# code

pass

car1 = Car()

car2 = Car()

car3 = Car()

print(type(car1))

**46. Python - OOP - Atributy a konstruktor**

**Video:** [**https://youtu.be/NAEVr8xnqIY**](https://youtu.be/NAEVr8xnqIY)

# Objektově orientované programování

# Atributy a metody

class WizardPlayer:

# constructor

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

user\_name = input("Jaké bude vaše jméno ve hře? ")

user\_age = int(input("Jaký je váš věk? "))

player1 = WizardPlayer(user\_name , user\_age)

print(player1.name)

print(player1.age)

# player2 = WizardPlayer("Anna", 18)

# print(player2.name)

# print(player2.age)

**47. Python - OOP - metody a jejich propojení s atributy**

**Video:** [**https://youtu.be/LGiV1ZZHTag**](https://youtu.be/LGiV1ZZHTag)

# Objektově orientované programování

# Atributy a metody

class WizardPlayer:

# constructor

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def attack(self):

print("Útok!")

def age\_checker(self):

if self.age >= 18:

print("Můžete hrát")

else:

print("Nemůžete hrát. Váš věk je příliš nízký.")

user\_name = input("Jaké bude vaše jméno ve hře? ")

user\_age = int(input("Jaký je váš věk? "))

player1 = WizardPlayer(user\_name , user\_age)

player1.attack()

player1.attack()

player1.attack()

player1.age\_checker()

**48. Python - OOP - Atribut mimo konstruktor**

**Video:** [**https://youtu.be/626XcAmDa\_4**](https://youtu.be/626XcAmDa_4)

# Objektově orientované programování

# Atributy a metody

class WizardPlayer:

wizard\_club = True

# constructor

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def attack(self):

print("Útok!")

def age\_checker(self):

if self.age >= 18:

print("Můžete hrát")

else:

print("Nemůžete hrát. Váš věk je příliš nízký.")

user\_name = input("Jaké bude vaše jméno ve hře? ")

user\_age = int(input("Jaký je váš věk? "))

player1 = WizardPlayer(user\_name , user\_age)

print(player1.wizard\_club)

**49. Python - OOP - Příkaz help**

**Video:** [**https://youtu.be/dx83BBU-fCY**](https://youtu.be/dx83BBU-fCY)

help(player1)

help(list)

**50. Python - OOP - Defaultní hodnoty u konstruktoru a co konstruktor umí**

**Video:** [**https://youtu.be/Y93V\_iT2Ya8**](https://youtu.be/Y93V_iT2Ya8)

# Objektově orientované programování

# Atributy a metody

class WizardPlayer:

wizard\_club = True

# constructor

def \_\_init\_\_(self, name="anonym", age=0):

if age >= 18:

self.name = name

self.age = age

def attack(self):

print("Útok!")

def age\_checker(self):

if self.age >= 18:

print("Můžete hrát")

else:

print("Nemůžete hrát. Váš věk je příliš nízký.")

# user\_name = input("Jaké bude vaše jméno ve hře? ")

# user\_age = int(input("Jaký je váš věk? "))

player1 = WizardPlayer("David", 10)

print(player1.age) # vyhodí chybu

**51. Python - OOP - Tvoříme smečku psů (procvičování)**

**Video:** [**https://youtu.be/DYMkqVxlsU0**](https://youtu.be/DYMkqVxlsU0)

Zadání

# Máte zadanou tuto classu

class Dog:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

# Vytvořte 3 objekty (instance) podle classy

# dokážete vysvětlit, jaký je vztah mezi classou a objektem?

# Vytvořte funkci, která určí nejstaršího psa z vámi zadaných

# Vypište výslednou větu "Věk nejstaršího psa: X"

Řešení - 1. způsob

# Máte zadanou tuto classu

class Dog:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

# Vytvořte 3 objekty (instance) podle classy

# dokážete vysvětlit, jaký je vztah mezi classou a objektem?

dog\_1 = Dog("dogty", 3)

dog\_2 = Dog("mogty", 2)

dog\_3 = Dog("hagty", 5)

