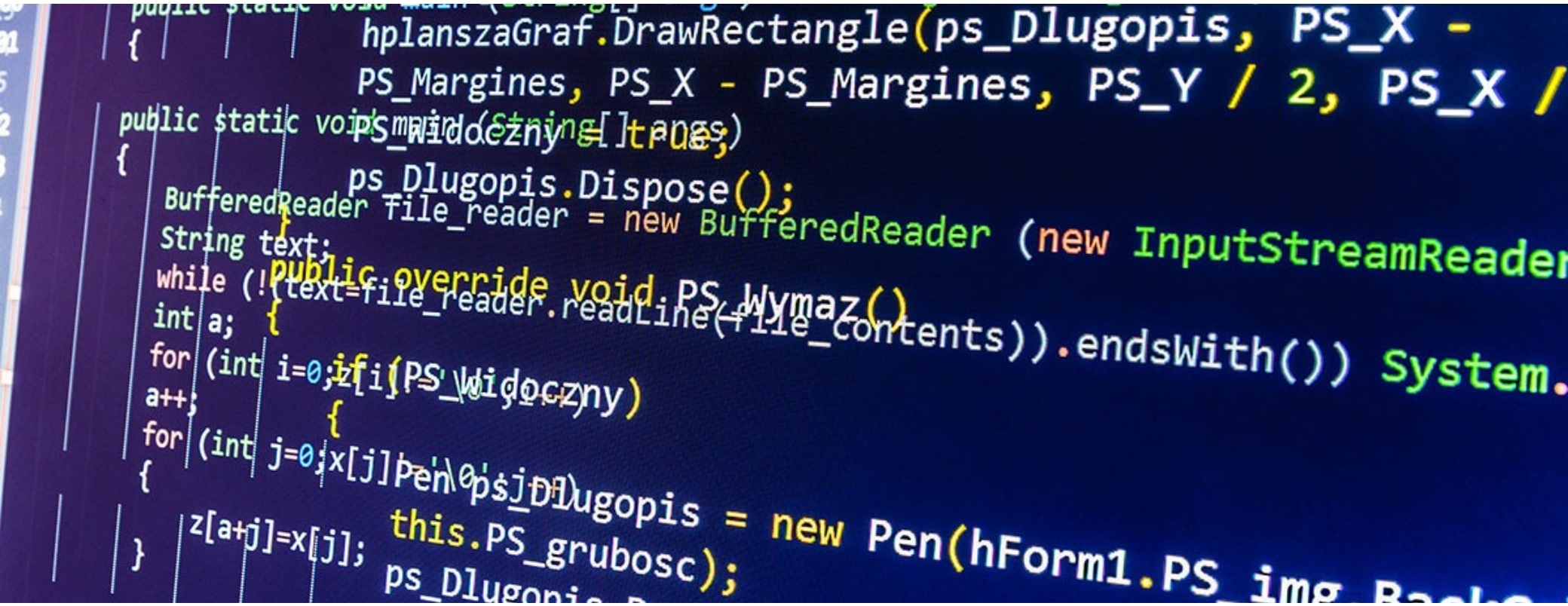
sfb Höhere Fachschule für Technologie und Management



Grundlagen Programmierung 1

PB4 – Error Handling, Funktionen, Zahlen und Wahrheitswerte

sfb Höhere Fachschule für Technologie und Management

Hausaufgaben und Fragen

/ Ergebnis Präsentation & Besprechung

/ Fragen?

sfb Höhere Fachschule für Technologie und Management

Error Handling

Wie kann man mit Fehlern umgehen?

Try und except

Was passiert hier?

/ Annahme:
  / Sie möchten einen Benutzer dazu bringen, eine Zahl einzugeben. Siehe Code von Theis (S. 66) hier:

```
Print(„Geben sie die ganze Zahl ein“)
z = input()
zahl = int(z)
```

  / Was passiert an der Rot markierten Stelle, wenn die Eingabe keine Zahl war?

