1. Technische Voraussetzungen

Die Hauptarbeit lag beim ZDV, ein besonderer Dank geht hier an Herrn Stein (Gruppe Röhle). Es musste ermöglicht werden, dass die Programme (Excel, Stata, Python) in der Klausurumgebung von ilias geöffnet werden können. Dies erfolgt über das Öffnen von in der Aufgabenstellung vorgegebenen Dateien.

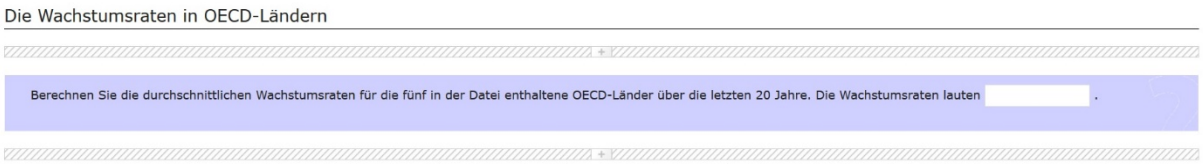
Die Einbindung von Excel in E-Klausuren wurde bereits im Sommersemester 2018 durch das psychologische Institut (**Herr Prof. Günter Meinhardt**) im Zusammenarbeit mit dem ZDV ermöglicht.

Ein herzlicher Dank geht auch an die studentischen Mitarbeiter Herrn Johannes Kochems und Herrn Sergej Samoilov. Diese haben viele Detailfragen gelöst und beantwortet.

2. Klausuraufgaben

2.1. Erstellen von Aufgaben

- Klausuraufgaben werden wie üblich in ilias erstellt
- Es werden Dateien der jeweils notwendigen Programme in der Aufgabenstellung mit eingebunden
    - Als Beispiel betrachten wir eine Aufgabe, die das Berechnen von durchschnittlichen Wachstumsraten in OECD Ländern fordert
    - Das Einbinden von Dateien erfolgt dann über das „Plus“ unten (siehe Abbildung)



- Nach Einbinden schaut die Oberfläche wie folgt aus

- Zur Überprüfung der Funktionalität ist ein spezieller Klausurbrowser für den jeweilig verwendeten PC notwendig
    - Dieser wird vom ZDV installiert (z.B. Frau Michalik)
    - Nach Anklicken der Datei (Datensatz.xlsx) startet Excel und die Datei wird lokal für die Nutzer (Klausurteilnehmer) abgespeichert
    - Klausurteilnehmer können dann in Excel die entsprechende Aufgabe lösen
    - Dies ist analog für andere Dateitypen und damit andere Software möglich

2.2. Lösen von Aufgaben

- Studierende öffnen das jeweilige Programm durch Doppelklick auf die Dateien
- Sie arbeiten im jeweiligen Programm

- Für die Bewertung der Lösung sind zwei Dinge relevant
    - Studierende kopieren das Programm in ein Fenster für freien Text
    - Studierende kopieren Ergebnisse in numerische Felder

3. Automatische Korrektur von Programmieraufgaben

Programmieraufgaben können auf verschiedene Arten gestellt und automatisch korrigiert werden.

3.1. Traditionelle Methoden

Es gibt verschiedene traditionelle Methoden, die schon seit längerem in E-Klausuren eingesetzt werden. Bei diesen traditionellen Fragen wird die dazugehörige Software (Excel, Stata, Python, etc) in der E-Klausur ***nicht*** benötigt. Die Beantwortung verlangt nach Kenntnis der Programmiersprache, die Software wird aber nicht verwendet.