# Vytvořte funkci, která určí nejstaršího psa z vámi zadaných

dogs = [dog\_1, dog\_2, dog\_3]

def oldest(all\_dogs):

oldest\_dog\_age = 0

for one\_dog in all\_dogs:

if one\_dog.age > oldest\_dog\_age:

oldest\_dog\_age = one\_dog.age

return oldest\_dog\_age

result = oldest(dogs)

# Vypište výslednou větu "Věk nejstaršího psa: X"

print(f"Věk nejstaršího psa: {result}")

Řešení - 2. způsob

# Máte zadanou tuto classu

class Dog:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

# Vytvořte 3 objekty (instance) podle classy

# dokážete vysvětlit, jaký je vztah mezi classou a objektem?

dog\_1 = Dog("dogty", 3)

dog\_2 = Dog("mogty", 2)

dog\_3 = Dog("hagty", 5)

# Vytvořte funkci, která určí nejstaršího psa z vámi zadaných

def oldest(\*args):

return max(args)

result = oldest(dog\_1.age, dog\_2.age, dog\_3.age, 120)

# Vypište výslednou větu "Věk nejstaršího psa: X"

print(f"Věk nejstaršího psa: {result}")

**52. Python - OOP - Statické metody, class metody**

**Video:** [**https://youtu.be/iaFMETldMNA**](https://youtu.be/iaFMETldMNA)

# Objektově orientované programování

# Atributy a metody

class WizardPlayer:

wizard\_club = True

# constructor

def \_\_init\_\_(self, name="anonym", age=0):

self.name = name

self.age = age

def attack(self):

print("Útok!")

def age\_checker(self):

if self.age >= 18:

print("Můžete hrát")

else:

print("Nemůžete hrát. Váš věk je příliš nízký.")

@staticmethod

def test\_function(n1, n2):

return n1 + n2

@classmethod

def test\_function2(cls, player\_name, n1, n2):

return cls(player\_name, n1 + n2)

# print(WizardPlayer.test\_function(60, 100))

test\_player = WizardPlayer.test\_function2("Ron", 30, 20)

print(test\_player.name)

print(test\_player.age)

test\_player2 = WizardPlayer.test\_function2("Hermiona", 10, 10)

print(test\_player2.name)

print(test\_player2.age)

**53. Python - OOP - Encapsulation neboli zapouzdření (1. pilíř OOP)**

**Video:** [**https://youtu.be/Bs-F9G7\_1QY**](https://youtu.be/Bs-F9G7_1QY)

# Encapsulation = zapouzdření

# Atributy a metody

class WizardPlayer:

def \_\_init\_\_(self, name="anonym", age=0):

self.name = name

self.age = age

def attack(self):

print("Útok!")

def age\_checker(self):

if self.age >= 18:

print("Můžete hrát")

else:

print("Nemůžete hrát. Váš věk je příliš nízký.")

# print(WizardPlayer.test\_function(60, 100))

player1 = WizardPlayer("david", 25)

**54. Python - OOP - Abstraction neboli abstrakce (2. pilíř OOP)**

**Video:** [**https://youtu.be/mropQqLueuI**](https://youtu.be/mropQqLueuI)

# 4 pilíře OOP

# Encapsulation = zapouzdření

# Abstraction = abstrakce = dáváme přístup pouze k tomu, co je zapotřebí

class WizardPlayer:

def \_\_init\_\_(self, name="anonym", age=0):

self.**\_**name = name

self.**\_**age = age

def attack(self):

print("Útok!")

def age\_checker(self):

if self.age >= 18:

print("Můžete hrát")

else:

print("Nemůžete hrát. Váš věk je příliš nízký.")