Try und except

/ Try umschliesst den Code, bei dem ein Fehler auftreten kann, z.B:

```
try:
    zahl = int(z)
    print("Sie haben die ganze Zahl richtig eingegeben")
```

Except umschliesst den Codeteil, welches den Fehler abfangen, z.B.

```
except:
    print("Sie haben keine Zahl eingegeben")
```

/ Programm läuft jetzt ohne Abbruch

sfb Höhere Fachschule für Technologie und Management

Einzelarbeit try und except – 20min

- Aufgabenstellung:
  - Nehmen sie ihre Lösung zur Aufgabe 2 vom PB 3 zur Hand
  - Fangen sie hier den möglichen Fehler ab, dass ein Benutzer Text anstatt eine Zahl eingeben kann
  - Geben sie aus, dass der Benutzer eine falsche Eingabe gemacht hat

  - Aufgabe Extra:
    - Den Benutzer beim Fehler dazu bringen, die Eingabe zu wiederholen (siehe Kap. 3.6.3)

sfb Höhere Fachschule für Technologie und Management

Funktionen und Module

Den Code strukturieren und mehrfach verwenden

Funktionen mit einem/mehreren Parametern
Funktionen mit Rückgabewert
Komplexe Funktionen

Modularisierung

- Modularisierung beschreibt das Zerlegen von Programmteilen in selbstgeschriebenen Funktionen, damit:
  - Der Codeteil mehrfach verwendet werden kann
  - Übersichtlicher vom Aufbau her
  - Welche dann Wartung und Pflege erleichtert
  - Verständlicher für andere Programmierer

Funktionen

/ Vordefinierte Funktionen, wie input() oder selbstdefinierbare
/ Rückgabewerte können definiert werden, sofern notwendig

/ Aufbau einer Funktion:

```
def funktionName(parameter, parameter2):
        # Code
```

/ Aufruf einer Funktion:

```
funktionName(1, 2)
```

Funktionen mit Rückgabewerten

/ Rückgabewerte werden mit return zurückgegeben

/ Aufbau einer Funktion mit Rückgabewert:

```
def funktionName(parameter, parameter2):
        # Code
        ergebnis = parameter + parameter2
        return ergebnis
```

/ Aufruf einer Funktion:

```
print("Das Ergebnis ist", funktionName(1, 2))
```

sfb Höhere Fachschule für Technologie und Management

Gruppenarbeit (2 Personen) – 30min

- Aufgabenstellung:
  - Schreiben Sie eine Funktion, der sie einen Parameter mph übergeben können, welche dann kmh Wert zurückgibt
  - Suchen sie dazu die benötigte Umrechnung im Internet

Komplexere Funktionen

```
/  Funktionen können mehrere Parameter als Input erhalten
/  Diese werden Kommasepariert ausgewiesen:
def complex_function(mph, kmh, windgeschwindigkeit):
    # Code hier
    print(mph)
    print(kmh)
    print(windgeschwindigkeit)

/  Man kann auch Standardwerte setzen lassen, falls die Variable nicht übergeben wird:
def complex_function(mph=60, kmh=100, windgeschwindigkeit=None):
    # Code hier
    print(mph)
    print(kmh)
    print(windgeschwindigkeit)
```

sfb Höhere Fachschule für Technologie und Management

Einzelaufgabe – 20min

- Schreibe eine Funktion namens parameter_funktion, die drei Parameter akzeptiert: param1, param2 und param3.
- Setze für jeden Parameter Standardwerte:
  - param1: "Standardwert für param1"
  - param2: 100
  - param3: Eine Liste [1, 2, 3]
- Innerhalb der Funktion überprüfe, ob die übergebenen Parameter den Wert None haben. Wenn ja, setze den entsprechenden Standardwert.
- Gib die Werte der Parameter aus, nachdem sie entweder durch die übergebenen Argumente oder durch die Standardwerte festgelegt wurden.
- Teste deine Funktion, indem du sie mehrmals aufrufst:
  - Rufe die Funktion ohne Argumente auf, um die Standardwerte zu verwenden.
  - Rufe die Funktion mit einem oder mehreren Argumenten auf, um zu überprüfen, ob die Standardwerte korrekt gesetzt werden, wenn None übergeben wird.
  - Rufe die Funktion mit allen Argumenten auf, um zu sehen, wie die übergebenen Werte die Standardwerte ersetzen.