- Methode 1: Sortieren von Programmbestandteilen
- Methode 2: Codeausgabe bestimmen *
  Ein Code wird vorgegeben und Prüfungsteilnehmer bestimmen die Ausgabe. Dazu muss die Funktionsweise einer Programmiersprache Schritt für Schritt nachvollzogen werden. Das Ergebnis des Programms wird aus einer vorgegebenen Liste mit möglichen Programmausgaben über ‚single choice' ausgewählt.
- Methode 3: Vervollständigen eines Programmes
  Für ein vorgegebenes Programmergebnis (z.B. erneut eine Grafik) wird ein Programm angeboten, das jedoch Lücken enthält. Über ein Drop-Down-Menü müssen die passenden Befehle ausgewählt werden. Alternativ können Freitextfelder verwendet werden. Letzteres erhöht natürlich den Korrigieraufwand. So ist z.B. unklar, ob ein Befehl ‚pl ot' (mit einem versehentlichen Leerzeichen) als richtig gewertet wird oder nur ‚plot'.

Diese Methoden sind in ilias implementiert und können **öffentlich** (mit ilias-Berechtigung) eingesehen werden. Die Aufgaben können von jedem Iliasnutzer bearbeitet werden, als wäre er Klausurteilnehmer. Sie können auch in persönliche ilias-Verzeichnisse kopiert werden. Es besteht jedoch kein Schreibrecht.

* Ein herzlicher Dank geht an **Prof. Franz Rothlauf**, von dem wir diese Aufgabentypen übernommen haben

3.2. Neue Methoden

Bei den nun folgenden zwei neuen Ansätzen ist ein Zugriff auf die jeweilige Software während der E-Klausur notwendig.

3.2.1. Methode 4: Bewerten von Lösungen in numerischen Feldern

Studierende geben die Lösung aus einer Programmieraufgabe (z.B. ein Schätzwert aus einer Regression oder der berechnete Wert der Höhe der zukünftigen Arbeitslosenquote) in ein numerisches Feld in ilias ein. Es empfiehlt sich, den numerischen Wert auf „viele" Nachkommastellen angeben zu lassen (4 bis 6). Damit ist sichergestellt, dass der Wert eindeutig ist und durch Raten (oder über Nachbarschaftshilfe) nicht so leicht gefunden werden kann. Die Anzahl der geforderten Nachkommastellen muss in der Aufgabenstellung angegeben werden, da ilias nur dann Punkte

vergibt, wenn der Wert exakt dem verlangten Wert (mit eben der verlangten Anzahl von Nachkommastellen) entspricht

3.2.2. Methode 5: Automatisches Bewerten eines geschriebenen Programmes

Es wird eine Programmieraufgabe gestellt, deren Bearbeitung z.B. 3 zentrale Befehle benötigt. Die Studierenden erstellen ein Programm und kopieren dies in ein Fenster für freien Text.

a) Die Bewertung der verwendeten Befehle

Ilias bietet (mindestens) drei Arten an, diesen freien Text zu bewerten.

- **Automatische Bewertung bei Nennung einzelner Begriffe**
  Für jeden der Befehle wird bei Nennung je ein Punkt vergeben. Daraus ergibt sich eine maximale Punktzahl von (im obigen Beispiel) 5 Punkten. Die Anzahl der Punkte lässt sich für jeden Begriff festlegen. Es kann also für einen „wichtigeren" Befehl mehr Punkte vergeben werden, als für „normale" Befehle.
- **Automatische Bewertung bei Nennung aller Begriffe**
  Eine zu bestimmende Punktzahl wird nur vergeben, wenn alle notwendigen Befehle im Freitextfeld erscheinen.
- **Automatische Bewertung bei Nennung eines Begriffs**
  Die ausgewählte Punktzahl wird vergeben, sofern mindestens einer der notwendigen Befehle genannt wurde. Damit gibt es entweder null Punkte (keiner der Befehle wurde genannt) oder die ausgewählte Punktzahl.

b) Die Bewertung der Anwendung der Befehle

Die Bewertungsmethode in a) kann nicht feststellen, ob die Befehle auch sinnvoll verwendet wurden. Wenn das richtige Programm lautet

```
import numpy as np                  # Fundamental package for working in Python
import matplotlib.pyplot as plt     # Package for Plots

x = np.linspace(1,15,8)
y = 2 - np.log(x)
plt.plot(x,y)
```

und nach den Befehlen linspace, log und plot gefragt wird, dann würde auch ein „Programm"

linspace-log-plot

die volle Punktzahl ergeben. Dies ist wenig sinnvoll.