# print(WizardPlayer.test\_function(60, 100))

player1 = WizardPlayer("david", 25)

player1.\_name = "martin"

# player1.attack = "ahoj"

# print(player1.attack)

**55. Python - OOP - Inheritance neboli dědění (3. pilíř OOP)**

**Video:** [**https://youtu.be/HSKMOsZ4j2Q**](https://youtu.be/HSKMOsZ4j2Q)

# 4 pilíře OOP

# Encapsulation = zapouzdření

# Abstraction = abstrakce = dáváme přístup pouze k tomu, co je zapotřebí

# Inheritance = dědění

class WizardPlayer:

def \_\_init\_\_(self, name="anonym", age=0):

self.name = name

self.age = age

def attack(self):

return "Útok!"

class HeadWizard(WizardPlayer):

def avada\_kedavra(self):

return "Avada Kedavra"

player1 = WizardPlayer("david", 25)

print(player1.name)

print(player1.age)

print(player1.attack())

print("--------------------")

player2 = HeadWizard("jana", 18)

print(player2.name)

print(player2.age)

print(player2.attack())

print(player2.avada\_kedavra())

**56. Python - OOP - Inheritance a isinstance()**

**Video:** [**https://youtu.be/IuvI6KYs1DY**](https://youtu.be/IuvI6KYs1DY)

Jakmile budeme mít více class a různé classy budou dědit od různých class, tak v tom může vzniknout docela nepořádek. V tu chvíli nám přijde vhod, když budeme chtít zjistit, jestli je nějaký objekt instance nějaké classy - jinak řečeno - jestli byl objekt vytvořen podle této classy nebo ne. Proto použijeme **isinstance()**

**isinstance()** vrací **True** (pokud objekt je vytvořen podle classy) nebo vrací **False** (pokud objekt není vytvořen podle classy)

player2 = HeadWizard("jana", 18)

print(player2.name)

print(player2.age)

print(player2.attack())

print(player2.avada\_kedavra())

print("--------------------")

print(isinstance(player1, WizardPlayer)) # true

print(isinstance(player1, HeadWizard)) # false

print(isinstance(player2, WizardPlayer)) # true

print(isinstance(player2, HeadWizard)) # true

**57. Python - OOP - Polymorphism neboli mnoho forem (4. pilíř OOP)**

**Video:** [**https://youtu.be/ZnaLnsK7dBM**](https://youtu.be/ZnaLnsK7dBM)

# 4 pilíře OOP

# Encapsulation = zapouzdření

# Abstraction = abstrakce = dáváme přístup pouze k tomu, co je zapotřebí

# Inheritance = dědění

# Polymorphism = mnoho forem

class WizardPlayer:

def \_\_init\_\_(self, name="anonym", age=0):

self.name = name

self.age = age

def attack(self):

return "Útok 1. stupně!"

class HeadWizard(WizardPlayer):

def attack(self):

return "Útok 2. stupně!"

def avada\_kedavra(self):

return "Avada Kedavra"

player1 = WizardPlayer("david", 25)

print(player1.attack())

print("--------------------")

player2 = HeadWizard("jana", 18)

print(player2.attack())

# print("--------------------")

# print(isinstance(player1, WizardPlayer)) # true

# print(isinstance(player1, HeadWizard)) # false

# print(isinstance(player2, WizardPlayer)) # true

# print(isinstance(player2, HeadWizard)) # true

**58. Python - OOP - Jak se používá super() v OOP**

**Video:** [**https://youtu.be/E2I1NFjqkcY**](https://youtu.be/E2I1NFjqkcY)

# 4 pilíře OOP

# Encapsulation = zapouzdření

# Abstraction = abstrakce = dáváme přístup pouze k tomu, co je zapotřebí

# Inheritance = dědění

# Polymorphism = mnoho forem

class WizardPlayer:

def \_\_init\_\_(self, name="anonym", age=0):

self.name = name

self.age = age

def attack(self):

return "Útok 1. stupně!"