Datentypen

Datentypen von Python kennenlernen

Ganze Zahlen,
Zahlen mit Nachkommastellen,
Wahrheitswerte,
Uvm.

Objekte

- Da Python Objektorientiertes programmieren ist, ist alles ein Objekt
- Datentyp einer Variable muss nicht festgelegt werden
- Wir unterscheiden zwei Typen von Objekten:
  - Einzelne Objekte, Zahlen oder Zeichen
  - Zusammengehörende Gruppen von Objekten, wie z.B. Strings, Listen, Tupel (folgt in PB5 und PB6)

Ganze Zahlen

/ Typ: int (integer)
/ Zahlen dieses Typs sind unendlich genau
/ Dezimalsystem wird benutzt

/ a = 5 -> Variable ist ein Objekttyp int

Zahlen mit Nachkommastellen

/ Typ: float (Fliesskommazahlen)
/ Zahlen haben Dezimalpunkt oder Exponentialschreibweise

/ a = 5.4 -> Variable ist ein Objekttyp float (Dezimalpunkt)
/ b = 4.2e-3 -> Variable ist ein Objekttyp float (Exponential)

Wahrheitswerte

/ Boolsche Werte true und false (Wahr und Falsch)
/ Alle Objekte besitzen einen Wahrheitswert
/ Funktion bool() liefert Wahrheitswert zurück, z.B.

x = 11>10
print(x) -> **Ausgabe: True**

Ermittlung des Datentyps

/ Ermittlung nützlich, denn es ist wichtig den Datentyp zu kennen
/ Dazu gibt es im Python die Funktion type()
/ Diese gibt die Klasse, in diesem Fall den Objekttyp, aus

```
a = 5
print("Der Objekttyp von a ist: ", type(a))
```

Rundung und Konvertierung

- Die Objekttypen wie int haben auch eine Konvertierungsfunktion int()
- Rundungen folgen mit Funktion round(objekt, rundungsstellen)
- Beispiel Theis (4.1.5):

```
x = 12/7
print("x:", x)                                  → Ausgabe Wert: 1.7142857142857142
rx = round(x, 3)
print("x gerundet auf drei Stellen", rx)        → Ausgabe Wert: 1.714
rx = round(x)
print("x gerundet auf null Stellen", rx)        → Ausgabe Wert: 2
ix = int(x)
print("x als int", ix)                          → Ausgabe Wert: 1
```

Module math

/ Mathematische Funktionen können über die Library math eingebunden werden

➢import math

/ Enthält Funktionen wie im Taschenrechner:
  / math.pi() -> Pi
  / math.sqrt() -> Quadratwurzel
  / math.log() -> Logarithmen
  / math.isclose() -> Nahe dran
  / math.remainder () -> Rest

Nichts (none)

- None bezeichnet das „Nichts“-Objekt
- Funktionen ohne Rückgabewert liefern None zurück
- None dient dazu zu ermitteln, ob Funktionen keine Rückgabewerte liefern oder falsche (Aufgrund Fehler z.B.)
- Beispiel im Buch anschauen (S. 137 & 138)

Gruppenarbeit (2 Personen)

- Aufgabenstellung:
  - Schreiben Sie eine Funktion, die den Typ des Parameters zurückgibt
  - Geben Sie dieser Funktion allen Ihnen bekannten Datentypen mit und sehen sie, ob dieser die Typen richtig zurückgibt

  - Berechnen Sie den Rest von 15.2 und 3.6

sfb Höhere Fachschule für Technologie und Management

Hausaufgaben

/ PDF:
Hausaufgaben_PB4_Fehler_Funktionen_
und_Datentypen.pdf