Deswegen sollte die Methode in a) immer ergänzt werden durch numerische Felder. In diese kann z.B. der dritte Wert von x oder der erste negative Wert von y eingegeben werden. Die Aufgabenstellung würde also auch verlangen, den Wert x(3) und den ersten negativen Wert in zwei numerische Felder einzugeben. Damit wird sichergestellt, dass linspace oder log richtig verwendet werden.

Die obige Beispielaufgaben sind ebenfalls in ilias implementiert und können **öffentlich** (mit ilias-Berechtigung) eingesehen werden.

c) Ein tatsächliches Beispiel

Da das bisherige Beispiel eher abstrakter Natur war, erfolgt nun eine Beispielaufgabe, wie sie im Studiengang Wirtschaftswissenschaften im 2. Jahr des Bachelors tatsächlich gestellt werden kann.

Die Aufgabenstelle laute: Betrachten Sie ein matching-Modell der Arbeitslosigkeit. Die Arbeitslosenquote u(t) zu einem Zeitpunkt t>0 sei durch die folgende Gleichung bestimmt,

u(t) = u* + (u0-u*) e^-(s+mu) t

Dabei ist u0 die anfängliche Arbeitslosenquote zum Zeitpunkt 0, s die Separationsrate, mu die Kontaktrate („matching rate“) und

u* = s/(s+mu)

die langfristige Arbeitslosenquote. Die Separationsrate sei 5% (also 0,05) pro Jahr, die Matchingrate sei 15% pro Monat (also 1,8 pro Jahr). Die anfängliche Arbeitslosenquote sei 5%.

(i) Berechnen Sie die langfristige Arbeitslosenquote u*. Kopieren Sie Ihren Pythoncode in das Freifeld und das Ergebnis in das numerische Feld.

(ii) Berechnen Sie die Arbeitslosenquote nach 3 Jahren. Kopieren Sie Ihren Pythoncode in das Freifeld und das Ergebnis in das numerische Feld.

(iii) Zu welchem Zeitpunkt t ist die Arbeitslosenquote bei 1%? (Antwort ohne zusätzliche Berechnung in Python möglich.) Wählen Sie aus den folgenden Optionen aus.
() nie
() nach einer zufälligen Reduktion der Arbeitslosenquote auf 1%
() nach etwa 8,2 Jahren
() im langfristigen Gleichgewicht, also wenn die Zeit gegen unendlich geht

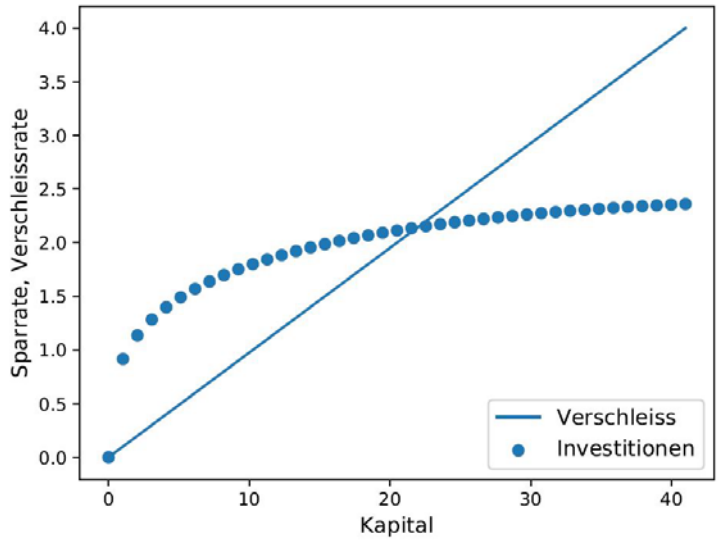
[Bewertung: 1 Punkt, 0,5 Punkte, 0 Punkte, 0 Punkte, falls dies leicht umsetzbar ist in Jogustine, sonst 1,1,0,0]

Diese Beispielaufgabe ist ebenfalls in ilias implementiert und kann **öffentlich** (mit ilias-Berechtigung) eingesehen werden.