class HeadWizard(WizardPlayer):

def \_\_init\_\_(self, type, name, age):

super().\_\_init\_\_(name, age)

self.type = type

def attack(self):

return "Útok 2. stupně!"

def avada\_kedavra(self):

return "Avada Kedavra"

# player1 = WizardPlayer("david", 25)

# print(player1.attack())

# print("--------------------")

player2 = HeadWizard("good", "david", 35)

print(player2.type)

print(player2.name)

print(player2.age)

# print("--------------------")

# print(isinstance(player1, WizardPlayer)) # true

# print(isinstance(player1, HeadWizard)) # false

# print(isinstance(player2, WizardPlayer)) # true

# print(isinstance(player2, HeadWizard)) # true

**59. Python - OOP - Introspekce v OOP**

**Video:** [**https://youtu.be/IRukOZr2bS0**](https://youtu.be/IRukOZr2bS0)

Může vás napadnout, co všechno player2 na sobě má - jaké atributy a jaké metody. To zjistíme, když si vyprintujeme funkci dir. Zápis bude vypadat takto.

# introspection

print(dir(player2))

Do terminálu nám to vypíše seznam atributů a metod, ke kterým má player2 přístup - např. age, name, type, \_\_init\_\_ atd.

**60. Python - OOP - Dunder methods v OOP**

**Video:** [**https://youtu.be/XGfgNWsWr3c**](https://youtu.be/XGfgNWsWr3c)

My jsme se s dunder methodami již setkali. Např. metoda init u konstruktoru (\_\_init\_\_) je dunder metoda. Těchto metod je ale více. Pojďme se podívat, jak fungují a k čemu slouží.

# Dunder Methods

print(dir(player2))

print("--------------------")

print(player2.\_\_dir\_\_())

print(len([5, 8, 9]))

print("--------------------")

print([5, 8, 9].\_\_len\_\_())

print(str(player2))

print("--------------------")

print(player2.\_\_str\_\_())

**61. Python - OOP - Method resolution order neboli MRO metoda**

**Video:** [**https://youtu.be/AIv-77wK1RQ**](https://youtu.be/AIv-77wK1RQ)

Co když jste v situaci, kdy před sebou máte složitý kód a vidíte, že něco dědí od něčeho jiného a něco dalšího dědí od dalších několika class atd. Tak se může hodit, pokud si vyjedete, od čeho konkrétní classa dědí - vyjedete si seznam.

Pozor - nejde jen o seznam, ale také o posloupnost toho, kde se daná metoda nebo atribut hledá.

K tomu všemu nám slouží mro - method resolution order

# Method resolution order = MRO

print(HeadWizard.mro())

print(HeadWizard.\_\_mro\_\_)

print(WizardPlayer.mro())

print(WizardPlayer.\_\_mro\_\_)

**62. Python pro pokročilé - Zpět k základům - Proč teď?**

**Video:** [**https://youtu.be/yYPkvsD\_QCg**](https://youtu.be/yYPkvsD_QCg)

**63. Python pro pokročilé - Zpět k základům - Co potřebujete k ovládnutí programovacího jazyka**

**Video:** [**https://youtu.be/R2t5\_23oQeU**](https://youtu.be/R2t5_23oQeU)

**64. Python pro pokročilé - Zpět k základům - Přehled datových typů**

**Video:** [**https://youtu.be/-jvKX-RiCfM**](https://youtu.be/-jvKX-RiCfM)

# Základní datové typy

str

int

float

bool

list

tuple

dict

set

# Classes -> custom type

WizardPlayer

# Special data types -> extra typy dat např. z modulů

Modules

# None -> nothing (absence hodnoty)

None

age = None

print(age)

**65. Python pro pokročilé - Zpět k základům - Matematické funkce**

**Video:** [**https://youtu.be/H60H6p9vAw4**](https://youtu.be/H60H6p9vAw4)

Seznam funkcí modulu math:

<https://www.programiz.com/python-programming/modules/math>

# Matematické funkce

# Import modulu math

import math

# Nepotřebujeme modul math

print(round(5.3))

print(round(5.9))

print(abs(-5))

# Potřebujeme modul math

print(math.sqrt(16))

**66. Python pro pokročilé - Zpět k základům - Binární čísla**

**Video:** [**https://youtu.be/I4qqO2CZ0ok**](https://youtu.be/I4qqO2CZ0ok)

# Binární čísla

bin1 = bin(5) #0b101

bin2 = bin(10) #0b1010

# Binární číslo zpět na celé číslo

print(int("0b101", 2))

**67. Python pro pokročilé - Zpět k základům - Proměnné a co dělat a nedělat**

**Video:** [**https://youtu.be/eWAgkdLWinw**](https://youtu.be/eWAgkdLWinw)

# Proměnné

# Běžné proměnné

height = 186

age = 40

# Konstanty

PI = 3.14

# Více proměnných

a, b, c = 1, 2, 3

print(a)

print(b)

print(c)

# prohození hodnot v proměnných

x = 8

y = 2

print(x, y)

# z = x

# x = y

# y = z

x, y = y, x

print(x, y)

# nikdy netvořit proměnnou s dvěma podtržítky na začátku!!

**68. Python pro pokročilé - Zpět k základům - Expression a statement**

**Video:** [**https://youtu.be/lBvmqQkqjLs**](https://youtu.be/lBvmqQkqjLs)

# Expression a statement

x = 5

x / 2 # expression

y = 10 # statement

user\_age = x / 2 # statement

**69. Python pro pokročilé - Zpět k základům - Další způsob výpisu stringu**

**Video:** [**https://youtu.be/rlG6rHeZiFc**](https://youtu.be/rlG6rHeZiFc)

# String

print("Ahoj")

long\_string = '''

jklfdsa

fkdlsa

jfkdlsa

kfjldsůa

jkfldsa

'''

print(long\_string)

**70. Python pro pokročilé - Zpět k základům - Escapování**

**Video:** [**https://youtu.be/FfF6FXGz6Hk**](https://youtu.be/FfF6FXGz6Hk)

# Escape sequence

info = 'it\'s mine'

enter = "text \n další text"

tabulator = "text \t další text"

**71. Python pro pokročilé - Zpět k základům - Formátovaný string s format()**

**Video:** [**https://youtu.be/Mnlq92\_zrUU**](https://youtu.be/Mnlq92_zrUU)

# Formátovaný string

name = "David"

age = 55

print("Ahoj, já jsem " + name + ". A je mi " + str(age))

print(f"Ahoj, já jsem {name}. A je mi {age}")

print("Ahoj, já jsem {}. A je mi {}".format("David", 55))

# print("Ahoj, já jsem {}. A je mi {}".format(55, "David"))

print("Ahoj, já jsem {}. A je mi {}".format(name, age))

print("Ahoj, já jsem {0}. A je mi {1}".format(name, age))

print("Ahoj, já jsem {my\_name}. A je mi {my\_age}".format(my\_name = "Harry", my\_age = 22))

print("Ahoj, já jsem {my\_name}. A je mi {my\_age}".format(my\_age = 22, my\_name = "Harry"))

**72. Python pro pokročilé - Zpět k základům - Práce se stringem pomocí indexů**

**Video:** [**https://youtu.be/BsRP1yvYV40**](https://youtu.be/BsRP1yvYV40)

# Práce se stringem

name = "testovaci"

#012345678

print(name[0]) # t

print(name[2]) # s

# [start:stop]

print(name[0:4]) # test

print(name[2:5]) # sto

#[start:stop:krok]

print(name[0:7:2]) # tsoa

print(name[0:7:3]) # tta

# kombinace

print(name[1:]) # estovaci

print(name[:6]) # testov

print(name[::1]) # testovaci

print(name[-1]) # i

print(name[-2]) # c

print(name[-3]) # a

print(name[::-1]) # icavotset

print(name[::-2]) # iaost

**73. Python pro pokročilé - Zpět k základům - Immutability neboli neměnnost**

**Video:** [**https://youtu.be/nrayPm5A3Kw**](https://youtu.be/nrayPm5A3Kw)

# Immutability = neměnnost

my\_name = "david"

my\_name[0] = "m" # vyhodí chybu

my\_name = "harry"

print(my\_name)

**74. Python pro pokročilé - Zpět k základům - Rozdíl mezi funkcí a metodou**

**Video:** [**https://youtu.be/Rl0eNn2nhxE**](https://youtu.be/Rl0eNn2nhxE)

# Metody a funkce

# Funkce

my\_name = "David"

print(len(my\_name))

print(abs(-9))

# Metody

print(my\_name.upper())

print(my\_name.lower())

**75. Python pro pokročilé - Zpět k základům - Jak na vyhvězdičkování hesla**

**Video:** [**https://youtu.be/f-MLIas2O3U**](https://youtu.be/f-MLIas2O3U)

user\_name = input("Zadejte své uživatelské jméno: ")

password = input("Zadejte své heslo: ")

print(f"{user\_name}, vaše heslo je {'\*' \* len(password)} a délka vašeho hesla je {len(password)}")

**76. Python pro pokročilé - Zpět k základům - Slicing a list**

**Video:** [**https://youtu.be/-vCguHRaWLk**](https://youtu.be/-vCguHRaWLk)

# Slicing

my\_name = "testovaci"

# print(my\_name[:5:2])

# my\_name[0] = "m"

to\_do = [

"nakrmit kočku",

"vyvenčit psa",

"udělat svačinu",

"vyměnit žárovku",

"dojít nakoupit"

]

# print(to\_do[::1])

# list je mutable

# to\_do[0] = "nový úkol"

# print(to\_do)

# pozor

# to\_do2 = to\_do

# to\_do2[0] = "super nový úkol"

# print(to\_do)

# print(to\_do2)

# zkopírování listu do nového

# to\_do3 = to\_do[:]

# to\_do3[0] = "něco udělej"

# print(to\_do)

# print(to\_do3)

**77. Python pro pokročilé - Zpět k základům - Matice a strojové učení**

**Video:** [**https://youtu.be/VnYPFE9xE34**](https://youtu.be/VnYPFE9xE34)

# Matrix - 2 dimezionální list

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print(matrix[0][1]) # 2

# 4, 8, 9

print(matrix[1][0]) # 4

print(matrix[2][1]) # 8

print(matrix[2][2]) # 9

matrix2 = [

[1, 0, 0],

[1, 0, 0],

[1, 1, 1]

]

**78. Python pro pokročilé - Zpět k základům - Metody listu (append, insert, extend, clear, pop. remove)**

**Video:** [**https://youtu.be/DlpUG3TMt9A**](https://youtu.be/DlpUG3TMt9A)

Seznam metod u listu:

<https://www.w3schools.com/python/python_ref_list.asp>

# Metody a list

to\_do = [

"nakrmit kočku",

"vyvenčit psa",

"udělat svačinu",

"vyvenčit psa"

]

# append přidává položku na konec listu

to\_do.append("vyměnit žárovku")

to\_do.append("koupit nový telefon")

# vkládá položku na konkrétní index (nic nemaže)

to\_do.insert(1, "utřít prach")

# rozšiřuje list o větší množství položek pomocí listu

to\_do.extend(["vyčistit odpad", "umýt okna"])

# promaže všechny položky v listu

to\_do.clear()

# vymaže poslední položku v listu nebo vymaže položku na zadaném indexu

to\_do.pop()

to\_do.pop(1)

# odstraní první výskyt položky, kterou zadáme

to\_do.remove("vyvenčit psa")

print(to\_do)