Methode 1: Sortieren von Programmbestandteilen

Hier ist ein Beispiel eines Aufgabentyps, der so schon vielfach verwendet wurde und auch weiterhin verwendet werden wird. Die Darstellung auf dieser Seite ist dem Aussehen der Aufgabenstellung in ilias sehr ähnlich.

Da Sie die Dynamik des Kapitalbestandes im Solow Wachstumsmodell nicht nur mathematisch, sondern auch graphisch darstellen wollen, haben Sie mit Hilfe von Python folgendes Phasendiagramm erstellt.



Als Sie stolz Ihren Kommilitonen davon berichten und Ihre Grafik zeigen möchten, stellen Sie fest, dass beim zugrundeliegenden Matlab-Code etwas durcheinandergeraten ist.

Gehen Sie davon aus, dass Sie nur folgenden Code vorliegen haben.

A	Investitionen = np.array(Investitionen)
B	Verschleiss = np.array(Verschleiss)
C	for K in range(0, t+1):
	Y = A*K**alpha * L**(1-alpha)-d*K
	investitionen_help = s*Y
	verschleiss_help = d*K
	Investitionen.append(investitionen_help)
	Verschleiss.append(verschleiss_help)
D	import numpy as np
	import matplotlib.pyplot as plt
E	Investitionen = []
F	Verschleiss = []
G	t = 40
	L = 10
	A = 1
	s = 0.2
	d = 0.1
	alpha = 0.33
H	plt.gca().legend(('Verschleiss', 'Investitionen'), loc='lower right', fontsize=12)
	plt.xlabel('Kapital', fontsize=12)
	plt.ylabel('Sparrate, Verschleissrate', fontsize=12)
I	plt.show()
J	ax.scatter(t, Investitionen)
	ax.plot(t, Verschleiss)
K	fig1 = plt.figure()
	ax = plt.axes()
	t = np.linspace(0, t+1, t+1)

Bringen Sie den Code anhand der einzelnen Abschnitte in die richtige Reihenfolge, sodass Sie wieder Ihre ursprüngliche Grafik erhalten.

- D, G, E, F, C, A, B, K, I, J, H
- D, G, E, F, C, A, B, K, J, H, I
- B, A, F, G, C, D, E, K, I, J, H
- E, F, C, B, K, J, H, I, A, D, G
- A, B, E, F, C, D, G, I, J, H, K
- D, A, B, C, G, E, F, K, H, I, J
- D, K, J, H, I, G, E, F, C, A, B

Methode 2: Codeausgabe bestimmen

Gegeben sei folgender Python Code:

```
1   w = 1
2   gamma = 0.5
3   theta = 0.5
4
5   for time_endowment in range(14, 24, 2):
6
7       if time_endowment in range(14, 22, 2):
8
9           leisure = 1 / (1 + (gamma / (1 - gamma)) ** (1 / (1 - theta)) * w ** (theta / (1 - theta))) * time_endowment
10
11          work = time_endowment - leisure
12
13          print(work)
14
15      else:
16          print('Working so many hours is not healthy.')
```

Welche Ausgabe erzeugt dieser Python-Code?