**79. Python pro pokročilé - Zpět k základům - Metody listu (index, použití in, count)**

**Video:** [**https://youtu.be/EC7UH2RA0Q4**](https://youtu.be/EC7UH2RA0Q4)

Seznam metod u listu:

<https://www.w3schools.com/python/python_ref_list.asp>

# Metody a list

to\_do = [

"nakrmit kočku",

"vyvenčit psa",

"udělat svačinu"

]

print(to\_do.index("udělat svačinu")) # 2

print("vyvenčit psa" in to\_do) # True

print("a" in "ahoj") # True

print(to\_do.count("udělat svačinu")) # 1

**80. Python pro pokročilé - Zpět k základům - Metoda copy a reverse, metody tzv. in place**

**Video:** [**https://youtu.be/UbugTqtIpM8**](https://youtu.be/UbugTqtIpM8)

Seznam metod u listu:

<https://www.w3schools.com/python/python_ref_list.asp>

# Metody a list

to\_do = [

"nakrmit kočku",

"vyvenčit psa",

"udělat svačinu",

"absolvovat lékařskou prohlídku"

]

# In place

to\_do2 = to\_do.sort() # None, protože sort nevrací hodnotu

# Obě dvě proměnné směřují v paměti na stejný list

to\_do2 = to\_do

# tato změna se promítne v to\_do i v to\_do2

to\_do2[0] = "nový úkol"

# Takto se zkopíruje to\_do do to\_do3 a jsou zcela oddělené

to\_do3 = to\_do.copy()

# Tato změna se promítne jen v to\_do3

to\_do3[0] = "NOVÝ ÚKOL"

# Obrácený výpis položek

to\_do.reverse()

print(to\_do)

**81. Python pro pokročilé - Zpět k základům - Metoda join**

**Video:** [**https://youtu.be/12Cr50INaP4**](https://youtu.be/12Cr50INaP4)

# Metody a list

to\_do = [

"nakrmit kočku",

"vyvenčit psa",

"udělat svačinu",

"absolvovat lékařskou prohlídku"

]

to\_do.sort()

# # ['absolvovat lékařskou prohlídku', 'nakrmit kočku', 'udělat svačinu', 'vyvenčit psa']

to\_do.reverse()

# # ['vyvenčit psa', 'udělat svačinu', 'nakrmit kočku', 'absolvovat lékařskou prohlídku']

print(to\_do[::-1])

# # ['absolvovat lékařskou prohlídku', 'nakrmit kočku', 'udělat svačinu', 'vyvenčit psa']

print(list(range(100)))

pozdrav = " ".join(["ahoj", "já", "jsem", "David"])

print(pozdrav)

**82. Python pro pokročilé - Zpět k základům - List unpacking**

**Video:** [**https://youtu.be/IB-lO3TJlJA**](https://youtu.be/IB-lO3TJlJA)

Základní list pro zkopírování

to\_do = [

"nakrmit kočku",

"vyvenčit psa",

"udělat svačinu",

"absolvovat lékařskou prohlídku",

"utřít prach",

"vymalovat pokoj",

"koupit nový telefon"

]

List Unpacking

a, b, c, d, e, f, g = to\_do

print(a)

print(b)

print(c)

print(d)

print(e)

print(f)

print(g)

a, \*rest, g = to\_do

print(a)

print(rest)

print(g)

print(to\_do)

**83. Python pro pokročilé - Zpět k základům - Metody a dictionary**

**Video:** [**https://youtu.be/\_sMrRrsRlYw**](https://youtu.be/_sMrRrsRlYw)

Seznam metod pro dictionary:

<https://www.w3schools.com/python/python_ref_dictionary.asp>

# Dictionary

book = {

"title": "Harry Potter a kámen mudrců",

"author": "J. K. Rowling",

"year": 1997

}

# print(book["title"])

# print(book["author"])

# print(book["year"])

# print("year" in book.keys())

# print(1997 in book.values())

# print(1997 in book.values())

# print(book.items())

# book.clear()

# print(book)

# book.pop("author")

# print(book)

# book.popitem()

# book.popitem()

# print(book)

# book.update({"year": 1998})

# print(book)

# book.update({"pages": 288})

# print(book)

**84. Python pro pokročilé - Zpět k základům - Tuples**

**Video:** [**https://youtu.be/0xZ-cPzmcyc**](https://youtu.be/0xZ-cPzmcyc)

# Tuple

first\_tuple = ("z", 1, 2, 3, 4, 5)

# first\_tuple[0] = "a"

print(first\_tuple[0])

print(3 in first\_tuple)

colors = {

(1, 2): (255, 0, 0),

"green": (0, 255, 0),

"blue": (0, 0, 255)

}

print(colors[(1, 2)])

**85. Python pro pokročilé - Zpět k základům - Pygame a tuple (ukázka)**

**Video:** [**https://youtu.be/PefGkpuI3q0**](https://youtu.be/PefGkpuI3q0)

**86. Python pro pokročilé - Zpět k základům - Tuple, slicing a metody**

**Video:** [**https://youtu.be/5JxhYySYZro**](https://youtu.be/5JxhYySYZro)

Metody pro tuple:

<https://www.w3schools.com/python/python_ref_tuple.asp>

# new\_tuple = first\_tuple[0:2]

# new\_tuple = first\_tuple[::2]

# print(new\_tuple)

# x = first\_tuple[0]

# y = first\_tuple[1]

# x, y = first\_tuple[0], first\_tuple[1]

# print(x)

# print(y)

# x, y, z, \*other = ("a", "b", "c", "d", "e", "f")

# print(x)

# print(y)

# print(z)

# print(other)

# Tuple

first\_tuple = ("a", "b", "c", "d", "c", "c")

# Metody

print(first\_tuple.count("c")) # 3

print(first\_tuple.index("b")) # 1

**87. Python pro pokročilé - Zpět k základům - datový typ set (unikátní neseřazené hodnoty)**

**Video:** [**https://youtu.be/iZ\_PzGsn3x8**](https://youtu.be/iZ_PzGsn3x8)

# str

# int, float

# bool

# list

# dict

# tuple

# set

# Set - unikátní neseřazené hodnoty

first\_set = {1, 2, 2, 2, 3, 8, 5, 5, 5}

# print(first\_set)

# first\_set.add(100)

# print(first\_set)

# first\_set.remove(2)

# print(first\_set)

my\_name = "davidsetek"

my\_set = set(my\_name)

# print(my\_set)

my\_list = ["david", "jana", "petr", "david"]

result = set(my\_list)

print(result)

**88. Python pro pokročilé - Zpět k základům - Set a co s ním dělat a nedělat**

**Video:** [**https://youtu.be/yFqEbu9QeGs**](https://youtu.be/yFqEbu9QeGs)

first\_set = {1, 2, 2, 2, 3, 8, 5, 5, 5}

# print(first\_set) # {1, 2, 3, 5, 8}

# print(first\_set[0])

# Vypsání pomocí cyklu

for x in first\_set:

print(x)

print(3 in first\_set) # True

print(len(first\_set)) # 5

old\_set = first\_set.copy() # kopírování setu

first\_set.add(100)

print(first\_set)

print(old\_set)

**89. Python pro pokročilé - Zpět k základům - Set a jeho metody**

**Video:** [**https://youtu.be/d6ZnFENVxN8**](https://youtu.be/d6ZnFENVxN8)

<https://www.w3schools.com/python/python_ref_set.asp>

first\_set = {1, 2, 3}

second\_set = {2, 3, 4, 5, 6, 7, 8}

# Rozdíl = difference

print(first\_set.difference(second\_set))

# Odstranění = remove a discard

# first\_set.remove(9)

first\_set.discard(9)

print(first\_set)

# Rozdíl, ale změní first\_set

first\_set.difference\_update(second\_set)

print(first\_set)