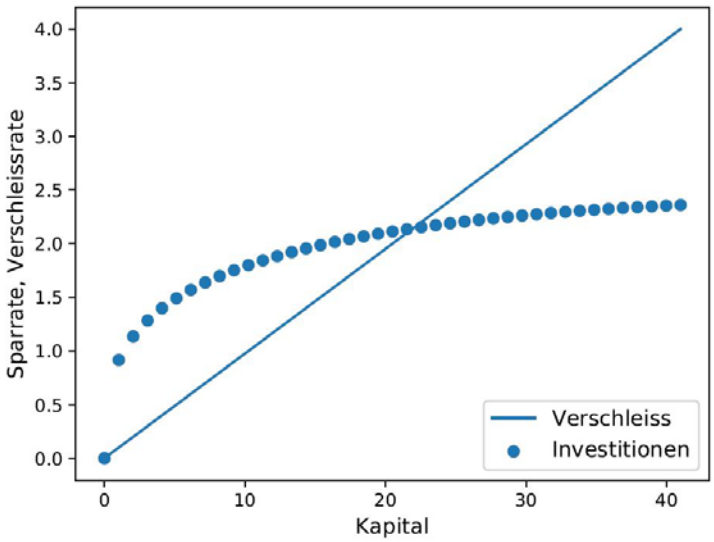
Richtige Antwort:

7.0
8.0
9.0
10.0
Working so many hours is not healthy.

Bemerkung: Diese Aufgabe ist etwas rechenintensiv für Klausurteilnehmer. Es muss die Gleichung in Zeile 9 viermal berechnet werden. Da die Parameterwerte für gamma und theta jedoch „rechenfreundlich“ gewählt wurden, hält sich der Aufwand in Grenzen. Der Rechenaufwand lässt sich reduzieren, wenn die obere Grenze (von 22) in Zeile 7 abgesenkt wird.

Methode 3: Vervollständigen eines Programmes

Gegeben sei folgende Grafik



Erzeugen Sie einen Python Code, welcher Ihnen diese Grafik ausgibt.

(Hinweis: Wählen Sie im Drop-Down-Menü den passenden Befehl/ die korrekte Variable aus.)

```
import numpy as np
import matplotlib.pyplot as plt

t = 40
L = 10
A = 1
s = 0.2
d = 0.1
alpha = 0.33

Investitionen = []
Verschleiss = []

for K in range(0, t+1):

    Y = A*K**alpha * L**(1-alpha)-d*K
    investitionen_help = s*Y
    verschleiss_help = d*K
    Investitionen.append(investitionen_help)
    Verschleiss.append(verschleiss_help)

Investitionen = np.array(Investitionen)
Verschleiss = np.array(Verschleiss)

fig1 = plt.figure()
```

```
ax = plt.axes()
t = np.linspace(0, t+1, t+1)
ax.scatter(t, Investitionen)
ax.plot(t, Verschleiss)

plt.gca().legend(('Verschleiss', 'Investitionen'), loc='lower right', fontsize=12)
plt.xlabel('Kapital', fontsize=12)
plt.ylabel('Sparrate, Verschleissrate', fontsize=12)
plt.show()
```

Ideen für Drop-Down-Menü:

			weitere Auswahlmöglichkeiten
Lücke	**1**	**[]**	(), {}, $$, ''
Lücke	**2**	**[]**	(), {}, $$, ''
Lücke	**3**	**for**	while, if, elif, else
Lücke	**4**	**K**	t, L, s, d
Lücke	**5**	**in range(0, t+1)**	(0, t), [0, t], in range[0, t+1]
Lücke	**6**	**np.array**	np.matrix, np.list, np.dict,np.dataframe
Lücke	**7**	**np.array**	np.matrix, np.list, np.dict,np.dataframe
Lücke	**8**	**np.linspace**	np.array, np.linespac, np.range, np.frame
Lücke	**9**	**xlabel**	label, x_axis, x.axislabel, xaxis
Lücke	**10**	**ylabel**	label, y_axis, y.axislabel, yaxis

[Richtige Antworten sind mit rot gekennzeichnet.]